REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie
****
UNIVERSITE DE YAOUNDE I
****
CENTRE DE RECHERCHE ET DE FORMATION DOCTORALE EN SCIENCES, TECHNOLOGIES ET GEOSCIENCES
****
UNITE DE RECHERCHE ET DE FORMATION DOCTORALE EN MATHÉMATIQUES, INFORMATIQUES, BIOINFORMATIQUE ET APPLICATIONS

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
****
UNIVERSITY OF YAOUNDE I
****
POSTGRADUATE SCHOOL OF SCIENCES, TECHNOLOGY AND GEOSCIENCES
****
RESEARCH AND POSTGRADUATE TRAINING UNIT OF MATHEMATICS, COMPUTER SCIENCES AND APPLICATIONS

DÉPARTEMENT DE MATHÉMATIQUES
DEPARTMENT OF MATHEMATICS

LABORATOIRE D'ANALYSE ET APPLICATIONS
LABORATORY OF ANALYSIS AND APPLICATIONS

PARTIAL DIFFERENTIAL EQUATIONS AND NUMERICAL ANALYSIS ON CURVED SURFACES

Ph D THESIS

IS THE FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY (Ph.D)

Speciality : Analysis

Option : Numerical analysis of PDEs

By :

NGUEMFOUO Marcial
MASTER IN MATHEMATICS
REGISTRATION NUMBER: **O5R457**

UNDER THE DIRECTION OF :

NNANG HUBERT
PROFESSOR
UNIVERSITY OF YAOUNDE 1

AND

NDJINGA MICHAEL
RESEARCH ENGINEER
UNIVERSITY OF PARIS SACLAY
CEA, France

Année académique 2024/2025

UNIVERSITÉ DE YAOUNDÉ I
****
FACULTÉ DES SCIENCES
****
DÉPARTEMENT DE MATHÉMATIQUES

THE UNIVERSITY OF YAOUNDE I
****
FACULTY OF SCIENCE
****
DEPARTMENT OF MATHEMATICS

ATTESTATION DE CORRECTION DE MÉMOIRE DE THÈSE DE DOCTORAT/Ph.D DE

Monsieur NGUEMFOUO Marcial

TITRE DE THÈSE : **Partial Differential Equations and Numerical Analysis on Curved Spaces.**

Nous soussignés, enseignants ci-dessous nommés, membres du jury de soutenance de thèse de Doctorat/Ph.D de **Monsieur NGUEMFOUO Marcial**, Matricule **05R457**, attestons que ce candidat a bel et bien pris en compte dans la mouture finale de sa thèse, toutes corrections et recommandations qui lui ont été faites au cours de sa soutenance en date du 23 Mai 2025.

En foi de quoi, la présente attestation de correction lui est délivrée pour servir et valoir ce que de droit.

Fait à Yaoundé, le

Le Jury:

Fonction	Nom et Prénom, Titre et Affiliation	Signature
Président du Jury	ETOUA Rémy Magloire Dieudonné, *Professeur* *(Université de Yaoundé I)*	
Rapporteurs	NNANG Hubert, *Professeur* *(Université de Yaoundé I)*	
	NDJINGA Michaël, *Chargé de recherche* *(Université de Paris-Saclay, France)*	
Examinateurs	HOFFMANN Franca, *Maître de conférences* *(California Institute of Technology, USA)*	
	MBEHOU Mohamed, *Maître de conférences* *(Université de Yaoundé I)*	

♣ Contents ♣

♣ List of Figures ♣

♣ Listings ♣

♣ Dedication ♣

To my mum Marie Mafouo who is the inspiration of each success I have ever had.

♣ Acknowledgements ♣

Above all, I offer my deepest gratitude to Almighty God for His boundless grace, strength, and guidance throughout my PhD journey. His unwavering support and divine wisdom have been my foundation, enabling me to overcome challenges and achieve this milestone.

I am deeply grateful to numerous individuals and institutions whose unwavering support, guidance, and encouragement have made this PhD journey possible.

First and foremost, I extend my heartfelt gratitude to my supervisors, Professor Hubert Nnang and Dr. Michaël Ndjinga, for their continuous support throughout my doctoral studies and research. Their patience, motivation, and profound expertise have been invaluable. Their insightful guidance not only shaped my research but also inspired me during the writing of this thesis. I could not have hoped for better advisors and mentors.

I am also immensely thankful to the Analysis Laboratory committee for their thoughtful comments, encouragement, and challenging questions, which broadened my research perspectives. My sincere appreciation goes to Professor Gabriel Nguetseng, Professor Marcel Dossa, Professor Norbert Noutchegueme, Professor Raoul Ayissi, Professor Pierre Noundjeu, Professor David Tegankong, Professor David Bekole, Professor Fidèle Ciake, Professor Etienne Takou, Professor Edgar Tchoundja, Professor Mohamed Mbehou, Professor Blaise Tchapnda, Dr. Eric Tetsadjo, Dr. Gilbert Chendjou, Dr. Hermann Douanla, and Dr. Marcel Fokam for their contributions.

My gratitude extends to the members of my doctoral committee for their valuable suggestions and encouragement, which greatly enriched my work. I also wish to express my deep appreciation to the faculty and staff of the Department of Mathematics for their kind support and assistance throughout my studies.

I am profoundly grateful to the University of Yaoundé 1 for providing an enabling academic environment that fostered my growth as a researcher. Special thanks go to the Rector, Professor Etoua Rémy Magloire Dieudonné, for his visionary leadership and commitment to academic excellence, which have been instrumental in creating opportunities for students like me to thrive.

I also acknowledge the Ndjinga Workshop for providing high-quality facilities and equipment, which significantly enhanced the quality of my research. My thanks go to all those involved in making these resources available.

To my fellow lab mates, I owe a debt of gratitude for the stimulating discussions, collaborative efforts, and camaraderie during late-night work sessions before deadlines. Your companionship made the challenges of this journey more bearable and memorable.

I am eternally grateful to my family for their unwavering support, patience, and belief in my dreams. Their encouragement has been the cornerstone of my success, and I dedicate this thesis to my parents, the most important people in my life, whose sacrifices and love have carried me this far.

My Christian family, including the Movement of the Lovers of Jesus, EEC Church, and Campus Crusade for Christ, deserves special recognition for their spiritual support and prayers, which sustained me throughout this journey.

Finally, to Pascaline Nguemfouo, Emeth Daniel Nguemfouo I owe more than words can express. Your love, encouragement, and presence have given this achievement its true meaning. Without you, this PhD would not hold the same significance.

♣ Abstract ♣

The objective of this thesis is the numerical study and simulation of partial differential equations of the elliptic type and their numerical analysis on differential and Lipschitz manifolds. This led to numerical simulations showing the potential applications of the subject which range from heat diffusion to road traffic modeling.

Firstly, the rigorous definition of a PDE on a given differentiable and Lipschitz manifold, a theory of existence of solutions to such a PDE and their numerical approximation are given. Precisely, we applied a P_1 finite element discretization to the Poisson problem for the Laplace-Beltrami operator on 2D surfaces immersed in $\mathbb{R}^3$ without boundary. Following Dziuk [16] we recalled existence and uniqueness results for continuous and discrete problems. We then investigated the condition number of the finite element operator. We gave an upper bound for the condition number that is similar to the Euclidean case.

Secondly We study the weak formulation of the Poisson problem on closed Lipschitz manifolds. Lipschitz manifolds do not admit tangent spaces everywhere and the definition of the Laplace-Beltrami operator is more technical than on classical differentiable manifolds [22]. They however arise naturally after the triangulation of a smooth surface for computer vision or simulation purposes. We derive Stokes' and Green's Theorems as well as a Poincaré's type inequality on Lipschitz manifolds. The existence and uniqueness of weak solutions of the Poisson problem are given in this new framework for both the continuous and discrete problems. As an example of application, numerical results are given for the Poisson problem on the boundary of the unit cube.

♣ Résumé ♣

L'objectif de cette thèse est l'étude et les simulations numériques des équations aux dérivées partielles de type elliptique et leur analyse numérique sur des variétés différentielles et Lipschitziennes. Le travail a abouti à des simulations numériques montrant les applications potentielles du sujet qui vont de la diffusion de la chaleur à la modélisation du trafic routier.

Premièrement, la définition rigoureuse d'une EDP sur une variété différentiable, la théorie de l'existence de solution à une telle EDP et leur approximation numérique sont données. Nous avons appliqué une discrétisation éléments finis P_1 au problème de Poisson pour l'opérateur de Laplace-Beltrami sur les surfaces compactes 2D immergées dans $\mathbb{R}^3$. Grâce à Dziuk [16], nous avons rappelé l'existence et l'unicité de solutions pour les problèmes continus et discrets. Nous avons ensuite investigué le conditionnement de l'opérateur élément fini. Nous avons ainsi donné une borne supérieure du conditionnement qui est similaire au cas Euclidien.

Deuxièmement Nous avons étudié la formulation faible du problème de Poisson sur les variétés Lipschitziennes fermées. Les variétés Lipschitziennes n'admettent pas d'espaces tangents partout et la définition de l'opérateur de Laplace-Beltrami est plus technique que sur les variétés différentiables classiques [voir, par exemple, 22]. Ils apparaissent cependant naturellement après la triangulation d'une surface lisse à des fins de vision par ordinateur ou de simulation. Nous avons proposé une preuve des théorèmes de Stokes et de Green ainsi qu'une inégalité de type Poincaré sur les variétés Lipschitziennes. L'existence et l'unicité d'une solution faible du problème de Poisson est donnée dans ce nouveau cadre pour les problèmes continus et discrets. A titre d'exemple d'application, des résultats numériques sont donnés pour le problème de Poisson sur la frontière du cube unité.

♣ Introduction ♣

Partial differential equations (PDEs) on manifolds appear in a variety of problems and applications in fluid dynamics, materials science, solid mechanics, biology and image processing. As examples are the modeling of interfaces in multiphase fluid flows and the modeling of surface active agents (surfactants). The theory of linear elliptic equations has been successfully transposed from the Euclidean spaces [3, 10] to curved spaces [28, 42, 46]. A complete theory of existence and uniqueness using the same functional analysis tools (e.g., Lax- Milgram Theorem) is now available to deal with elliptic equations on manifolds. Even the spectral properties of the Laplacian have been extensively studied (see, e.g., [11]). The eigenvalues of the Laplacian on a manifold allow the recovering of some important information about the manifold. For instance, the Weyl's law [44, Chapter 11] expresses the volume of a manifold in terms of asymptotic behavior of its Laplacian eigenvalues. However there is not enough information in the spectrum to obtain an entire characterization of the manifold. Indeed the spectral theory of manifolds culminated in 1992 with a negative answer to the question: can we hear the shape of a drum ?(See, [25]), even though the 2D counter example consists of non convex sets. Analysis of PDEs and numerical analysis on manifolds yield numerous applications in engineering, biology and environmental sciences through the ability to model and simulate numerically complex systems on computers. Besides yielding numerous concrete applications, the ability to build and manipulate manifolds on a computer as well as visualizing fields on it has the immense advantage of helping the building of a better intuition of the mathematical objects, testing empirically open conjectures and communicating visually new ideas and results. Recent advances in computational power as well as tools designed for the meshing of complex surfaces, the generation and manipulation of large matrices, allow now an environment to validate the theory and explore new conjectures. We have used the open source platform Salome [7, 41, 52] for the Computer aided design of our manifold (GEOM), the meshing of the shape (SMESH), the handling of the discrete fields (MED module) and the visualization of the numerical results (Paravis module).

The primary objective of this thesis is to investigate the influence curvature has on the condition number of the finite element method in the particular case of the Laplace-Beltrami operator. We therefore study the finite element approximation of the Poisson problem

$$-\Delta_\Gamma u = f \text{ on } \Gamma, \tag{1}$$

where $\Gamma \subset \mathbb{R}^3$ is a closed surface. f is a given $L^2(\Gamma)$ source with zero mean for the problem to admit a unique solution and u being the unknown solution.

Since $-\Delta_\Gamma$ is symmetric positive definite then Lax-Milgram's Theorem provides the existence, uniqueness and stability of solution on mean-zero functions in $H^1(\Gamma)$; the article [15] adapts the classical Euclidean finite element approach in order to deal with curved surfaces. Existence and asymptotic error estimates are proven by Dziuk and Elliott [16]. An important feature of the method described in [15] is the avoidance of charts both in the problem formulation and the numerical method. The surface finite element method is based simply on triangulated surfaces and requires

the geometry solely through knowledge of the vertices and normal vector of each triangle. In the case of a closed surface the domain Γ is first approximated by the surface of a polyhedron Γ_h using a Delaunay triangulation (See, i.e., [30, 53]) described in Chapter 3. The finite elements (i.e., the 3D triangles) are still planar as in the Euclidean case, but they are no longer parallel. The source f is approximated by a function $f_h \in L^2(\Gamma_h)$ with zero mean.

The solution u of (1) on Γ is then approximated by the solution u_h of the following Poisson problem on Γ_h

$$-\Delta_{\Gamma_h} u_h = f_h \text{ on } \Gamma_h, \tag{2}$$

where $\Gamma_h \subset \mathbb{R}^3$ is a closed piecewise triangular surface . Γ_h being a closed manifold, the source f_h must have zero mean for (2) to admit a unique solution ($\int_{\Gamma_h} f_h = 0$).

The finite element method proposed in [15] then consists in approximating $u_h \in H^1(\Gamma_h)$ by its projection $\tilde{u}_h$ on $PL(\Gamma_h)$, the subspace of $H^1(\Gamma_h)$, of continuous piecewise linear functions. Doing so, the partial differential operator Δ_{Γ_h} is approximated by a finite dimension linear operator with matrix $A_{\triangle_{\Gamma_h}}$ and f_h is approximated by the one column matrix b_h. This approximation uses the variational formulation of (2) and reduces the Poisson problem (1) to the resolution of a linear system

$$A_{\triangle_{\Gamma_h}} X = b_h, \tag{3}$$

where X and b_h are vectors with zero mean.

The numerical error generated by the resolution of the linear system (3) can be bounded using an upper bound of the condition number $\mathcal{K}(A_{\triangle_{\Gamma_h}})$. For this reason, and though it is not possible in practice to give an exact expression of $\mathcal{K}(A_{\triangle_{\Gamma_h}})$, we will analyze to obtain an upper bound. The condition number of the finite element matrix has been extensively studied in the Euclidean case [19, 20]. However few works exist in the case of the Laplace-Beltrami operator on curved surfaces, particularly when they are closed.

Moreover, classical preconditioners are based on Incomplete LU factorizations (See [43, Section 10.3.1]), which fails if the matrix is not an M-matrix (See [43, Theorem 10.2]), in particular if the matrix is singular (See [43, Definition 1.4]). Yet, the discretization of PDEs on closed surfaces yields singular matrices, hence the resolution of the associated linear systems are restricted to meshes with a small number of elements. There is therefore an important need for *ad hoc* preconditioners for the numerical simulation of PDEs on closed surfaces.

The first step in this project is to give an estimate of the condition number of the finite element matrix.

There are several approaches to finite elements on manifolds (for a review, see [16] and the references therein). One approach uses a family of local charts of the manifold (See, e.g., [38]). Another approach uses a global parametrization of the manifold (see, e.g., [33]). A third approach uses a meshing of the bulk space containing the surface and intersects it with the surface in order to obtain a discrete problem [14].

The first two approaches are difficult to carry numerically for a general surface since local charts and global parametrization are complex and computationally expensive. The third approach is computationally expensive because of the meshing of the bulk space and the fact that the nodes of the approximation are not on the surface may generate approximation errors. We have chosen the approach of [15] which triangulates the surface with nodes belonging to the surface.

One technical difficulty is that the finite element matrix is not invertible on $\mathbb{R}^3$ since the domain has no boundary. The finite element operator is however invertible on the space of zero mean vectors. The other technical difficulty is the handling of different normal vectors arising from the non zero curvature.

As our first contribution, we are still able adapt the Euclidean approach following Ern and Guermond [19] to derive an upper bound of the condition number.

Our second contribution comes from the observation that Dziuk [15] defined the differential gradient, divergence and Laplace operators on supposedly differentiable surfaces, and even C^2- surfaces. However, the first step of its finite element approximation consists in approximating the C^2-surface by a polyhedron, and replacing the initial Poisson problem on the C^2- manifold by a Poisson problem on a polyhedron. The geometric framework proposed by Dziuk [15] therefore seemed incomplete to us. Another consequence of the surface regularity assumption C^2 is that his work does not apply to non-regular surfaces such as the cube skin. We have therefore proposed a more general framework for dealing with non-differentiable surfaces and which can be applied to polyhedra. We have for these reasons proposed the geometric framework of Lipschitz manifolds introduced in [22, 24] and have shown the well-posed character of the Poisson problem in this framework. The finite element approximation of the Poisson problem on a Lipschitz manifold therefore no longer requires the C^2 character of the manifold. We have shown the advantages of this new framework by presenting simulations of the Poisson problem on the surface of a cube. After determining the exact solution, we observed the convergence to order 2 of the finite element approximation. This therefore confirms that there is no theoretical or practical obstacle to the approximation of a non-differentiable manifold.

Outline of the thesis

This thesis is structured into four chapters, with research contributions focused within all these four chapters:

In the first Chapter, we provide, following Dziuk [15, 16], the notion of differential geometry, the differential calculus on smooth hypersurfaces. The definition of the differential operators on hypersurface relies on the so-called Fermi-Coordinates, Fermi coordinates are global coordinates on hypersurfaces Γ that avoid working with charts. Finaly we formulate an example of PDEs on closed surfaces and obtain their variational formulation thanks to Green's Theorem, the existence, uniqueness and stability of solution for Poisson problem for the Laplace-Beltrami operator via the Lax-Milgram theorem. The chapter ends with a few notion of eigenvalue problem.

The second Chapter is concerned with the Poisson Problem on closed Lipschitz manifolds. We introduce the notion of Lipschitz manifolds. Lipschitz manifolds are an intermediate structure between topological manifolds where the tangent space is defined nowhere and C^1 manifolds where the tangent space is defined everywhere. Differentiability almost everywhere thanks to Rademacher's Theorem is the key ingredient of the calculus on Lipschtz manifolds. On Lipschitz manifolds the tangent space is defined almost everywhere, which is enough to define spaces of functions that are weakly differentiable. Following [22], we define Lebesgue L^p spaces, Sobolev $W^{1,p}$ and $W^{-1,p}$ spaces, the gradient, divergence and Laplace-Beltrami operator on Lipschitz manifolds. After defining the measurable and Lipschitz differential forms, we propose a new Stokes'Theorem as well as Green's Theorem in order to obtain the weak formulation of Poisson problem and therefore the existence, uniqueness and stability of solutions thanks to the Lax-Milgram's Theorem.

Chapter three is devoted to the surface finite element methods and the theoretical estimate of the condition number. We first recall the basic notions of Galerkin finite element method, and introduce the finite elements method thanks to Dziuk [15, 16]. We adapted the technique used by Ern and Guermond [20] to derive the theoretical upper bound of condition number for the Laplace-Beltrami operator. This condition number is important to quantify the error propagation during computer resolution of the linear system resulting from finite element approximation. We adapted the technique used by Ern and Guermond [20] in the Euclidean frame to the curved frame. The

expression is similar to the condition number of the Laplace operator in the Euclidean case, with the curvature affecting the condition number through the Poincaré's constant. However in the case of closed surfaces one has to deal firstly with the fact that the finite element matrix is singular, secondly with the presence of curvature field.

Finally in Chapter four, we present the numerical results for the different Poisson problems on closed smooth manifolds(Sphere, Torus) and on Lipschitz manifold (Cube skin). We compare our result with the theoretical solutions and we plot the convergence rate as compared with the theoretical error estimates in each case. More precisely, for the design and meshing of the domain we use GEOMETRY and MESH modules of the software SALOME (See [41, 52]). For the coding of the script, we use Python with the open-source Linux based library SOLVERLAB [60] which is very practical for the manipulation of large matrices, vectors, meshes and fields. It (SOLVERLAB) can handle finite element and finite volume discretizations, read general $1D, 2D$ and $3D$ geometries and meshes generated by SALOME. For the numerical resolution of our discrete problem, we use an iterative solver because the stiffness matrix $A_{\triangle_{\Gamma_h}}$ is large, sparse (See [43]) and singular. The library PETSc [6], encapsulated in SOLVERLAB, provides linear solvers for singular systems. For the visualization of the result, we use the PARAVIS module included in SALOME (See [52, 59]).

The thesis end with a general conclusion and further possible studies where we summarize our thesis and provide new investigations of our work.

The Contributions of the thesis

The thesis contains the following contributions:

- Geometric frame work for Lipschitz manifols.
- Statement and proof of Stokes'Theorem 2.3 on Lipschitz manifolds.
- Statement and proof of Green's Theorem 2.4 on Lipschitz manifolds as well as Poincaré's Inequality type Lemma 2.2.
- Weak formulation of the Poisson problem on Lipschitz manifolds.
- Statement and proof of the existence, uniqueness and stability of solution of (1) thanks to the Lax-Milgram Lemma 1.6.
- Estimation of the upper bound of the condition number for the Laplace-Beltrami operator Section 3.3.
- Numerical resolution of the Poisson problem on closed manifolds (smooth and Lipschitz surfaces) Chapter 4.
- Numerical result for the condition number Subsection 4.5.4.

Publications from the thesis

The articles from this thesis are:

1. Marcial Nguemfouo, Michael Ndjinga: On the condition number of the finite element method for Laplace Beltrami operator. *J. Elliptic. Parabol. Equ. (2023).https://doi.org/10.100/s41808-023-00251-7*[36]

2. Michael Ndjinga, Marcial Nguemfouo: Well posedness for the Poisson problem on closed Lipschitz manifolds. *Partial. Differ. Equ. Appl. 4, 44 (2023). https://doi.org/10.1007/s42985-023-00263-x*[35]

⋆ ⋆ CHAPTER ONE ⋆ ⋆

THEORETICAL STUDY OF POISSON PROBLEM ON CLOSED C^2 SURFACE

Contents

1.1 Preliminaries

We introduce a few concepts on differential geometry, which will be used in the sequel. The notion of differential geometry is recalled in Subsection 1.1.1and those of differential operators in Subsection 1.1.2 which is important for example to verify that the source function f leads to the exact solution in the Laplace-Beltrami equation on sphere and on torus.

1.1.1 Some notions of differential geometry

This Subsection is devoted to the presentation of geometry notions. We will use in the sequel, including a brief introduction to Riemannian geometry and the definition of differential operators on it.

Let X be a topological space. We will say that X is a n-dimensional topological manifold if each point of X has a neighborhood homeomorphic to an open subset of $\mathbb{R}^n$. Clearly speaking, a topological space X is a n-dimensional topological manifold if given $x \in X$ there exists an open neighborhood U_x of x in X and a mapping $\varphi_x : U_x \to \mathbb{R}^n$ such that $\varphi_x : U_x \to \varphi_x(U_x)$ is an homeomorphism. This being so, (U_x, φ_x) is called a (local) chart with domain U_x. A collection of charts covering X is an atlas. If (U, φ) and (V, ψ) are two local charts on X and $U \cap V$ is non empty, we define the transition function $\varphi \circ \psi^{-1} : \psi(U \cap V) \to \mathbb{R}^n$. If all the transition functions are k-times continuously differentiables, the manifold X is said to be a $\mathcal{C}^k$-differential manifold.

If X is a n-dimensional manifold and X is (viewed as) a subset of $\mathbb{R}^{n+1}$, then X is (viewed as) an hypersurface of $\mathbb{R}^{n+1}$; when $n = 2$, X is (viewed as) a surface.

Let X be a n-dimensional manifold. A point $x \in X$ is said to be a boundary point of X if there exists an open neighborhood W of x that is homeomorphic to $\mathbb{R}^n_+$ where x is mapped to the origin of $\mathbb{R}^n$.

A n-dimensional closed manifold is any n-dimensional manifold X that contains no boundary points.

Remark 1.1.
A closed surface is simply (viewed as) a closed two-dimensional manifold.

An immersion is a differentiable mapping between two differentiable manifolds such that its derivative is everywhere injective.

Let $i : X \to Y$ be an immersion. we will say that i is an embedding if i is a homeomorphism from X onto $i(X)$.

Some examples of surfaces are given by Figure 1.1.1

- Examples of closed surfaces are: the sphere, the torus, and the surface of a cube.
- Examples of non closed surfaces are: disk, square and hemisphere surface.

1.1.2 Differential operators on manifolds

In standard calculus, the operators like gradient, divergence and Laplace are classically defined on the Euclidean space $\mathbb{R}^n$. They can also be extended on a class of manifolds.

A Riemannian manifold is a pair (M, g) where:

- M is a C^∞ manifold.
- g is a smoothly chosen inner product g_x defined on $T_xM \times T_xM$ with values on $\mathbb{R}$ for each $x \in M$ where T_xM is the tangent space on x, for the vector space of tangent vectors based on x.

Remark 1.2. *The dual space of T_xM is called the cotangent space to x at M and we denote it by $T^\star_xM$.*

More precisely, for $x \in M$, g_x satisfies:

1. g_x is symmetric, i.e., $g_x(u, v) = g(v, u)$ for all $u, v \in T_xM$
2. g_x is positive, i.e., $g_x(u, u) \geq 0$, for all $u \in T_xM$
3. g_x is definite, i.e., $g_x(u, u) = 0$ if and only if $u = 0$ (the null vector of T_xM

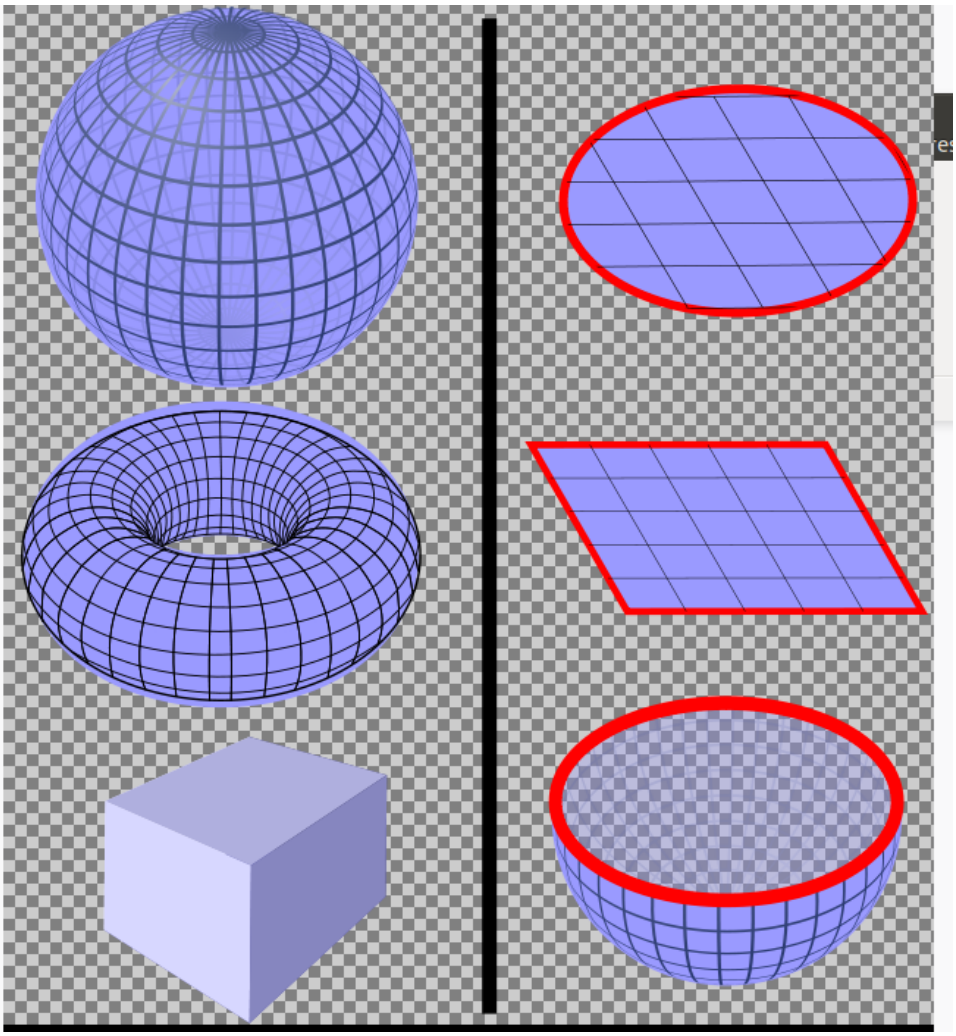


Figure 1.1: Examples of closed surfaces (Left) and non closed surfaces (right) from *https://en.wikipedia.org/wiki/File:SurfacesWithAndWithoutBoundary.svg*(2020 at 5 PM)

Let us note that g is smooth in the sens that for any smooth fields u and v, the function $x \to g_x(u_x, v_x)$ is smooth.

Example 1.1 (Examples of Riemannian manifolds)**.**
A open subset of $\mathbb{R}^n$*, a sphere* $\mathbb{S}^n$*, a Torus* $\mathbb{T}^n$

The Laplace operator Δ_M on a manifold M cannot be expressed by coordinates globally because the latter in general does not have a global coordinate. If $(U_x, \varphi_x = (x_1, \cdots, x_n))$ is a chart of M, a local expression for the Riemannian metric tensor g can be given as follows

$$g = \sum_{i,j=1}^{n} g_{ij} dx_i \otimes dx_j,$$

Let us express each symbol or term of the right hand side of this equality:

We begin by introducing the basis $\{(\frac{\partial}{\partial x_i})_x, 1 \leq i \leq n\}$ which is the natural basis of T_xM, then $\{(dx_i)_x, 1 \leq i \leq n\}$ is its dual basis associated to this chart (U_x, φ_x);

The matrix $(g_{ij}(x) = g_x((\frac{\partial}{\partial x_i})_x, (\frac{\partial}{\partial x_j})_x))_{n\times n}$ denoted G, is naturally symmetric and positive definite; its inverse G^{-1} is denoted by $(g^{ij})_{n\times n}$

$\otimes$ is a tensor product on $T_x^\star M$

This being so, we set $g = det(G)$ so that, the volume element in M is locally given by

$$dVol = \sqrt{|g|}dx_1 \otimes \cdots \otimes dx_n \equiv \sqrt{|det(G)|}dx_1 \otimes \cdots \otimes dx_n.$$

And operators started before are expressed as follows:

- Local gradient of $u \in C^1(M)$ as

$$\nabla_g u = \sum_{i,j=1}^{n} g^{ij}\frac{\partial u}{\partial x_j}\partial_{x_i} = \left(g^{1j}\partial_{x_j}u, g^{2j}\partial_{x_j}u, \cdots, g^{nj}\partial_{x_j}u\right) \equiv \left(\nabla_g^1 u, \nabla_g^2 u, \cdots, \nabla_g^n u\right)$$

  where $\partial_{x_i} = \frac{\partial}{\partial x_i}$

- Local divergence of a vector field $\mathbf{V} = \sum_{i=1}^n v_i\partial_{x_i} \in (\mathcal{C}^1(M))^n$ as

$$\text{div}_g\mathbf{V} = \frac{1}{\sqrt{|det(G)|}}\sum_{i=1}^{n}\partial_{x_i}(v_i\sqrt{|det(G)|}).$$

- Local Laplace-Beltrami operator as

$$\begin{aligned}
\Delta_g u &= -\text{div}_g(\nabla_g u) \\
&= -\frac{1}{\sqrt{|det(G)|}}\sum_{i=1}^{n}\partial_{x_i}\left(\nabla_g^i u\sqrt{|det(G)|}\right) \\
&= -\frac{1}{\sqrt{|det(G)|}}\sum_{i=1}^{n}\partial_{x_i}\left(g^{ij}\partial_{x_i}u\sqrt{|det(G)|}\right) \\
&= -\frac{1}{\sqrt{|det(G)|}}\sum_{i,j=1}^{n}\partial_{x_i}\left(g^{ij}\sqrt{|det(G)|}\partial_{x_j}u\right).
\end{aligned}$$

We chose for simplicity to define the differential operators on a closed embedded manifolds $\Gamma \subset \mathbb{R}^3$. This approach is also easier to handle from a practical point of view. For instance formula (1.5) is used in the determination of the finite element coefficient on each triangle (equation 3.22).

We define in the following an embedded surfaces, the tangential gradient operator ∇_Γ , the tangential divergence operator $\nabla_\Gamma\cdot$, and the the Laplace-Beltrami operator $\triangle_\Gamma$ as the composition of the tangential divergence operator and the tangential gradient operator

1.2 Calculus on hypersurfaces

The definition of differential operators on hypersurfaces require the so-called Fermi coordinates [16], inspired by the approach of Fermi normal coordinates around a curve Theorem 1.1. The Fermi coordinates are also useful in the proof of the Poincaré inequality (See [16, Theorems 2.8 and 2.12]) and for the existence of a polyhedral surface Γ_h that approximates the hypersurface Γ (Section 3.2.3).

We define embedded surfaces in Subsection 1.2.1, the tangential gradient ∇_Γ in Subsection 1.2.2, the tangential divergence $\nabla_\Gamma\cdot$ in Subsection 1.2.3, and then the Laplace-Beltrami operator $\triangle_\Gamma$ as the composition of divergence and gradient in Subsection 1.2.3.

1.2.1 Embedded surfaces and Fermi coordinates

Following [16], the definition of tangential gradient operator and the tangential divergence requires the definition of embedded surfaces and their normals.

Definition 1.1 (Embedded C^k-hypersurface)**.**
$\Gamma \subset \mathbb{R}^3$ *is called a* C^k*-hypersurface if for each point* $x_0 \in \Gamma$*, there exists an open set* $U_{x_0} \subset \mathbb{R}^3$ *containing* x_0 *and a function* $\phi_{x_0} \in C^k(U_{x_0})$ *with the following properties*

$$\begin{aligned} &\nabla\phi_{x_0} \neq 0 \ \ on\ \Gamma \cap U_{x_0} \\ &\Gamma \cap U_{x_0} = \{x \in U_{x_0}; \phi_{x_0}(x) = 0\} \end{aligned} \tag{1.1}$$

where ∇ is the classical $\mathbb{R}^3$ gradient.

The set of functions ϕ_{x_0} is called the level set functions of Γ at x_0.
For any C^1-hypersurface $\Gamma \subset \mathbb{R}^3$, the **unit normal** at any point $x \in U_{x_0}$ is defined as

$$\mathbf{n}(x) = \frac{\nabla\phi_{x_0}(x)}{||\nabla\phi_{x_0}(x)||}, \quad (x \in U_{x_0}) \tag{1.2}$$

Let's define ϵ-tubular neighborhoods (called ϵ-strip around Γ), see Figure 1.2:

$$U_{\epsilon,\Gamma} = \{x \in \mathbb{R}^3, dist(x, \Gamma) < \epsilon\}, \qquad (\epsilon > 0). \tag{1.3}$$

A function u defined on Γ can be extended on $U_{\epsilon,\Gamma}$ using the Fermi coordinates around Γ as follows: for $\epsilon > 0$ small enough, we define the projection $a_\epsilon : U_{\epsilon,\Gamma} \to \Gamma$ and the distance $d_\epsilon : U_{\epsilon,\Gamma} \to \mathbb{R}_+$ to Γ.

Theorem 1.1 (Fermi coordinates)**.**
Let Γ *be an embedded* C^2 *hypersurface of* $\mathbb{R}^3$*. There exists a tubular neighborhood of Fermi, there exists* $\epsilon_0 > 0$ *such that*

$$\begin{aligned} \forall x \in U_{\epsilon_F,\Gamma}, \exists! y = a_{\epsilon_0}(x) \in \Gamma \text{ and } d_{\epsilon_0} \in C^2(U_{\epsilon_F,\Gamma}) \quad \text{such that} \\ x = a_{\epsilon_0}(x) + d_{\epsilon_0}(x)\mathbf{n}(a_{\epsilon_0}(x)) \equiv a_{\epsilon_0}(x) + d_{\epsilon_0}(x)\mathbf{n}(x). \end{aligned} \tag{1.4}$$

Proof: See Lemma 2.8 in [16]

□

Which defines, for each $\epsilon \leq \epsilon_0$, two mapping : $a_\epsilon : U_{\epsilon,\Gamma} \to \Gamma$ and $d_\epsilon : U_{\epsilon,\Gamma} \to \mathbb{R}_+$; a_ϵ is projection operator from $U_{\epsilon,\Gamma}$ on Γ and d_ϵ is the distance of a point $x \in U_{\epsilon,\Gamma}$ to Γ.

In the following, the Fermi-coordinates will allow us to extend a function defined on Γ to a neighborhood of Γ, which will be useful in the definitions of tangential gradient and divergence (Subsections 1.2.2 and 1.2.3) as well as Laplace-Beltrami operator.

1.2.2 The tangential gradient operator

The notions of hypersurface (Definition 1.1) and associated normal vectors (1.2) defined in the previous subsection are useful in the following definition of the projected gradient operator.

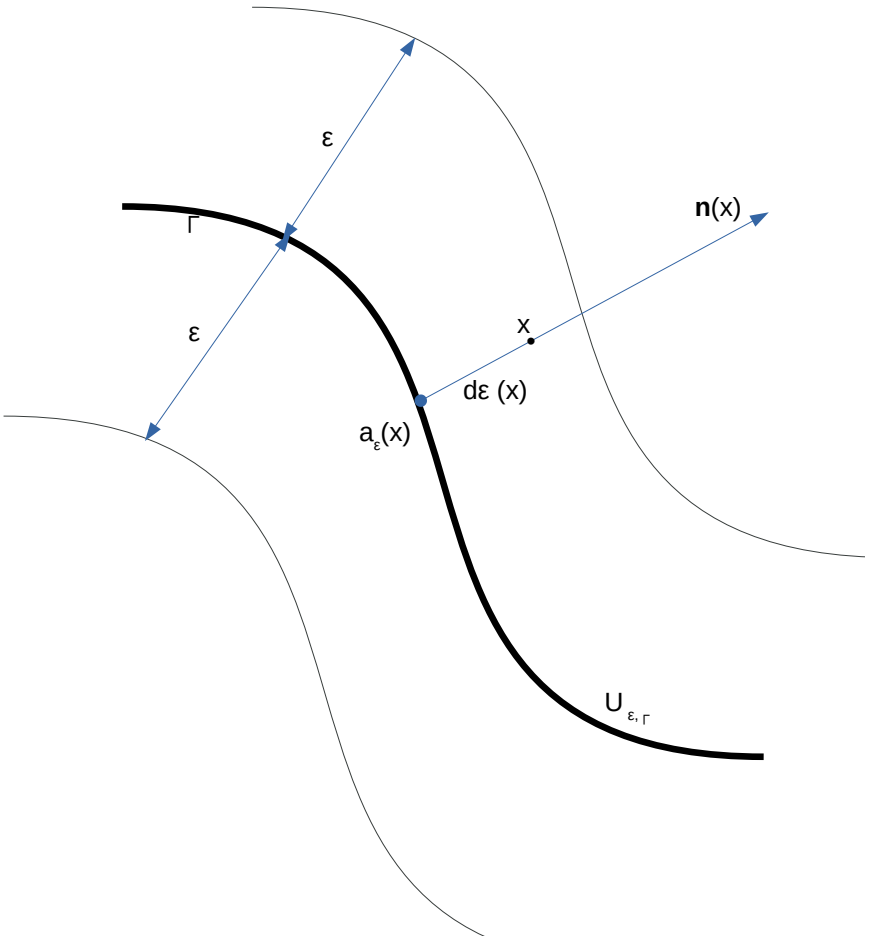


Figure 1.2: ϵ-strip around Γ

Definition 1.2 (Projected $\mathbb{R}^3$ gradient)**.**
Let Γ be an embedded C^1 hypersurface, $U_{\epsilon,\Gamma}$ a neighborhood of Γ in $\mathbb{R}^3$, and $u \in C^1(U_{\epsilon,\Gamma})$. The projected gradient operator of u on Γ is

$$\begin{aligned} \nabla_{P\Gamma} : C^1(U_{\epsilon,\Gamma}) &\rightarrow C(\Gamma)^3 \\ u &\rightarrow \nabla_{P\Gamma} u \end{aligned}$$

such that

$$\nabla_{P\Gamma} u = \nabla u|_\Gamma - (\mathbf{n} \cdot \nabla \mathbf{u}|_\Gamma)\mathbf{n}$$

From the projected gradient operator on $U_{\epsilon,\Gamma}$, one can define the tangential gradient operator on Γ as follows.

Definition 1.3 (Tangential gradient)**.**
Let Γ be an embedded C^2 hypersurface of $\mathbb{R}^3$. The tangential operator ∇_Γ is the mapping from $C^1(\Gamma)$ into $C(\Gamma)^3$ defined by

$$\nabla_\Gamma u = \nabla_{P\Gamma}\bar{u} \quad (u \in C^1(\Gamma)), \tag{1.5}$$

where $\bar{u} \in C^1(U_{\epsilon,\Gamma})$ such that $\bar{u}|_\Gamma = u$ on Γ.

Remark 1.3. *The i^{th} coordinate of $\nabla_\Gamma u$ is denoted by*

$$\underline{D}_i u, \quad i = 1, 2, 3$$

with

$$\underline{D}_i u = \partial_i \bar{u}|_\Gamma - \sum_{j=1}^{3} n_j \partial_j \bar{u}|_\Gamma n_i$$

The assumption that Γ be C^2 comes from the fact that the existence of an extension $\bar{u}$ of u on a neighborhood of Γ requires Fermi coordinates and Fermi coordinates are defined for C^2 manifolds. The existence of a function $\bar{u}$ in Definition 1.3 is a consequence of the existence of Fermi coordinates defined in the subsection 1.2.1 (Definition 1.1). Indeed, the function $\bar{u}$ can be defined for example as $\bar{u}(x) = u(a_\epsilon(x))$ where a_ϵ is the first Fermi coordinate (the projection operator).

The value of the tangential gradient operator of $\bar{u} \in C^1(U_{\epsilon,\Gamma})$ on an embedded hypersurface Γ does not depend on the choice of $\bar{u}$. It depends only on the values taken by u on Γ as expressed in the following lemma (See [16, Lemma 2.4]).

Lemma 1.1.
$\nabla_\Gamma f$ only depends on the values of f on Γ.

Proof: See Lemma 2.4 in [16].

□

One key property of the tangential gradient operator is the following. The tangential gradient operator belongs to the tangent plane and is orthogonal to the normal vector

$$(\nabla_\Gamma u) \cdot \mathbf{n} = 0.$$

Definition 1.4 (Curvature)**.** *For a C^2 embedded manifold Γ and a point $x \in \Gamma$, the curvature can be defined as*

$$\mathcal{H}(x) = \nabla \cdot \boldsymbol{n}(x), \text{ with } \quad \mathcal{H}_{ij}(x) = \underline{D}_i n_j(x)$$

- The matrix $\mathcal{H}$ is symmetric and possesses an eigenvalue 0 in the normal direction: $\mathcal{H} \cdot \mathbf{n} = 0$
- $\mathcal{H}$ is called extended Weingarten map
- For $x \in \Gamma$, the quantity

$$H(x) = \text{trace}\mathcal{H}(x) = \sum_{i=1}^{n+1} \mathcal{H}_{i,i}(x)$$

  is the mean curvature of Γ at the point x.

- The eigenvalues $\kappa_1, \cdots, \kappa_n$ of $\mathcal{H}$ (except 0 in the normal direction) are the principal curvatures of Γ.

Let us have a look at the most simple example.

The sphere of radius $r > 0$, $\Gamma_s = \{x \in \mathbb{R}^{n+1} || |x| = r\}$, is given by the level set function $\phi(x) = |x| - r$ for $0 < |x| < \infty$

$$\mathbf{n}(x) = \frac{\nabla \phi_x(x)}{||\nabla \phi_x(x)||} = \frac{x}{|x|}$$

and get for $x \in \Gamma$

$$\begin{aligned} \mathcal{H}_{ij}(x) &= \underline{D}_i n_j(x) \\ &= \underline{D}_i \frac{x_j}{|x|} \\ &= \frac{1}{r} \underline{D}_i x_j \\ &= \frac{1}{r}(\delta_{ij} - n_i n_j) \\ &= \frac{1}{r}(\delta_{ij} - \frac{x_i x_j}{r^2}) \end{aligned}$$

The matrix $(\mathcal{H}_{ij}(x))_{ij}$ has an eigenvalue 0 with eigenvector $\frac{x}{r}$ and n eigenvalues $\kappa_j = \frac{1}{r}$ $(j = 1, \cdots, n)$. The mean curvature is then given as $H = \frac{n}{r}$.

1.2.3 The tangential divergence operator

The tangential divergence operator on Γ is defined as the contribution to the full divergence arising from the tangent space to Γ. We first define the projected divergence operator on Γ as follows.

Definition 1.5 (Projected $\mathbb{R}^3$ divergence)**.**
Let Γ be an embedded C^1 hypersurface, $U_{\epsilon,\Gamma}$ a neighborhood of Γ in $\mathbb{R}^3$ in the sense of (1.2). The projected divergence operator on Γ is the operator denoted $div_{P\Gamma}$ and defined by:

$$\begin{aligned} div_{P\Gamma} : C^1(U_{\epsilon,\Gamma})^3 &\rightarrow C(\Gamma) \\ \mathbf{V} &\rightarrow div_{P\Gamma}\mathbf{V} \end{aligned}$$

such that
$div_{P\Gamma}\mathbf{V} = (\nabla \cdot \mathbf{V})|_\Gamma - {}^t\mathbf{n}(\nabla \cdot \mathbf{V})|_\Gamma \mathbf{n}$

From the projected divergence operator on $U_{\epsilon,\Gamma}$, one can define the tangential divergence operator on Γ as follows.

Definition 1.6 (Tangential divergence)**.**
Let Γ be an embedded C^2 hypersurface, The tangential divergence operator is the mapping from $C^1(\Gamma)^3$ into $C(\Gamma)$ defined by

$$div_\Gamma \boldsymbol{V} = div_{P\Gamma}\overline{\mathbf{V}}. \tag{1.6}$$

where $\overline{\mathbf{V}}$ represents $\mathbf{V}$ in $C^1(U_{\epsilon,\Gamma})^3$, that is, $\overline{\mathbf{V}} \in C^1(U_{\epsilon,\Gamma})^3$ and $\overline{\mathbf{V}}|_\Gamma = \mathbf{V}$

The assumption that Γ be C^2 in Definition 1.6 comes from the fact that Fermi coordinates exist for C^2 manifolds. The existence of a function $\overline{\mathbf{V}}$ in Definition 1.6 is a consequence of the existence of Fermi coordinates defined in the section 1.2.1 (Theorem 1.1).

The value of the tangential divergence operator on a surface Γ does not depend on the choice of $\overline{\mathbf{V}}$. It depends only on the values taken by $\mathbf{V}$ on Γ.

In the next section we define the Laplace-Beltrami operator using the tangential gradient operator and the tangential divergence operator.

The Laplace-Beltrami operator

Since we have defined the tangential gradient operator and divergence operator in Subsections 1.2.2 and 1.2.3, the Laplace-Beltrami-operator on Γ, can be defined as the composition of the tangential divergence operator and the tangential gradient operator on $C^2(\Gamma)$.

Definition 1.7 (Laplace-Beltrami operator)**.**
Let Γ be an embedded C^2 hypersurface. The Laplace-Beltrami operator applied to $u \in C^2(\Gamma)$ is

$$\begin{aligned} \Delta_\Gamma : C^2(\Gamma) &\rightarrow C(\Gamma) \\ u &\rightarrow div_\Gamma \nabla_\Gamma u. \end{aligned}$$

1.2.4 Functional spaces

In what follows, all spaces are considered with a real scalar field. A Hilbert space is a complete inner product space, that is a space equipped with an inner product and having the property that every Cauchy sequence is convergent.

We give below some useful Hilbert spaces in this work. For complete definitions and properties see [2, 3, 16]

The functions space $L^p(\Gamma)$ are defined by

$$L^p(\Gamma) = \{u : \Gamma \to \mathbb{R} \text{ measurable } : ||u||_{L^p(\Gamma)} < \infty\}, 1 \leq p \leq +\infty,$$

where

$$||u||_{L^p(\Gamma)} = \left(\int_\Gamma |u|^p ds\right)^{\frac{1}{p}}, \quad 1 \leq p < +\infty$$

and

$$||u||_{L^\infty(\Gamma)} = ess \sup_{x \in \Gamma} |u(x)|.$$

For $p = 2$, the function space $L^2(\Gamma)$ is a Hilbert space for the norm

$$||u||_{L^2(\Gamma)} = \left(\int_\Gamma |u|^2 ds\right)^{\frac{1}{2}}, \quad u \in L^2(\Gamma),$$

which is associated to the inner product

$$(u, v)_{L^2(\Gamma)} = \int_\Gamma uv ds, \quad u, v \in L^2(\Gamma).$$

Some useful subspaces of $L^2(\Gamma)$ are $L^2_\#(\Gamma)$, $H^1(\Gamma)$ and $H^1_\#(\Gamma)$ where

$$L^2_\#(\Gamma) = \left\{u \in L^2(\Gamma), \int_\Gamma u = 0\right\}, H^1(\Gamma) = \{u \in L^2(\Gamma) : \nabla_\Gamma u \in (L^2(\Gamma))^3\}, \text{ and}$$

$$H^1_\#(\Gamma) = \left\{u \in H^1(\Gamma), \int_\Gamma u = 0\right\}.$$

Obviously $H^1(\Gamma)$ is a Hilbert space for the norm

$$||u||^2_{H^1(\Gamma)} = ||u||^2_{L^2(\Gamma)} + ||\nabla_\Gamma u||^2_{L^2(\Gamma)} \quad (u \in H^1(\Gamma)).$$

1.2.5 Requirement for the proof of Existence theorem

As for the classical gradient operator, there is a Poincaré's inequality involving the tangential gradient.

Theorem 1.2 (Poincaré's inequality)**.**
Assume that Γ is an embedded C^2 hypersurface. There exists a constant c such that, for every function $f \in H^1(\Gamma)$ with $\int_\Gamma f = 0$, we have the inequality

$$||f||_{L^2(\Gamma)} \leq c||\nabla_\Gamma f||_{L^2(\Gamma)}.$$

Proof: See Theorem 2.12 in [16]

□

The Green's formulas below (1.7) and (1.8) have been established for functions of class $C^1(\Gamma)$ and $C^2(\Gamma)$ as follows: Let Γ be an embedded closed manifold of class C^1 in $\mathbb{R}^3$, $\mathbf{n}$ an exterior unit normal and H the mean curvature. If u is at least C^1 on the neighbordhood of Γ, then

$$\int_\Gamma \nabla_\Gamma u ds = \int_\Gamma uH\mathbf{n}ds \tag{1.7}$$

Furthermore, if Γ is of class C^2, we have for all $v \in C^1(\Gamma)$, and all $u \in C^2(\Gamma)$:

$$-\int_\Gamma v\Delta_\Gamma u ds = \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma v ds \tag{1.8}$$

Since C^1 and C^2 are dense in $H^1(\Gamma)$ and $H^2(\Gamma)$, these Green's formulas are generalized to functions of $H^1(\Gamma)$ and $H^2(\Gamma)$ thanks to the density argument as stated in the following theorems:

Theorem 1.3 (Green's Theorem for divergence)**.**
Let Γ be a closed embedded hypersurface of class C^1. Then for all $u \in H^1(\Gamma)$, we have

$$\int_\Gamma \nabla_\Gamma u ds = \int_\Gamma uH\boldsymbol{n}ds,$$

where $\boldsymbol{n}$ is the outer unit normal and H is the mean curvature.

Theorem 1.4 (Green's Theorem for Laplacian)**.**
Let Γ be a closed embedded C^2 hypersurface. Then for all $v \in H^1(\Gamma)$, and all $u \in H^2(\Gamma)$ we have:

$$-\int_\Gamma v\Delta_\Gamma u ds = \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma v ds. \tag{1.9}$$

Proof: Let $u \in H^1(\Gamma)$. Since $C^1(\Gamma)$ is dense in $H^1(\Gamma)$ there is a sequence $(u_n)_{n\geq 1} \subset C^1(\Gamma)$ so that

$$\|u_n - u\|_{H^1(\Gamma)} \to 0, \text{ as } n \to \infty$$

we have

$$-\int_\Gamma u_n\Delta_\Gamma \psi ds = \int_\Gamma \nabla_\Gamma u_n \cdot \nabla_\Gamma \psi ds \quad \forall \psi \in C^2(\Gamma)$$

and $\phi \to \int_\Gamma \phi\Delta_\Gamma \psi ds$ is a continuous linear form on $L^2(\Gamma)$ thus

$$-\int_\Gamma u_n\Delta_\Gamma \psi ds \to \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma \psi ds \quad \text{ as } n \to \infty$$

therefore, $\phi \to \int_\Gamma \nabla_\Gamma \phi \nabla_\Gamma \psi ds$ is a continuous linear form on $H^1(\Gamma)$, it follows that

$$\int_\Gamma \nabla_\Gamma u_n \cdot \nabla_\Gamma \psi ds \to \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma \psi ds \quad \text{as } n \to \infty$$

and from classical Green formula (1.8), we are led to

$$\int_\Gamma u \Delta_\Gamma \psi ds = -\int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma \psi ds \quad \forall u \in H^1(\Gamma) \quad \forall \psi \in C^2(\Gamma)$$

In the similar manner, for $v \in H^2(\Gamma)$ there is $(v_n)_{n\geq 1} \subset C^2(\Gamma)$ so that

$$\|v_n - v\|_{H^2(\Gamma)} \to 0, \text{ as } n \to \infty$$

and

$$-\int_\Gamma \phi \Delta_\Gamma v_n ds = \int_\Gamma \nabla_\Gamma \psi \cdot \nabla_\Gamma v_n ds \quad \forall \psi \in C^1(\Gamma).$$

Since $\phi \to \int_\Gamma \phi \Delta_\Gamma v ds$ is a continuous linear form on $L^2(\Gamma)$ thus

$$-\int_\Gamma \psi \Delta_\Gamma v_n ds \to \int_\Gamma \nabla_\Gamma \psi \cdot \nabla_\Gamma v ds \quad \text{as } n \to \infty$$

therefore, $\phi \to \int_\Gamma \nabla_\Gamma \psi \nabla_\Gamma \phi ds$ is a continuous linear form on $H^2(\Gamma)$, which implies

$$\int_\Gamma \nabla_\Gamma \psi \cdot \nabla_\Gamma v_n ds \to \int_\Gamma \nabla_\Gamma v \cdot \nabla_\Gamma \psi ds \quad \text{as } n \to \infty$$

and from classical Green formula (1.8), we are led to

$$\int_\Gamma v \Delta_\Gamma \psi ds = -\int_\Gamma \nabla_\Gamma v \cdot \nabla_\Gamma \psi ds \quad \forall v \in H^2(\Gamma) \quad \forall \psi \in C^1(\Gamma)$$

□

A consequence of the Green's Theorem 1.4 is that the operator $-\Delta_\Gamma$ is symmetric and positive.

Theorem 1.5 ($-\Delta_\Gamma$ is symmetric and positive)**.**
Let Γ be a closed embedded C^2 hypersurface. The Laplace-Beltrami operator is symmetric and positive, more precisely we have: for all $u, v \in H^2(\Gamma)$

$$\int_\Gamma v \Delta_\Gamma u ds = \int_\Gamma u \Delta_\Gamma v ds, \tag{1.10}$$

$$-\int_\Gamma u \Delta_\Gamma u ds = \int_\Gamma ||\nabla_\Gamma u||^2 \geq 0. \tag{1.11}$$

1.3 The Poisson problem on a closed smooth surface

Now that we have defined the Laplace-Beltrami operator in Subsection 1.2.3, we can study the Poisson problem on closed hypersurfaces. We start by setting the problem and the weak formulation in Subsection 1.3.1. We end up by summarising the existence result in Subsection 1.3.2.

1.3.1 Position of the problem and its weak form

Let Γ be a closed C^2 hypersurface in $\mathbb{R}^3$. Since Γ is closed ($\partial\Gamma = \emptyset$), all the constant functions are in the kernel of Δ_Γ which is therefore non invertible on the space of functions $u \in H^1(\Gamma)$. Hence we have to impose the global condition $\int_\Gamma u = 0$ to guarantee the uniqueness of the solution.

Letting $f \in L^2(\Gamma)$, we are looking for a solution $u : \Gamma \to \mathbb{R}$ such that

$$-\Delta_\Gamma u = f \text{ on } \Gamma \tag{1.12}$$

Since Γ is closed, we need an additional condition to guarentee the well-posedness of (1.12). The reason is that $u = const$ makes the left hand side of (1.12) to vanish. Taking into account the fact that the null space of the Laplace-Beltrami operator on closed surface is the space of constant functions, we need to add an additional condition to allow the zero function: $\int_\Gamma uds = 0$. The equation (1.12) is multiplied by a test function $v \in H^1(\Gamma)$ followed by integration by parts over Γ,

$$\int_\Gamma \Delta_\Gamma uvds = \int_\Gamma fvds$$

and after applying Green's formula (Theorem 1.4), we have the following weak formulation

$$\text{Find } u \in H^1_\#(\Gamma) \text{ such that } \forall v \in H^1_\#(\Gamma), \quad \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma vds = \int_\Gamma fvds. \tag{1.13}$$

1.3.2 Existence result on a smooth manifold

We describe an abstract theory to obtain the existence and the uniqueness of the solution of a variational formulation in a Hilbert space. We denote by V a real Hilbert space with its norm $||.||$. We consider a variational formulation of the type:

$$\text{find } u \in V \text{such that } a(u, v) = b(v) \text{ for every } v \in V. \tag{1.14}$$

with the following hypotheses on $a(\cdot, \cdot)$ and $b(\cdot)$:

- $b(\cdot)$ is a continuous linear form on V. i.e.,

$$\exists C_b > 0, \qquad \forall v \in V, |b(v)| \leq C_b||v||.$$

- $a(\cdot, \cdot)$ is a continuous bilinear form on $V \times V$, i.e.,

$$\exists C_a > 0, \quad \forall u, v \in V, \quad |a(u, v)| \leq C_a||u||||v||.$$

- $a(\cdot, \cdot)$ is coercive (or elliptic) on V i.e.,

$$\exists c_a > 0, \quad \forall u \in V, \quad |a(u, u)| \geq c_a||u||^2.$$

Existence, uniqueness and the stability of solution is the consequences of the following Lax-Milgram result

Theorem 1.6 (Lax Milgram Lemma)**.** *Let V be a real Hilbert space, $b(\cdot)$ a continuous linear form on V, $a(\cdot, \cdot)$ a continuous bilinear form on $V \times V$ which is V-coercive. Then the variational formulation* (1.14) *has a unique solution. Further, this solution depends continuously on the linear form $b(\cdot)$.*

1.3.3 Application to the Poisson problem

The variational formulation (1.13) of the Laplacian with the global condition is written in the general form (1.14) with:

- $V = H^1_{\#}(\Gamma)$
- $a(u,v) = \int_\Gamma \nabla_\Gamma u.\nabla_\Gamma v ds$
- $b(v) = \int_\Gamma f v ds$

$H^1_{\#}(\Gamma)$ is the Hilbert space endowed with the norm $||u||_{H^1_{\#}(\Gamma)} \equiv ||u||_{H^1(\Gamma)}$. It is obvious that $a(\cdot,\cdot)$ is a bilinear form on $H^1_{\#}(\Gamma) \times H^1_{\#}(\Gamma)$ and $b(\cdot)$ is a linear form on $H^1_{\#}(\Gamma)$; and it is easy to show that a and b are continuous (Cauchy-Schwartz inequality). Indeed

$$\begin{aligned}
\forall u, v \in H^1_{\#}(\Gamma), \quad |a(u,v)| &= \left| \int_\Gamma \nabla_\Gamma u.\nabla_\Gamma v ds \right| \\
&\leq \int_\Gamma |\nabla_\Gamma u||\nabla_\Gamma v| ds \\
&\leq \left(\int_\Gamma ||\nabla_\Gamma u||^2 ds \right)^{\frac{1}{2}} . \left(\int_\Gamma ||\nabla_\Gamma v||^2 ds \right)^{\frac{1}{2}} \\
&\leq C||u||_{H^1(\Gamma)}.||v||_{H^1(\Gamma)}.
\end{aligned}$$

and for all $v \in H^1_{\#}(\Gamma)$,

$$\begin{aligned}
|b(v)| &= \left| \int_\Gamma f v ds \right| \\
&\leq ||f||_{L^2(\Gamma)}.||v||_{L^2(\Gamma)} \text{ by Hölder} \\
&\leq C||f||_{L^2(\Gamma)}.||\nabla v||_{L^2(\Gamma)} \text{ by Poincaré's inequality (Theorem 1.2)} \\
&\leq C||f||_{L^2(\Gamma)}.||v||_{H^1(\Gamma)} \\
&\leq C_1||v||_{H^1(\Gamma)}
\end{aligned}$$

The coercivity of $a(\cdot,\cdot)$ is a consequence of Poincaré inequality (Theorem 1.2):
we have

$$\begin{aligned}
a(u,u) &= \int_\Gamma |\nabla_\Gamma u|^2 ds \\
&= \frac{1}{2}||\nabla_\Gamma u||^2_{L^2(\Gamma)} + \frac{1}{2}||\nabla_\Gamma u||^2_{L^2(\Gamma)} \\
&\geq \frac{1}{2}||\nabla_\Gamma u||^2_{L^2(\Gamma)} + \frac{1}{2c^2}||u||^2_{L^2(\Gamma)} \\
&\geq \min(\frac{1}{2}, \frac{1}{2c^2})||u||^2_{H^1(\Gamma)} \\
&\geq C'||u||^2_{H^1(\Gamma)}.
\end{aligned}$$

By application of the Lax-Milgram theorem (Theorem 1.6) we obtain the existence of a unique weak solution for the variational formulation (1.13) of problem (1.12).

The regularity of the solution requires the regularity of both the right hand side and the manifold. The following theorem is taken from [16, Theorem 3.3].

Theorem 1.7 (Regularity of solution)**.**
Let Γ, be a compact C^2 hypersurface in $\mathbb{R}^3$ and assume that $f \in L^2(\Gamma)$ with $\int_\Gamma f = 0$. Then, the unique weak solution of (1.13) *satisfies $u \in H^2_\#(\Gamma)$, and there exists a constant $C > 0$ such that*

$$||u||_{H^2(\Gamma)} \leq C||f||_{L^2(\Gamma)}.$$

Proof: See Theorem 3.3 in [16]

□

For more details see [3] for Euclidean case and [15, 16] for the case of curved surfaces.

The method is extended to general linear elliptic Partial Differential Equations in divergence form and to boundary value problem on surface we the boundary.

1.3.4 General elliptic partial differential equations on manifolds

Following [16, 21], we define a more general linear elliptic PDEs on manifolds Γ expressed in divergence form as follows: Given a source function $F \in L^2(\Gamma)$, find $u : \Gamma \to \mathbb{R}$ such that

$$-\sum_{i,j=1}^{n+1} \underline{D}_i(a_{ij}\underline{D}_j u) + \sum_{i=1}^{n+1} a_i \underline{D}_i(u) + c(x)u = F \text{ on } \Gamma, \tag{1.15}$$

Where

- a_{ij}, a_i and $c(x)$ are bounded
- $a(x) = (a_i(x))_{i=1}^{n+1} \in T_x\Gamma$
- $a_{ij} = a_{ji}$
- The elliptic condition is given by the following so-called Ladyzhenskaya condition: there exists a number $c_0 > 0$ such that

$$\sum_{i,j=1}^{n+1} a_{ij}\xi_i\xi_j + \sum_{i=1}^{n+1} a_i\xi_i\xi_0 + c(x)\xi_0^2 \geq c_0 \sum_{i=1}^{n+1} \xi_i^2 a.e. x \in \Gamma, \forall \xi_0 \in \mathbb{R},$$
$$\forall \xi = (\xi_i)_{i=1}^{n+1} \in \mathbb{R}^{n+1}, \quad \xi \cdot x = 0$$

Ellipticity means that for each point $x \in \Gamma$, the symmetric matrix $(a_{ij}(x))$ is positive definite, with smallest eigenvalue greater than or equal to c_0 on the space $\xi \cdot x = 0$.

An obvious example is $a_{ij} \equiv \delta_{ij},\ a_i \equiv 0,\ c \equiv 0$, in which case the operator is $-\Delta_\Gamma$.

The bilinear form associated with the divergence form (1.15) is

$$a(u, v) = \int_\Gamma \left(\sum_{i,j=1}^{n+1} a_{ij}\underline{D}_i u \underline{D}_j v + \sum_{i=1}^{n+1} a_i v \underline{D}_i u + cuv \right) ds$$

for $u,\ v \in H^1(\Gamma)$ and the linear form is

$$L(v) = (F, v) = \int_\Gamma F v ds.$$

Theorem 1.8. *Let Γ be a C^2 compact hypersurface. Assume that the coefficients satisfy the above conditions. Then there exists a unique weak solution of* (1.15)*, that is, there exists a unique $u \in H^1(\Gamma)$ such that*

$$a(u, \phi) = b(\phi)$$

for every $\phi \in H^1(\Gamma)$

Proof: Direct application of the Lax–Milgram theorem.

□

1.4 Eigenvalues of the Laplacian

The study of the eigenvalues of the Laplacian has numerous interests. Among other, the spectrum of the Laplace-Beltrami has been the subject of analysis and the base of a method to compare and analyse different shapes. The allows to study particular the sound coming from music instruments. The operator $-\Delta_\Gamma$ is compact and symmetric and we can show as in finite dimension that it can be diagonalized in a Hilbert base (See, e.g, [50, Theorem 44]).

A function $u \in C^2(\Gamma)$ is an eigenfunction of $-\Delta_\Gamma$ associated to the eigenvalue λ if

$$-\Delta_\Gamma u = \lambda u \text{ on } \Gamma \tag{1.16}$$

Theorem 1.9 (Sturm-Liouville's decomposition)**.**
Given a compact Riemannian manifold Γ there is an orthonormal basis $u_1, \cdots, u_j, \cdots$ of eigenfunctions of the Laplacian Δ_Γ with respective eigenvalues $0 \leq \lambda_1 < \cdots < \lambda_j < \cdots$ such that any function $u \in L^2(\Gamma)$ can be written as a convergent series in $L^2(\Gamma)$.

$$u = \sum_{j=1}^{\infty} \alpha_j u_j$$

where coefficients $\alpha_j \in \mathbb{R}$

Proof : See Section 1.5 in [50].

□

Previously, the Poisson equation $-\Delta_\Gamma u = f$ was written in the variational form,

$$\forall v \in H^1_\#(\Gamma) \quad a(u, v) = b(v).$$

We obtain the existence, uniqueness, stability and regularity of solution. We will proceed in this manner to study the spectrum and to suggest a variational formulation for the spectral problem as follows: find $u \in H^1_\#(\Gamma)$ and $\lambda \in \mathbb{R}$ such that

$$\text{for all } v \in H^1_\#(\Gamma), \qquad a(u, v) = \lambda < u, v >_{L^2(\Gamma)} . \tag{1.17}$$

Let us show that

Lemma 1.2.
Two eigenfunctions of the operator $-\Delta_\Gamma$ associated to different eigenvalues are orthogonal in $H^1_\#(\Gamma)$.

Proof:
Let u_1 and u_2 be two eigenfunctions associated respectively to the eigenvalues λ_1 and λ_2 such that $\lambda_1 \neq \lambda_2$. Given that u_1 and u_2 are eigenfunctions of the operator $-\Delta_\Gamma$ and that $a(\cdot,\cdot)$ is a bilinear symmetric form, for all $v \in H^1_\#(\Gamma)$, we have:

$$a(u_1, v) = \lambda_1 < u_1, v >_{L^2(\Gamma)}, \tag{1.18}$$
$$a(u_2, v) = \lambda_2 < u_2, v >_{L^2(\Gamma)} . \tag{1.19}$$

Taking v to be u_2 in (1.18) and v to be u_1 in (1.19) and using also the fact that $a(\cdot,\cdot)$ is symmetric, we have:

$$\begin{array}{l} -\left\{ \begin{array}{l} a(u_1, u_2) = \lambda_1 < u_1, u_2 >_{L^2(\Gamma)} \\ a(u_1, u_2) = \lambda_2 < u_1, u_2 >_{L^2(\Gamma)}, \end{array} \right. \\ \hline 0 = \lambda_1 < u_1, u_2 >_{L^2(\Gamma)} - \lambda_2 < u_1, u_2 >_{L^2(\Gamma)} \end{array}$$

which gives

$$0 = (\lambda_1 - \lambda_2) < u_1, u_2 >_{L^2(\Gamma)}$$

then

$$< u_1, u_2 >_{L^2(\Gamma)} = 0 \text{ since } \lambda_1 \neq \lambda_2.$$

Thus u_1 and u_2 are orthogonal.

□

The spectral theory of compact self-adjoint operators or the direct approach by the study of the Rayleigh quotient helps to prove the following theorems.

Theorem 1.10 (Existence of an Hilbertian basis of eigenfunctions)**.**

Let Γ be a compact Riemannian manifold. There exist an increasing sequence $(\lambda_k)_{k\geq 1}$ of positive real numbers which tends to infinity and an Hilbert basis $(u_k)_{k\geq 1}$ of $L^2_\#(\Gamma)$ functions such that $\forall k \geq 1,\ u_k \in H^1_\#(\Gamma)$ and

$$\forall v \in H^1_\#(\Gamma), \quad a(u_k, v) = \lambda_k < u_k, v >_{L^2_\#(\Gamma)} .$$

Proof: See, e.g., [3, 11, 34, 50].

□

⋆ ⋆ CHAPTER TWO ⋆ ⋆

POISSON PROBLEM ON LIPSCHITZ MANIFOLDS

Contents

2.1 Introduction

In standard calculus, the gradient, divergence and Laplace operators are classically defined on the Euclidean space $\mathbb{R}^d$. They can however also be defined on a C^1 manifold M in an intrinsic way (no immersion into $\mathbb{R}^d$) using a Riemannian metric (See for example [28, Section 1.2]) or the Hodge operator ([4, Section 11.2]). Dziuk in [16] defines differential operators (tangential gradient and divergence) on embedded C^2 surfaces using Fermi coordinates. Fermi coordinates ([16, Section 2.3]) are global coordinates on a hypersurface $\Gamma \in \mathbb{R}^d$ that avoid working with charts. Unfortunately, the existence of Fermi coordinates requires Γ to be C^2, or more precisely to satisfy both an interior

and an exterior sphere condition. The issue of polyhedral surfaces is raised in [15, Page 3] , which quotes [49] for the definition of Sobolev spaces on $C^{0,1}$ manifolds. However, [49, Definition 2.10 and Section 4.2] deal with $C^{2,\kappa}$ manifolds. In order to perform calculus on polyhedra, we chose instead to define the differential operators on Lipschitz manifolds following [24, Section 3] , [22, appendix A] and [32]. Indeed we will later approximate the C^1 surface Γ by a polyhedral manifold Γ_h (Chapter 3). Both polyhedral manifolds and C^1 manifolds are particular cases of Lipschitz manifolds, hence it is possible to use the same framework for both the continuous and discrete settings. Lipschitz manifolds are an intermediate structure between topological manifolds where the tangent space is defined nowhere and C^1 manifolds where the tangent space is defined everywhere. On Lipschitz manifolds the tangent space is defined almost everywhere, which is enough to define spaces of functions that are weakly differentiable. [22] defines Lebesgue L^p spaces, Sobolev $W^{1,p}$ and $W^{-1,p'}$, $\frac{1}{p}+\frac{1}{p'}=1$ spaces. Following [22], we recall the definition of Lipschitz manifolds in Section 2.2. Sobolev spaces on Lipschitz manifolds are introduced in Subsection 2.2.4, and the Laplace-Beltrami operator on Lipschitz manifolds in Subsection 2.2.5. The Stokes'Theorem, Green's Theorem as well as Poincaré inequality type are proven in order to obtain the existence, uniqueness and stability of solution on Lipschitz manifolds in Section 2.3.

2.2 Calculus on Lipschitz manifolds

We recall that a **topological manifold** $\mathcal{M}$ of dimension $d \in \mathbb{N}^*$ is a topological space such that for every $x \in \mathcal{M}$ there exists an open set $U \subset \mathcal{M}$ and a map $\phi : U \to \mathbb{R}^d$ such that $x \in U$ and ϕ is an homeomorphism onto its image $\phi(U)$. The couple (U, ϕ) where $U \in \mathcal{M}$ is an open set around x and $\phi : U \to \phi(U) \subset \mathbb{R}^d$ an homeomorphism is called a **local coordinate chart**. An atlas on $\mathcal{M}$ is a countable familly $\{U_i, \phi_i\}_{i\in I}$ such that $\mathcal{M} = \cup_{i\in I} U_i$ and (U_i, ϕ_i) is a local coordinate chart for each $i \in I$.

We recall that a map $\phi : E \to F$ between two metric spaces E and F is a **biLipschitz map** if it is an homeomorphism onto its image $\phi(E)$, and its inverse is also Lipschitz.

Definition 2.1 (Lipschitz manifold)**.**
A **Lipschitz manifold** *is a topological manifold with an atlas (called a* **Lipschitz atlas***)* $\{U_i, \phi_i\}_{i\in I}$ *such that for any* $i, j \in I$ *the transition map* $\phi_i \circ \phi_j^{-1}$ *is biLipschitz from* $\phi_j(U_i \cap U_j)$ *to* $\phi_i(U_i \cap U_j)$*.*

The functional analysis on Lipschitz manifolds requires the definition of negligible sets.

Definition 2.2 (Negligible sets)**.**
Let Γ *be a Lipschitz manifold and* $\{U_i, \phi_i\}_{i\in I}$ *its Lipschitz atlas.*
A set $S \subset \Gamma$ *is negligible provided* $\phi_i(U_i \cap S)$ *has measure zero in* $\mathbb{R}^d$*.*

The following Rademacher's Theorem is taken from [45, Corollary 11.7]

Theorem 2.1 (Rademacher)**.**
Any locally Lipschitz continuous function defined on an open set of $\mathbb{R}^d$ *is differentiable (in the classic sense) almost everywhere.*

Proof: See Corollary 11.7 in [45].

□

A property that is true on all but a negligible set of points of Γ is said to hold almost everywhere.

Given a Lipschitz manifold Γ with its structural atlas $\{U_i, \phi_i\}_{i\in I}$, a point $x \in \Gamma$ is said to be a **singular point** if there exists $i, j \in I$ such that $x \in U_i \cap U_j$ and $\phi_i \circ \phi_j^{-1}$ is not differentiable at $\phi_j(x)$. A point x that is not singular is called a **regular point**.

A consequence of the Rademacher's Theorem 2.1 is that **the set of singular points is negligible**. Hence for any Lipschitz manifold one can define properly a tangent space only almost everywhere (See Definition 2.5 below). Moreover, if the Lipschitz manifold is embedded in $\mathbb{R}^d$, it admits almost everywhere a normal vector.

Unlike smooth manifolds where differentiability can take place everywhere and be of any order, differentiability on Lipschitz manifolds can happen only at regular points and is of order one.

Definition 2.3 (Differentiable map)**.**
Let Γ be a Lipschitz manifold. A map $f : \Gamma \to \mathbb{R}$ is said to be differentiable at $x \in \Gamma$ provided

- *x is a regular point of Γ,*
- *there exists a local chart (U, ϕ) such that $x \in U$ and $f \circ \phi^{-1} : \phi(U) \to \mathbb{R}$ is differentiable at $\phi(x)$.*

Definition 2.4 (Measurable and Lipschitz maps)**.**
Let Γ be a Lipschitz manifold, $m \in \mathbb{N}^*$. A map $f : \Gamma \to \mathbb{R}^m$ is measurable (resp. Lipschitz) if for any local coordinate chart (U, ϕ), the map $f \circ \phi^{-1} : \phi(U) \to \mathbb{R}^m$ is measurable (resp. Lipschitz).

Lipschitz maps are differentiable almost everywhere thanks to the Rademacher's Theorem 2.1.

2.2.1 Tangent space

Let Γ a Lipschitz manifold and $\{U_i, \phi_i\}_{i\in I}$ its Lipschitz atlas. Given a point $x \in \Gamma$, two real valued maps f and g defined on a neighborhood of x are said to be **equivalent maps** if there exists an open set $U \ni x$ such that $f|_U = g|_U$. Classes of equivalence modulo this relation are called **germs** at x (See [31, Definition 3.9]). We denote $Diff_x(\Gamma)$ the vector space of **germs of differentiable functions** at a regular point x.

Let $x \in \Gamma$ be a regular point, $\epsilon > 0$. A **path through** x is a continuous map $\gamma_x :]-\epsilon, \epsilon[\to \Gamma$ with $\gamma_x(0) = x$ and such that there exists $i \in I$, $x \in U_i$ and $\phi_i \circ \gamma_x$ is differentiable at 0.
We define the following equivalence relation between paths through x : γ_{1x} and γ_{2x} are equivalent if they share the same derivative at x :

$$\gamma_{1x} \equiv_x \gamma_{2x} \Leftrightarrow (\phi_i \circ \gamma_{1x})'(0) = (\phi_i \circ \gamma_{2x})'(0). \tag{2.1}$$

Following [22, Appendix A] and [24, Section 3.1], we define the tangent space almost everywhere as the space of equivalence classes $\dot{\gamma_x}$ for the relation (2.1).

Definition 2.5 (Tangent space)**.**
Let Γ a Lipschitz manifold. The tangent space of Γ at $x \in \Gamma$ is the vector space

$$\begin{aligned} T_x\Gamma &= \{\dot{\gamma_x}, \gamma_x \text{ a path through } x\} && \text{if } x \text{ is a regular point} \\ T_x\Gamma &= \{0\} && \text{if } x \text{ is a singular point} \end{aligned} \tag{2.2}$$

If Γ is a Lipschitz manifold of dimension $d \in \mathbb{N}^*$, its tangent space at each regular point is a vector space of dimension d. At every regular point $x \in \Gamma$ belonging to a local chart $(U, \phi = (\phi^1, \ldots, \phi^d))$, we can define a basis of the tangent space $T_x\Gamma$ by considering the d curves $\phi_x^j(t) = \phi_i^{-1}(t_1^x, \ldots, t_{j-1}^x, t_j^x + t, t_{j+1}^x, \ldots, t_d^x)$ where $(t_1^x, \ldots, t_{j-1}^x, t_j^x + t, t_{j+1}^x, \ldots, t_d^x) = \phi_i^{-1}(x)$. The curves $t \to \phi^j(t)$ form d independent classes for the relation (2.1). Hence $\dot{\phi_x^j}$ form a basis of $T_x\Gamma$.

Definition 2.6 (Strong gradient map)**.**
Let Γ be a Lipschitz manifold, $x \in \Gamma$ a regular point and $f : \Gamma \to \mathbb{R}$ be differentiable at $x \in \Gamma$. The linear tangent map of f at x denoted $\nabla_x f$ is the map sending the equivalence class $\dot{\gamma}$ of the relation $\equiv_x$ to the equivalence class $f \dot{\circ} \gamma$ of the relation $\equiv_{f(x)}$:

$$\begin{aligned} \nabla_x f : T_x\Gamma &\to \mathbb{R} && (2.3) \\ \frac{\mathrm{d}}{\mathrm{d}\gamma_x} &\to \frac{\mathrm{d}}{\mathrm{d}t}(f \circ \gamma)(t)\Big|_{t=0}. && (2.4) \end{aligned}$$

On a Lipschitz manifold of dimension d, if $x \in \Gamma$ is a regular point, $T_x\Gamma$ is a real vector space of dimension d and basis $\dot{\phi}^1_x, \dots \dot{\phi}^d_x$. Following a classical duality argument, the strong gradient of f at x, which is a linear form, can therefore be represented by a d-dimensional real vector denoted $\nabla_x f$ such that

$$\nabla_x f \left(\sum_{i=1}^{d} a_i \dot{\phi}^i_x \right) = (a_1, \dots, a_d) \cdot \nabla_x f.$$

2.2.2 Measurable and Lipschitz differential forms

On any vector space E of dimension d, we introduce for any integer $l \in \{0, 1, \dots, d\}$ the l-th exterior power $\Lambda^l E$ as the vector space of l-linear alternate forms on E.
Because of the lack of differentiability of Lipschitz manifolds, we cannot work with smooth differential forms as done classically. Instead, following [24, Section 3.2] , we introduce measurable and Lipschitz differential forms as follows.

Definition 2.7 (Measurable and Lipschitz differential forms)**.**
Let $p \in [1, \infty]$, $l \in \{0, 1, \dots, d\}$, Γ be a closed Lipschitz manifold of dimension d, and $\{U_i, \phi_i\}_{i \in I}$ its Lipschitz atlas.
A measurable (resp. Lipschitz) differential form ω of degree l is a mapping defined almost everywhere on Γ such that

- *$\omega \in \Lambda^l T_x\Gamma$ for almost every $x \in \Gamma$*
- *for any local chart $(U, \phi = (\phi^1, \dots, \phi^d))$ and any multi-index J of lenght l, there exist a_J is measurable (resp. Lipschitz) such that almost everywhere*

$$\omega_U = \sum_{|J|=l} a_J \mathrm{d}\phi^J. \tag{2.5}$$

The **support of a form** α is the closure of the set of points x where $\alpha_x \neq 0$.
Due to the lack of regularity, we cannot define the exterior derivative as an endomorphism on differential forms as done classically (See [31, Theorem 5.42]). Instead, the derivative of a Lipschitz differentiable form is a measurable differentiable form defined as follows.

Definition 2.8 (Derivative of a Lipschitz form)**.**
Let Γ be a Lipschitz manifold of dimension d, and $\{U_i, \phi_i\}_{i \in I}$ its Lipschitz atlas. Let $l \in \{0, 1, \dots, d\}$, and ω a Lipschitz differential form of degree l on Γ. The exterior derivative of ω is the measurable differential form $\mathrm{d}\omega$ of degree $l + 1$ such that on any local chart $(U, \phi = (\phi^1, \dots, \phi^d))$

$$\mathrm{d}\omega_U = \sum_{|J|=l} \sum_{j \in J} \frac{\partial a_J}{\partial \phi^j} \mathrm{d}\phi^j \wedge \mathrm{d}\phi^J, \tag{2.6}$$

where a_J are the Lipschitz coefficients of ω : $\omega_U = \sum_{|J|=l} a_J \mathrm{d}\phi^J$.

In Definition 2.8, the differentiability of a Lipschitz form follows from the Rademacher's Theorem 2.1.

The orientability of a manifold is a necessary condition for the existence of a volume form (See [31, Theorem 6.5]), which in turns yields a measure and integration theory on the manifold. The classical definition of orientability involves the positiveness *e*verywhere of the Jacobian of the transition maps $\phi_i \circ \phi_j^{-1}$. However, in the case of Lipschitz manifolds, the transition maps are differentiable only *a*lmost everywhere.

Definition 2.9 (Orientation)**.**
A Lipschitz manifold Γ with Lipschitz atlas $\{U_i, \phi_i\}_{i\in I}$ is oriented provided the change of coordinates $\phi_i \circ \phi_j^{-1}$ has positive Jacobian on $U_i \cap U_j$ almost everywhere.

An orientation is therefore given by an atlas such that the coordinate charts $(U_i, \phi_i = (\phi_{i1}, \dots, \phi_{id}))$ define a positively oriented basis $\left(\frac{\mathrm{d}}{\mathrm{d}\phi_{i1}}, \dots, \frac{\mathrm{d}}{\mathrm{d}\phi_{id}}\right)$ of the tangent space almost everywhere.

On any oriented Lipschitz manifold Γ of dimension d, there exists $\mathrm{d}V_\Gamma(x) \in \Lambda^d T_x\Gamma$ for almost every $x \in \Gamma$ such that $\mathrm{d}V_\Gamma(x) \neq 0$ a.e. The proof is a direct adaptation of [31, Theorem 6.5]. Such a form is called a **volume form** (see [31, Definition 6.3]) and takes the form $\mathrm{d}V_\Gamma = f\mathrm{d}\phi^1 \cdots \mathrm{d}\phi^d$, where f is measurable.

Integrating volume forms requires the pullback operator. Following the Definition 5.18 in [31] we define the pullback of a volume form by a Lipschitz map.

Definition 2.10 (Pullback of a volume form)**.**
Let Γ be a Lipschitz manifold of dimension $d \in \mathbb{N}^*$, U an open subset of $\mathbb{R}^d$ and let $g : U \to \Gamma$ be a Lipschitz map.
The pullback of a measurable differentiable form ω on Γ by g is the measurable differentiable form $g^*\omega$ on $\mathbb{R}^d$ defined by

$$(g^*\omega)_x(v_1, \dots, v_q) = \omega_{g(x)}(\nabla_{g(x)}g \cdot v_1, \dots, \nabla_{g(x)}g \cdot v_q) \quad a.e.\ x \in U,\ \forall v_1, \dots, v_q \in \mathbb{R}^d. \tag{2.7}$$

Note that the pullback of a Lipschitz differential form by a Lipschitz map is merely a measurable differential form (not Lipschitz). For instance, the pullback of a volume form $\mathrm{d}V_\Gamma = f\mathrm{d}\phi^1 \cdots \mathrm{d}\phi^d$ by a Lipschitz map g is the form $(g^*\omega)_x = f \circ g(x) \det(\nabla_{g(x)}g)\mathrm{d}x^1 \cdots \mathrm{d}x^d$ (See [31, Proposition 6.12]), which is not necessarily Lipschitz.

On $\mathbb{R}^d$, volume forms take the form $\mathrm{d}V_{\mathbb{R}^d}(x) = f(x)\mathrm{d}x^1 \cdots \mathrm{d}x^d$ and their integral is simply $\int_{\mathbb{R}^d} \mathrm{d}V_{\mathbb{R}^d} = \int_{\mathbb{R}^d} f(x)\mathrm{d}x^1 \dots \mathrm{d}x^d$. On a general Lipschitz manifold, the **integral of a volume form** (See [31, Definition 6.16] and [22, Formula A.10]) is defined by

$$\int_\Gamma f\mathrm{d}\lambda_\Gamma = \sum_{i\in I} \int_{\phi_i(U_i)} (\phi_i^{-1})^*(\theta_i f \mathrm{d}V_\Gamma), \tag{2.8}$$

where $(\theta_i)_{i\in I}$ is a Lipschitz partition of unity on Γ, subordinate to the cover $(U_i)_{i\in I}$: $supp(\theta_i) \subset U_i$, $0 \leq \theta_i \leq 1$ and $\sum_{i\in I} \theta_i = 1$ (See [31, Proposition 6.14]).
The value of the integral (2.8) does not depend on the choice of coordinate charts nor the partition of unity (See [31, Theorem 6.15]).

2.2.3 Riemannian metric on Lipschitz manifolds

Definition 2.11 (Riemannian metric on a Lipschitz manifold)**.**

- *for any chart (U, ϕ), the functions*

$$g_{ij}^U(x) = \langle \frac{\mathrm{d}}{\mathrm{d}\phi_i}, \frac{\mathrm{d}}{\mathrm{d}\phi_j} \rangle_x, \quad x \in U, \quad 1 \leq i, j, \leq n, \tag{2.9}$$

  are measurable on U : $g_{ij}^U \circ \phi^{-1}$ should be measurable on $\phi(U)$.

- *there exist two constant $C_1, C_2 > 0$ such that for any chart (U, ϕ), for almost every point $x \in U$ and any path γ_x through x such that $\phi \circ \gamma_x$ is differentiable at 0*

$$C_1 ||(\phi \circ \gamma_x)'(0)||_{\mathbb{R}^d}^2 \leq \langle \frac{\mathrm{d}}{\mathrm{d}\gamma_x}, \frac{\mathrm{d}}{\mathrm{d}\gamma_x} \rangle_x \leq C_2 ||(\phi \circ \gamma_x)'(0)||_{\mathbb{R}^d}^2. \tag{2.10}$$

On a Lipschitz Riemannian manifold, the matrix $G^U(x) = (g_{ij}^U(x))_{1 \leq i,j, \leq d}$ is bounded, symmetric, and positive definite in an uniform way : for almost every $x \in U$

$$C_1 ||v||_{\mathbb{R}^d}^2 \leq \langle G^U(x)v, v \rangle_x \leq C_2 ||v||_{\mathbb{R}^d}^2, \quad \forall v \in \mathbb{R}^d. \tag{2.11}$$

The existence of a Riemannian metric on any compact Lipschitz manifold is given by [22, Proposition A.3].

On any oriented Lipschitz manifold Γ of dimension d, the volume form $\mathrm{d}V_\Gamma$ can be normalized to one for the norm associated to the Riemannian metric. On every local chart $(U_i, \phi_i = (\phi_i^1, \ldots, \phi_i^d))$, for almost every $x \in U$ the volume form then takes the form

$$\mathrm{d}V_\Gamma = \sqrt{|\det\left(G^U(x)\right)|}\ \mathrm{d}\phi^1 \wedge \cdots \wedge \mathrm{d}\phi^d. \tag{2.12}$$

The volume form $\mathrm{d}V_\Gamma$ (2.12) gives rise to a Borel regular measure $\mathrm{d}\lambda_\Gamma$ called the **Riemannian measure** such that

$$\int_\Gamma f \mathrm{d}\lambda_\Gamma = \sum_{i \in I} \int_{\phi_i(U_i)} \left[\sqrt{|\det\left(G^{U_i}\right)|}\ \theta_i f \right] \circ \phi_i^{-1}(y)\, \mathrm{d}y, \tag{2.13}$$

where $(\theta_i)_{i \in I}$ is a Lipschitz partition of unity on Γ, subordinate to the cover $(U_i)_{i \in I}$: $supp(\theta_i) \subset U_i$, $0 \leq \theta_i \leq 1$ and $\sum_{i \in I} \theta_i = 1$ (See [31, Proposition 6.14]). The value of the integral (2.13) does not depend on the choice of coordinate charts nor the partition of unity (See [31, Theorem 6.15]).

2.2.4 Sobolev spaces on Lipschitz manifolds

The Lebesgue spaces are defined by a pullback of the Euclidean Lebesgue space.

Definition 2.12 (Lebesgue space $L^p(\Gamma)$)**.**
Let $p \in [1, \infty]$, Γ be a compact Lipschitz manifold of dimension d and $\{U_i, \phi_i\}_{i \in I}$ its Lipschitz atlas. The space $L^p(\Gamma)$ is defined as the set of real valued functions f defined almost everywhere on Γ such that for any local chart $\{U_i, \phi_i\}$, $f \circ \phi_i^{-1} \in L^p(\phi_i(U_i))$.

The associated norm is

$$||f||_p = \left(\int_\Gamma |f|^p \, \mathrm{d}\lambda_\Gamma \right)^{\frac{1}{p}}. \tag{2.14}$$

The strong gradient $\nabla_x \phi$ was defined in Definition 2.6 for a differentiable function f at a regular point x. The usual definition of Sobolev Spaces (see [2]) is based on smooth tests functions. Since differentiability is limited to order one on Lipschitz manifolds we choose to replace smooth tests function by Lipschitz test function.

Definition 2.13. *A function $f \in L^p(\Gamma)$ is said to be* **weakly differentiable** *if there exists $\vec{g} \in L^p(\Gamma)^d$ such that for any Lipschitz function $\phi : \Gamma \to \mathbb{R}$ we have*

$$\int_\Gamma f \nabla\phi \, \mathrm{d}\lambda_\Gamma = -\int_\Gamma \phi\vec{g} \, \mathrm{d}\lambda_\Gamma. \tag{2.15}$$

$\vec{g}$ is called the **weak gradient** *of f and denoted $\nabla_\Gamma f$.*

Because Lipschitz manifolds admit only one order of differentiability, we can study only the space of order one weakly differentiable functions. The Sobolev space $W^{1,p}(\Gamma)$ is defined in a similar way to the Euclidean case.

Definition 2.14 (Sobolev space $W^{1,p}(\Gamma)$)**.**
Let $p \in [1, \infty]$, Γ be a compact Lipschitz manifold. The space $W^{1,p}(\Gamma)$ is defined as the set of weakly differentiable functions equipped with the norm

$$||f||^p_{W^{1,p}} = ||f||^p_p + ||\nabla_\Gamma f||^p_p. \tag{2.16}$$

We introduce the following classical notations in the hilbertian case : $H^1(\Gamma) = W^{1,2}(\Gamma)$.

From the Definition 2.14 of Sobolev spaces, it is straightforward that

$$\nabla_\Gamma : W^{1,p}(\Gamma) \to L^p(\Gamma)^d. \tag{2.17}$$

The definition of the weak divergence by a duality approach as well as the study of weak solution of the Poisson problem both require Sobolev spaces with negative index. Following [2, Sections 3.7 to 3.13], we define $W^{-1,p'}(\Gamma)$ using duality.

Definition 2.15 (Sobolev space $W^{-1,p'}(\Gamma)$)**.**
Let $p \in]1, \infty[$, Γ be a compact Lipschitz manifold. The space $W^{-1,p'}(\Gamma)$ is defined as the dual of the space $W^{1,p}(\Gamma)$: $W^{-1,p'}(\Gamma) = (W^{1,p}(\Gamma))'$, where p' is the conjugate of p : $\frac{1}{p'} + \frac{1}{p} = 1$.
It is equipped with the norm

$$||L||_{W^{-1,p'}(\Gamma)} = \sup_{f \in W^{1,p}(\Gamma), f \neq 0} \frac{|L(f)|}{||f||_{W^{1,p}(\Gamma)}}. \tag{2.18}$$

$W^{-1,p'}(\Gamma)$ is the extension to $W^{1,p}(\Gamma)$ of distributions that act normally on infinitely smooth test functions ([2, Section 3.10]).
The **weak divergence** is defined for $p \in]1, \infty[$ as the adjoint of the weak gradient

$$\nabla_\Gamma \cdot : L^p(\Gamma)^d \to W^{-1,p'}(\Gamma). \tag{2.19}$$

Indeed, if $p' < \infty$ is the conjugate of p, $\nabla_\Gamma : W^{1,p'}(\Gamma) \to L^{p'}(\Gamma)^d$ yields an adjoint operator $\nabla_\Gamma \cdot$ from $(L^{p'}(\Gamma)^d)'$ (which is isometrically equivalent to $L^p(\Gamma)^d$) into $(W^{1,p}(\Gamma))' = W^{-1,p'}(\Gamma)$.
The surface divergence of a general function $\vec{f} \in L^p(\Gamma)^d$ is therefore a linear operator acting on $W^{1,p}(\Gamma)$. However, for $\vec{f} \in W^{1,p}(\Gamma)$, $\nabla \cdot \vec{f}$ can be identified by duality with a function in $L^p(\Gamma)$.

2.2.5 The Laplace-Beltrami operator on Lipschitz manifolds

The Laplace-Beltrami operator is classically defined as a second order differential operator on C^2 manifolds (see for instance [4]). The lack of smoothness on a Lipschitz manifold prevents us from making sense of the Laplace–Beltrami operator as a differential operator of order two.
One way of defining the Laplace-Beltrami operator on a Lipschitz manifold Γ is

$$\triangle_\Gamma := \nabla_\Gamma \cdot \nabla_\Gamma : W^{1,p}(\Gamma) \to W^{-1,p'}(\Gamma). \tag{2.20}$$

The following theorem, taken from [22, Theorem 1.3] gives some important properties of the Laplace-Beltrami operator on Lipschitz manifolds on the Sobolev space $H^1(\Gamma) = W^{1,2}(\Gamma)$.

Theorem 2.2 (Laplace-Beltrami on Lipschitz manifolds)**.**
Let Γ be a compact, connected, oriented Lipschitz manifold. The operator $\triangle_\Gamma := \nabla_\Gamma \cdot \nabla_\Gamma$ is well defined, self adjoint and bounded from $W^{1,p}(\Gamma)$ to $W^{-1,p'}(\Gamma)$.
Moreover the operator $-\triangle_\Gamma$ has a purely discrete spectrum

$$0 = \lambda_0 < \lambda_1 < \cdots < \lambda_j < \dots \tag{2.21}$$

with $\lambda_j \to \infty$ as $j \to \infty$.
Furthermore, there is an Hilbertian basis $(\psi_j)_{j\geq 0})$ of $L^2(\Gamma)$ composed of eigenvectors $\psi_j \in H^1(\Gamma)$ of $-\triangle_\Gamma$. The eigenvalues λ_j are listed according to their multiplicity.

Lemma 2.1 (Rayleigh quotient and its consequences)**.**

- *let $u \in H^1_\#(\Gamma) - \{0\}$ the Rayleigh quotient associated to u is*

$$R[u] = \frac{a(u,u)}{\|u\|^2}$$

- *The first non null eigenvalue*

$$\lambda_1 = \inf_{u \in H^1_\#(\Gamma) - \{0\}} R[u],$$

  and the minimum is reached if and only if u is eigenvector associated to λ_1

- *λ_1 is simple and the associated eigenvalue ψ_1 can be chosen to be positive on Γ*

2.3 Preliminary requirements on the existence result on Lipschitz manifolds

2.3.1 Stokes'Theorem

We are now ready to state Stokes' Theorem for Lipschitz forms, which is required for the integration by part (See the proof of Green's Theorem 2.4).

Theorem 2.3 (Stokes' Theorem)**.**
Let Γ be a compact oriented Lipschitz manifold of dimension d. Let ω be a Lipschitz form of degree $d-1$ on Γ. Then

$$\int_\Gamma \mathrm{d}\omega = 0. \tag{2.22}$$

Proof :

Since Γ is compact, there is a finite set I' and a finite number of charts $(U_i, \phi_i)_{i\in I'}$ that covers Γ : $\Gamma = \cup_{i\in I'} U_i$. Let $(\alpha_i)_{i\in I'}$ be a smooth partition of unity subordinate to that cover. The existence of $(\alpha_i)_{i\in I'}$ is given by [31, Proposition 6.14]. Since $\omega = \sum_{i\in I'} \alpha_i\omega$, it is sufficient to prove the theorem for $\alpha_i\omega$ that is, for a Lipschitz form supported within an open subset U_i.

We assume therefore in the following that $supp(\omega) \subset U_i$ where $(U_i, \phi_i = (\phi_i^1, \ldots, \phi_i^d))$ is a local chart. ω being a Lipschitz form of degree $d-1$, for any $j \in \{1, 2, \ldots, d\}$, there exists a Lipschitz function a_j supported in U_i such that almost everywhere

$$\omega(x) = \sum_{j=1}^{d} a_j(x) \mathrm{d}\phi^1 \wedge \mathrm{d}\phi^2 \wedge \cdots \wedge \widehat{\mathrm{d}\phi^j} \wedge \cdots \wedge \mathrm{d}\phi^d, \quad x \in \Gamma. \tag{2.23}$$

The exterior derivative of ω is therefore

$$\begin{aligned} \mathrm{d}\omega(x) &= \sum_{j=1}^{d} \frac{\partial a_j}{\partial \phi^j}(x)\, \mathrm{d}\phi^j \wedge \mathrm{d}\phi^1 \wedge \mathrm{d}\phi^2 \wedge \cdots \wedge \widehat{\mathrm{d}\phi^j} \wedge \cdots \wedge \mathrm{d}\phi^d & (2.24)\\ &= \left(\sum_{j=1}^{d} (-1)^{j-1} \frac{\partial a_j}{\partial \phi^j}(x)\right) \mathrm{d}\phi^1 \wedge \mathrm{d}\phi^2 \wedge \cdots \wedge \mathrm{d}\phi^d, & (2.25) \end{aligned}$$

where ∇a_j denotes the tangent map of a_j. It goes from $T_x\Gamma$ to $\mathbb{R}$ and can therefore be identified with a vector denoted $\left(\frac{\partial a_j}{\partial \phi^1}, \ldots, \frac{\partial a_j}{\partial \phi^d}\right)$.
From the definition of the tangent map (Definition 2.6) we have

$$\frac{\partial a_j}{\partial \phi^j}(\phi_i^{-1}(x)) = \frac{\partial a_j \circ \phi_i^{-1}}{\partial x^j}(x). \tag{2.26}$$

The pullback of the exterior derivative is

$$\begin{aligned} (\phi_i^{-1})^*(\mathrm{d}\omega) &= \left(\sum_{j=1}^{d} (-1)^{j-1} \frac{\partial a_j}{\partial \phi^j}\right) (\phi_i^{-1}(x))\, (\phi_i^{-1})^*(\mathrm{d}\phi^1)(\phi_i^{-1})^*(\mathrm{d}\phi^2) \ldots (\phi_i^{-1})^*(\mathrm{d}\phi^d) & (2.27)\\ &= \left(\sum_{j=1}^{d} (-1)^{j} \frac{\partial a_j}{\partial \phi^j}\right) (\phi_i^{-1}(x))\, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \mathrm{d}x^d., & (2.28) \end{aligned}$$

since $(\phi_i^{-1})^*(\mathrm{d}\phi_i^j) = \mathrm{d}\phi^j \circ \phi_i^{-1} = \mathrm{d}x^j$.
From the definition of form integrals (2.8) :

$$\begin{aligned} \int_{U_i} \mathrm{d}\omega &= \int_{\phi_i(U_i)} (\phi_i^{-1})^*(\mathrm{d}\omega) & (2.29)\\ &= \int_{\phi_i(U_i)} \left(\sum_{j=1}^{d} (-1)^{j-1} \frac{\partial a_j}{\partial \phi^j}\right) (\phi_i^{-1}(x))\, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \mathrm{d}x^d. & (2.30) \end{aligned}$$

Using the tangent map property (2.26) we obtain

$$\int_{U_i} \mathrm{d}\omega = \int_{\phi_i(U_i)} \left(\sum_{j=1}^{d} (-1)^{j} \frac{\partial a_j \circ \phi_i^{-1}}{\partial x^j}\right) (x)\, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \mathrm{d}x^d. \tag{2.31}$$

Since a_j is compactly supported within the open set U_i, a_j is zero in a neighborhood of ∂U_i and therefore on ∂U_i we have $a_j = 0$ and $\nabla a_j = 0$. Similarly on $\partial\phi_i(U_i)$, $a_j \circ \phi_i^{-1} = 0$ and $\nabla a_j \circ \phi_i^{-1} = 0$. We can therefore extend $a_j \circ \phi_i^{-1}$ to $\mathbb{R}^d$ as a Lipschitz function f_j with compact support :

$$f_j(x) = \begin{cases} a_j \circ \phi_i^{-1}(x) & \text{if} \quad x \in U_i \\ 0 & \text{if} \quad x \notin U_i \end{cases} . \tag{2.32}$$

Now (2.31) becomes

$$\begin{aligned} \int_{U_i} \mathrm{d}\omega &= \int_{\mathbb{R}^d} \left(\sum_{j=1}^{d} (-1)^{j-1} \frac{\partial f_j}{\partial x^j} \right) (x)\, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \mathrm{d}x^d. \\ &= \sum_{j=1}^{d} (-1)^{j-1} \int_{\mathbb{R}^d} \frac{\partial f_j}{\partial x^j}(x)\, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \mathrm{d}x^d. \\ &= \sum_{j=1}^{d} (-1)^{j-1} \int_{\mathbb{R}^{d-1}} \left(\int_{-\infty}^{+\infty} \frac{\partial f_j}{\partial x^j}(x)\, \mathrm{d}x^j \right) \wedge\ \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \ldots \\ &\quad \wedge \widehat{\mathrm{d}x^j} \wedge \cdots \wedge \mathrm{d}x^d \end{aligned} \tag{2.33}$$

where (2.33) *is a consequence of Fubini's theorem.*
The result follows from the fact that f_j has compact support.

□

2.3.2 Green's Theorem

The following Green's Theorem connects the divergence and the gradient on Lipschitz manifolds. From a functional analytic point of view, it is an extension of the definition of weak differentiability (2.15) to test functions in $H^1(\Gamma)$.

Theorem 2.4 (Green's Theorem)**.** *Let Γ be a compact oriented Lipschitz manifold of dimension d, $u \in H^1(\Gamma)$ and $\vec{v} \in H^1(\Gamma)^d$. Then*

$$\int_\Gamma \vec{v} \cdot \nabla_\Gamma u\, \mathrm{d}\lambda_\Gamma = -\int_\Gamma u \nabla_\Gamma \cdot \vec{v}\, \mathrm{d}\lambda_\Gamma. \tag{2.34}$$

Proof :
Define the following Lipschitz differential forms of order $(d-1)$:

$$\omega_i = u v_i\, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \widehat{\mathrm{d}x^i} \wedge \cdots \wedge \mathrm{d}x^d, \quad i = 1, \ldots, d,$$

where $v_i, i = 1, \ldots, d$ are the components of v.
Since

$$\mathrm{d}\omega_i = \frac{\partial}{\partial x^i}(u v_i)\, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \mathrm{d}x^d = \frac{\partial}{\partial x^i}(u v_i)\, \mathrm{d}\lambda_\Gamma,$$

we have

$$\nabla_\Gamma \cdot (u\vec{v})\, \mathrm{d}\lambda_\Gamma = \sum_{i=1}^{d} \frac{\partial}{\partial x^i}(u v_i)\, \mathrm{d}\lambda_\Gamma = \sum_{i=1}^{d} \mathrm{d}\omega_i.$$

Given that in a similar way to the strong gradient, the weak gradient satisfies the product rule

$$\nabla_\Gamma \cdot (u\vec{v}) = u\nabla_\Gamma \cdot \vec{v} + \vec{v} \cdot \nabla_\Gamma u,$$

the result follows from Stokes'Theorem 2.3 :

$$\int_\Gamma \vec{v} \cdot \nabla_\Gamma u \, \mathrm{d}\lambda_\Gamma + \int_\Gamma u\nabla_\Gamma \cdot \vec{v} \, \mathrm{d}\lambda_\Gamma = \int_\Gamma \nabla_\Gamma \cdot (u\vec{v}) \, \mathrm{d}\lambda_\Gamma = \sum_{i=1}^{d} \int_\Gamma \mathrm{d}\omega_i = 0.$$

□

A consequence of the Green's Theorem 2.4 is that the operator $-\Delta_\Gamma$ is symmetric and positive. Green's Theorem 2.4 will be used to perform the integration by part needed to obtain the weak formulation (2.40) of the Laplace-Beltrami operator (Theorem 2.2).

2.3.3 Poincaré inequality type

Lemma 2.2 (Poincaré's inequality)**.**
Let Γ be a compact, connected, oriented Lipschitz manifold. There exists a strictly positive constant $c(\Gamma)$ only depending on Γ such that

$$\forall u \in H^1_\#(\Gamma), \quad ||u||_{L^2(\Gamma)} \leq c(\Gamma) \, ||\nabla_\Gamma u||_{L^2(\Gamma)} . \tag{2.35}$$

Proof :
We suppose by contradiction that

$$\forall c(\Gamma) > 0, \exists u \in H^1_\#(\Gamma), \quad ||u||_{L^2(\Gamma)} > c(\Gamma) \, ||\nabla_\Gamma u||_{L^2(\Gamma)} . \tag{2.36}$$

In particular
$\forall k \geq 1$, there exists $u_k \in H^1_\#(\Gamma)$ such that

$$\|u_k\|_2 > k\|\nabla_\Gamma u_k\|_2$$

Necessarily, $u_k \neq 0$ since $\|u_k\|_2 > k\|\nabla_\Gamma u_k\|_2 \geq 0$. We then pose

$$v_k = \frac{u_k}{\|u_k\|_2}, \quad \forall k \geq 1, v_k \in H^1_\#(\Gamma)$$

and

$$\int_\Gamma v_k ds = 0, \quad \|v_k\|_2 = 1, \text{ and } \|\nabla_\Gamma v_k\|_2 < \frac{1}{k}$$

we have

$$\|v_k\|^2_{H^1(\Gamma)} = \|\nabla_\Gamma v_k\|_2^2 + \|v_k\|_2^2 \leq (\frac{1}{k^2} + 1) \leq 2$$

hence the sequence (v_k) is bounded in $H^1(\Gamma)$. Then $(v_k)_k$ admits a sub sequence (v_{l_k}) ($l : \mathbb{N}^\star \to \mathbb{N}^\star$ strictly increasing embedding) weakly convergence to v in $H^1(\Gamma)$. More precisely,

$$\forall u \in (H^1(\Gamma))' \quad \langle v_{l_k}, u\rangle \to \langle v, u\rangle \text{ as } k \to +\infty$$

In particular for all $u \in H^1(\Gamma)$,

$$\int_\Gamma (\nabla_\Gamma v_{l_k} \cdot \nabla_\Gamma u + v_{l_k} u) \, ds \to \int_\Gamma (\nabla_\Gamma v \cdot \nabla_\Gamma u + vu) \, ds, \text{ as } k \to +\infty.$$

Since, the embedding $H^1(\Gamma) \hookrightarrow L^2(\Gamma)$ is compact [4, Theorem 2.34], v_{l_k} converges strongly to v in $L^2(\Gamma)$.

i.e.,

$$\lim_{k\to\infty} \|v_{l_k} - v\|_2 = 0.$$

The inequality

$$|\|v_{l_k}\|_2 - \|v\|_2| \leq \|v_{l_k} - v\|_2,$$

yields the convergence in norm

$$\|v_{l_k}\|_2 \to \|v\|_2 \text{ as } k \to +\infty$$

Therefore $\|v\|_2 = 1$.

The function $\|\nabla_\Gamma \cdot\|_2 \to \mathbb{R}$ is continuous and then semi-continuous below, hence, we have

$$\|\nabla_\Gamma v\|_2 \leq \liminf_{k\to\infty} \|\nabla_\Gamma v_{l_k}\|_2.$$

In particular, $\|\nabla_\Gamma v\|_2 \leq 0 \implies \|\nabla_\Gamma v\|_2 = 0$ and thus v is constant on Γ.

The convergence of v_{l_k} to v implies $\int_\Gamma v_{l_k} \to \int_\Gamma v$. since $\int_\Gamma v_k ds = 0$ therefore $\int_\Gamma v ds = 0$; thus $v \in H^1_\#(\Gamma)$. Given that $v = c \in \mathbb{R}$, we have $c = 0$ and thus $v = 0$. This contradicts $\|v\|_2 = 1$.

□

A classical consequence of Poincaré's inequality is that provided $\int_\Gamma u = 0$, the $H^1_\#(\Gamma)$ norm of u is equivalent to the $H^1(\Gamma)$ norm of u :

$$\int_\Gamma u = 0 \Rightarrow \|\nabla_\Gamma u\|_{L^2(\Gamma)} \leq \|u\|_{H^1(\Gamma)} \leq (1+C)\|\nabla_\Gamma u\|_{L^2(\Gamma)}$$
$$\|u\|_{H^1_\#(\Gamma)} \leq \|u\|_{H^1(\Gamma)} \leq (1+C)\|u\|_{H^1_\#(\Gamma)}. \tag{2.37}$$

The best Poincare's constant is an important geometric invariant of a compact manifold. It is the inverse of the smallest non null eigenvalue of the Laplace-Beltrami operator

2.4 The Poisson problem on a closed Lipschitz surface

Now that we have defined the Laplace-Beltrami operator in Subsection 2.2.5, we can study the Poisson problem on closed Lipschitz manifolds. We start by setting the problem and the relevant functional spaces in Subsection 2.4.1. We then give the weak formulation of the problem in Subsection 2.4.2. We end up by summarizing the existence result in Susection 2.4.3.

2.4.1 Position of the problem

Let Γ be a closed Lipschitz manifold in $\mathbb{R}^3$. Since Γ is closed, it has no boundary, and all the constant functions u are in the kernel of Δ_Γ. We have to impose the global condition $\int_\Gamma u = 0$ to guarantee the uniqueness of solution. Hence we consider the following Poisson problem on a Lipschitz manifold Γ.

$$\begin{cases} -\Delta_\Gamma u = f \text{ on } \Gamma \\ \int_\Gamma u = 0 \end{cases} \tag{2.38}$$

where $f \in L^2_\#(\Gamma)(L^2_\#(\Gamma)$ the real Hilbert space defined in Chapter 1), and u is the unknown function.

In the next section we give the weak form of the Poisson problem (2.38).

2.4.2 Weak form of the Poisson problem on Lipschitz surface

The classical Laplace-Beltrami operator acts on C^2 functions. A classical solution of (2.38) is therefore a function $u \in C^2(\Gamma)$.

We now define a weak form of the Laplace Beltrami operator still denoted by Δ_Γ, which acts on $H^1(\Gamma)$ instead of $C^2(\Gamma)$. The **weak Laplace-Beltrami operator on** Γ, $\Delta_\Gamma : H^1_\#(\Gamma) \to H^1_\#(\Gamma)^*$ is the operator sending $u \in H^1_\#(\Gamma)$ to the linear functional

$$\begin{aligned} H^1_\#(\Gamma) &\to \mathbb{R} \\ v &\to \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma v ds, \end{aligned} \tag{2.39}$$

where $\nabla_\Gamma u$ is the weak gradient of u on Γ (Definition 2.15).

The variational formulation for (2.38) is the following.

$$\text{Find } u \in H^1_\#(\Gamma) \text{ such that } \forall v \in H^1_\#(\Gamma), \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma v ds = \int_\Gamma f v ds. \tag{2.40}$$

We obtain (2.40) from (2.38) using the Green Theorem 2.4.

A solution $u \in H^1_\#(\Gamma)$ of (2.40) is called **weak solution** for the Poisson problem. Indeed, it is only one time weakly differentiable whereas a strong (classical) solution is twice strongly differentiable.

In the next Section we are going to prove that the Poisson problem with a zero mean right hand side admits, a unique solution with zero mean on a closed Lipschitz manifold.

2.4.3 Existence result on Lipschitz manifolds

The existence and uniqueness of weak solutions for the variational formulation (2.40) of problem (2.38) on non smooth manifolds Γ is to our knowledge open. Classical existence results require the smoothness of Γ see for instance [4, Chapter 4, Section 1.2].

Unfortunately on a Lipschitz manifold, similar existence results can not be found in the literature. Besides, as mentioned in [15, Page 3], it is important to check that the space $H^1(\Gamma_h)$ is well defined, where Γ_h results from a triangulation of Γ.

Our existence theorem on Lipschitz manifolds relies on the Stokes' Theorem 2.3 and on the Lax-Milgram's Theorem 1.6.

Theorem 2.5 (Existence theorem on Lipschitz manifolds)**.**
Let Γ be a closed Lipschitz manifold. Let $f \in L^2_\#(\Gamma)$. There exists a unique weak solution $u \in H^1_\#(\Gamma)$ of $-\Delta_\Gamma u = f$ on Γ. Furthermore, u depends continuously on the right hand side:

$$\exists C' > 0, \|u\|_{H^1(\Gamma)} \leq C' \|f\|_{L^2(\Gamma)}. \tag{2.41}$$

Proof. In order to use the Lax-Milgram's Theorem, using the bilinear form

$$\begin{array}{ccc} a(\cdot,\cdot) : H^1_\#(\Gamma) \times H^1_\#(\Gamma) & \to & \mathbb{R} \\ (u,v) & \to & \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma v \, \mathrm{d}\lambda_\Gamma \\ \text{and the linear form} & & \\ b(\cdot) : H^1_\#(\Gamma) & \to & \mathbb{R} \\ v & \to & \int_\Gamma f v \, \mathrm{d}\lambda_\Gamma \end{array}$$

we rewrite the weak formulation (2.40) of (2.38) as

$$a(u,v) = b(v),\ \forall v \in H^1_{\#}(\Gamma)$$

- The continuity of the bilinear form $a(\cdot,\cdot)$ is a consequence of the Cauchy-Schwarz's inequality :

$$\begin{aligned} |a(u,v)| &= |\int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma v \, \mathrm{d}\lambda_\Gamma| \\ &\leq \|\nabla_\Gamma u\|_{L^2(\Gamma)} . \|\nabla_\Gamma v\|_{L^2(\Gamma)} \\ &\leq \|u\|_{H^1(\Gamma)} \|v\|_{H^1(\Gamma)} \end{aligned}$$

- We obtain the coercivity of $a(\cdot,\cdot)$ thanks to the Poincaré's inequality (Lemma 2.2) or more precisely inequality (2.37):

$$\begin{aligned} a(u,u) &= (\nabla_\Gamma u, \nabla_\Gamma u)_{L^2(\Gamma)} \\ &= \|\nabla_\Gamma u\|^2_{L^2(\Gamma)} \\ &\geq \frac{1}{(1+C)^2} \|u\|^2_{H^1(\Gamma)} \end{aligned}$$

- The continuity of $b(\cdot)$ is the consequence of the Cauchy-Schwarz's inequality and the Poincaré's inequality (Lemma 2.2)

$$\begin{aligned} |b(v)| = |(f,v)_{L^2(\Gamma)}| &\leq \|f\|_{L^2(\Gamma)} \|v\|_{L^2(\Gamma)} \\ &\leq \|f\|_{L^2(\Gamma)} \|v\|_{H^1(\Gamma)} \end{aligned}$$

Since the requirement of the Lax-Milgram's theorem are satisfied, the existence of a unique solution to the weak formulation (2.40) satisfying the stability estimate (2.41) holds with $C' = (1+C)^2$. □

2.5 Conclusion

The notion of Lipschitz manifold has been studied by some authors [22, 24] and is usefull in laying the foundation of the finite element method on general compact manifolds. We showed how it allows the proper definition of tangent spaces, gradient and divergence operators almost everywhere. Differentiability almost everywhere thanks to Rademacher's Theorem is the key ingredient of the calculus on Lipschtz manifolds. We have proposed proofs of the Stokes' Theorem 2.3 and Green's Theorem 2.4 as well as a Poincaré's inequality type. This enables the weak formulation of the Poisson problem. The well-posedness of the weak formulation of the Poisson problem is then obtained thanks to the Lax-Milgram Theorem.

In the next Chapter, we introduce the linear finite element method and investigate the estimate of the condition number

⋆ ⋆ CHAPTER THREE ⋆ ⋆

SURFACE FINITE ELEMENTS METHOD AND CONDITION NUMBER

3.1 Introduction

Numerically solving partial differential equations on surfaces is a very active research topic, and finite element methods play a central part in many approaches. The finite elements method is one of the main tools for the numerical calculation of solution of elliptic boundary value problems. It is also used for parabolic or hyperbolic partial differential equations. This method is based on the variational formulation that we have studied in Chapter 2. The idea is to replace the Hilbert space $H^1_\#(\Gamma)$ on which we pose the variational formulation by subspaces $PL(\Gamma_h)$ of finite dimension. The approximated problem posed over $PL(\Gamma_h)$ reduces to the simple solution of the linear system, whose matrix is called the **stiffness matrix**. We can choose $PL(\Gamma_h)$ such a way that the subspace $PL(\Gamma_h)$ is a good approximation of $H^1_\#(\Gamma)$ and that the solution $u_h \in PL(\Gamma_h)$ of the variational formulation is close to the exact solution $u \in H^1_\#(\Gamma)$. The finite element method has been studied by several authors [3, 9, 15, 16].

3.1.1 General internal approximation

Given a continuous bilinear form $a(\cdot,\cdot)$ on $H^1_\#(\Gamma) \times H^1_\#(\Gamma)$ ($H^1_\#(\Gamma)$ the real Hilbert space defined in Chapter 1) and $H^1_\#(\Gamma)$-coercive, and given a continuous linear form $l(\cdot)$ on $H^1_\#(\Gamma)$, we consider the weak formulation

$$\text{Find } u \in H^1_\#(\Gamma) \text{ such that } a(u,v) = l(v) \qquad \forall v \in H^1_\#(\Gamma) \tag{3.1}$$

which has a unique solution by the Lax-Milgram Theorem 1.6. The internal approximation of (3.1) consists of replacing the Hilbert space $H^1_\#(\Gamma)$ by an appropriate finite dimensional subspaces $PL(\Gamma_h)$, $h \in (0,1]$, converging to $H^1_\#(\Gamma)$ as $h \to 0$, and look for the h-solution of:

$$\text{Find } u_h \in PL(\Gamma_h) \text{ such that } a(u_h,v) = l(v) \qquad \forall v \in PL(\Gamma_h) \tag{3.2}$$

To put (3.2) in a simple form, we introduce a basis $(\phi^h_i), i = 1, \cdots, n$ of $PL(\Gamma_h)$. Assuming u_h has the form

$$u_h = \sum_{i=1}^{n} u_i \phi^h_i, \text{ with } u_i \in \mathbb{R} \tag{3.3}$$

we are led to the system of equations

$$\sum_{j=1}^{n} a(u_j\phi_j^h, \phi_i^h) = l(\phi_i^h), \qquad i = 1, \cdots, n \tag{3.4}$$

Which can be written in the form of linear system

$$\mathcal{A}_h U = b_h \tag{3.5}$$

where $\mathcal{A}_h$ is the stiffness matrix with entries $A_{ij}^h = a(\phi_i^h, \phi_j^h)$ and b_h is the load vector with entries $b_i^h = l(\phi_i^h)$.

The Jean Céa Lemma express the error in terms of the exact solution. For example, the best approximation result states that the error $u - u_h$ satisfies

$$\|u - u_h\|_{H^1_{\#}(\Gamma)} \leq \frac{C_a}{c_a} \inf_{v_h \in PL(\Gamma_h)} \|u - v_h\|_{H^1_{\#}(\Gamma)} \tag{3.6}$$

where u is the solution of (3.1) and u_h that of (3.2), C_a being a continuity constant and c_a a coercivity constant of the bilinear form $a(\cdot, \cdot)$ on $H^1_{\#}(\Gamma) \times H^1_{\#}(\Gamma)$.

The following lemma gives the convergence of the variational approximation. More precisely, as $h \to 0$, the internal approximation (3.2) converges toward the variational formulation (3.1).

Lemma 3.1. *Let u be the solution of (3.1) and u_h that of (3.2). We assume that there exists a subspace $\mathcal{V} \subset H^1_{\#}(\Gamma)$ which is dense in $H^1_{\#}(\Gamma)$ and a mapping I from $\mathcal{V}$ into $H^1_{\#}(\Gamma)$ (called an **interpolation operator**) such that:*

$$\lim_{h \to 0} \|v - I(v)\|_{H^1_{\#}(\Gamma)} = 0 \qquad \forall v \in \mathcal{V}. \tag{3.7}$$

Then the method of internal variational approximation converges, that is

$$\lim_{h \to 0} \|u - u_h\|_{H^1_{\#}(\Gamma)} = 0. \tag{3.8}$$

To obtain a numerical approximation of the exact solution of (3.1), we must introduce a finite dimensional space $PL(\Gamma_h)$ in order to solve a simple linear system associated with the internal variational approximation (3.2). But the choice of $PL(\Gamma_h)$ is not obvious and the finite element method consists of providing such good space $PL(\Gamma_h)$. Let us say something about the Galerkin method which appears in this subsection.

3.1.2 Principle of Galerkin finite element method

The Galerkin method appears in the framework of internal variational approximation studied above. Here the interpolation operator I is simply the orthogonal projection over $PL(\Gamma_h)$ which is defined in all $H^1_{\#}(\Gamma)$ and not only on $\mathcal{V}$. It is the precursor of the finite element method. It is very useful from the theoretical point of view. Despite of this, the Galerkin method does not have any numerical interest in general from a numerical point of view because all the coefficients of the matrix $\mathcal{A}_h$ are nonzero in general, and that is, the numerical solution of linear system will be unstable and very sensitive to rounding errors on the computer.

From this point of view, the finite element method is more powerful and more preferable to the Galerkin method when numerically solving elliptic partial differential equations since the finite element matrice is sparse. [3, 9].

3.1.3 Finite element method

The finite element method (FEM), is a numerical method for solving problems of engineering and mathematical physics. Typical problems include structural analysis, heat transfer, fluid flow, mass transport, and electromagnetic potential [17]. The solution of these problems generally requires the solution to boundary value problems for partial differential equations. The FEM then uses weak forms from the calculus of variations to approximate a solution by minimizing an associated functional. To solve the problem, the domain is subdivided into smaller, simpler parts that are called finite elements. The simple equations that model these finite elements are then assembled into a larger system of equations that models the entire problem. The finite element approximation project the original problem onto a finite dimensional space. This results in a system of algebraic equations. The method yields approximate values of the unknowns on a finite number of points over the domain [3].

3.2 Finite element discretization on triangulated surfaces

Following Ciarlet [12, P. 93] we have the following definitions

Definition 3.1. *A finite element is a triplet $(\mathcal{K}, \mathcal{P}, \mathcal{N})$, where*

1. *$\mathcal{K} \subset \mathbb{R}^n$ is a bounded closed set with non empty interior and piecewice smooth boundary*
2. *$\mathcal{P}$ is a finite dimensional space of functions on $\mathcal{K}$ (the space of shape functions)*
3. *$\mathcal{N} = \{N_1, N_2, \cdots, N_k\}$ is a basis for the dual space of $\mathcal{P}$ (the set of nodal variables or the dual basis).*

Definition 3.2. *Let $(\mathcal{K}, \mathcal{P}, \mathcal{N})$ be a finite element. The basis $\{\phi_1, \phi_2, \cdots, \phi_k\}$ of $\mathcal{P}$ dual to $\mathcal{N}$ (i.e., $N_i(\phi_j) = \delta_i^j$) is called the nodal basis of $\mathcal{P}$.*

We consider a triangulated surface Γ_h. We are going to approximate the solution that belong to $H^1(\Gamma)$ by its projection on the space $H^1(\Gamma_h)$ of continuous piecewise affine functions on Γ_h. Those are called linear elements on the surface Γ_h, i.e. u_h is a linear polynomial on each triangle on Γ_h and globally continuous. In what follows, we recall the Delaunay triangulation in Subsection 3.2.2, the assumption on the triangulation in Subsection 3.2.3, the analysis of finite elements matrix in Subsection 3.2.5, the convergence of discrete solution toward the continuous solution in Subsection 3.2.6.

3.2.1 Discrete approximation of a surface by a triangular mesh

In this Subsection, the notion of triangulation is recalled with some few examples.

Definition 3.3 (Triangulation)**.**

A triangulation of a point set $\mathbf{P}$ is a set of triangles $\mathcal{T}$, such that we have the following:

1. *All points, $p_i \in \mathcal{P}$, are vertices in at least one triangle.*
2. *The interiors of any two triangles do not intersect.*
3. *All the points p_i only intersect the triangles $T_i \in \mathcal{T}$ at the triangles vertices.*
4. *The union of all the vertices of the triangles is $\mathbf{P}$.*

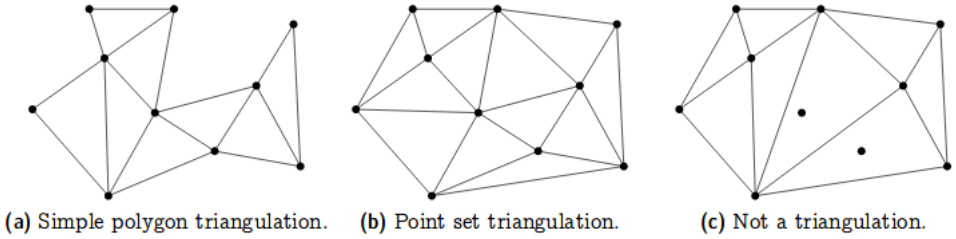


Figure 3.1: Examples of (non-) triangulations [5]

3.2.2 Delaunay Triangulation

Following [5], we have the following definitions and properties

Definition 3.4 (Empty Circle)**.**

A circle is empty if and only if its interior contains no points of $\mathbf{P}$.

Definition 3.5 (Circumcircle of an Edge)**.**

A circumcircle of an edge from p_i *to* p_j *is a circle going through* p_i *and* p_j .

Note that for a given edge there are infinitely many circumcircles.

Definition 3.6 (Delaunay Edge)**.**

An edge is Delaunay if and only if it has an empty circumcircle.

Definition 3.7 (Delaunay Triangulation)**.**

A triangulation, $\boldsymbol{D}$*, is a Delaunay triangulation, if an edge is Delaunay if and only if it is in* $\boldsymbol{D}$.

Definition 3.8 (Circumcircle of a Triangle)**.**

The circumcircle of a triangle is the unique circle going through its three vertices.

Definition 3.9 (Delaunay Triangle)**.**

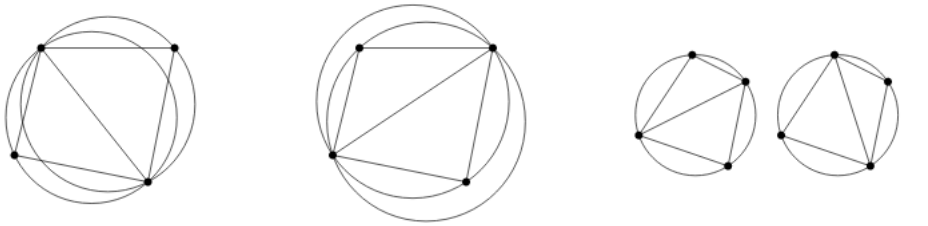
A triangle is Delaunay if and only if its circumcircle is empty.

The concepts of Delaunay triangles and Delaunay edges are connected, as stated in the following Lemma.

Lemma 3.2.
For a given triangulation $\mathcal{T}$ *of the point set* $\mathbf{P}$*, all the triangles of* $\mathcal{T}$ *are Delaunay if, and only if, all the edges are Delaunay.*

Proof. See, [5] □

A common problem in triangulation is that some triangles become too skinny, in that one of their angles becomes too small. This, among other, relates to finite element simulations, where very skinny triangles can cause numerical instability. To avoid this as best as possible the Delaunay triangulation is a good choice, since among all triangulation of a point set $\mathbf{P}$ the Delaunay triangulation maximizes the minimal angle [30, 53].



(a) Delaunay triangulation. (b) Non-Delaunay triangulation. (c) Two Delaunay triangulations.

Figure 3.2: Triangulations of four points in convex positions [5]

After recalling the notion of Delaunay triangulation, we are concerned with the partition of a closed manifold Γ. One can show that each point $x \in U_{\epsilon,\Gamma}$ can be uniquely projected onto the smooth surface, yielding the fermi-coordinates (Theorem 1.1). Given a refinement of Γ_h, a new refined triangulation of Γ may be obtained by projecting the new nodes of Γ_h onto Γ [16].

3.2.3 Assumptions on the triangulation

The author [15] assumed the existence of a polyhedron surface Γ_h made of triangles with nodes on Γ. Furthermore, in order to define a lift operator (3.9) from Γ_h onto Γ, it is assumed that $\Gamma_h \subset U_{\epsilon,\Gamma}$ where $U_{\epsilon,\Gamma}$ is a neighborhood of Γ defined in (1.3). Moreover in order to avoid a double covering of Γ by Γ_h, it is required that for any $y \in \Gamma_h$ there is a unique $x \in \Gamma$ such that

$$y = x + d_\epsilon(y)\mathbf{n}(x) = a_\epsilon(x_h) + d_\epsilon(x)\mathbf{n}(a_\epsilon(y))$$

so that each triangle $T_{h_k} \subset \Gamma_h$ is in bijection with a curved triangle $\bar{T}_h = a_\epsilon(T_{h_k}) \subset \Gamma$. We recall that both T_{h_k} and $\bar{T}_h$ share the same nodes.
In order to compare an approximate solution u_h on Γ_h to the exact solution u which is defined on Γ, we need to introduce a lift operator L such that

$$\begin{array}{rccc} L : C(\Gamma_h) & \to & C(\Gamma) \\ u_h & \to & u_h \circ a_\epsilon^{-1} \end{array}, \tag{3.9}$$

and

$$\forall y \in \Gamma_h, \quad u_h(y) = Lu_h(a_\epsilon(y)) = Lu_h(y - d_\epsilon(y)\mathbf{n}(y)) = u_h(a_\epsilon^{-1}(y - d_\epsilon(y)n(y))). \tag{3.10}$$

provided

$$\Gamma_h \subset U_{\epsilon_F,\Gamma}. \tag{3.11}$$

However the use of Fermi coordinates (1.4) in the definition of the lift operator L (3.9) requires not only that the approximating surface Γ_h being included in $U_{\epsilon,\Gamma}$ but also that the projection operator a be one way between Γ_h and Γ.
To build a triangulation satisfying the above assumptions, the nodes of the triangulation must be close to each other in order to avoid the approximated surface crossing itself or having points outside $U_{\epsilon,\Gamma}$.

In Section 3.2.4, we look for $\tilde{u}_h \in PL_\#(\Gamma_h)$ the projection of the solution $u_h \in H^1_\#(\Gamma_h)$ of the Poisson problem

$$-\triangle_{\Gamma_h} u_h = f_h, \tag{3.12}$$

on the space of continuous piecewise linear functions with zero mean $PL_\#(\Gamma_h)$.

3.2.4 The finite element operator

We consider a triangulated surface Γ_h that satisfies the assumptions of the previous section. In a similar fashion as in equations (2.39), the **weak Laplace-Beltrami operator on** Γ_h is the operator sending $v \in H^1_\#(\Gamma_h)$ to the linear functional

$$\begin{aligned} H^1_\#(\Gamma_h) &\to \mathbb{R} \\ w &\to \int_{\Gamma_h} \nabla_{\Gamma_h} v \cdot \nabla_{\Gamma_h} w, \end{aligned} \tag{3.13}$$

where $\nabla_{\Gamma_h} v$ is the weak gradient of v on Γ_h (Definition 2.15).

We set $PL_\#(\Gamma_h) \subset H^1_\#(\Gamma_h)$, the subspace of piecewise linear functions on Γ_h that have zero mean. Functions ϕ of $PL_\#(\Gamma_h)$ are linear in the sense that they take the form $\phi(\overrightarrow{x}) = \alpha^{\mathcal{T}} x + \beta^{\mathcal{T}} y + \gamma^{\mathcal{T}} z + \zeta^{\mathcal{T}}$ on each of the triangles $\mathcal{T}$ composing Γ_h.
The discrete form of the variational formulation of the Poisson equation (2.38) is the following.

$$\text{Find } \tilde{u}_h \in PL_\#(\Gamma_h) \text{ such that } \forall v \in PL_\#(\Gamma_h), \int_{\Gamma_h} \nabla_{\Gamma_h} \tilde{u}_h \cdot \nabla_{\Gamma_h} v = \int_{\Gamma_h} f_h v. \tag{3.14}$$

In a similar fashion as in equations (2.39), the **discrete Laplace-Beltrami operator** is the operator sending $\tilde{u}_h \in PL_\#(\Gamma_h)$ to the linear functional

$$\begin{aligned} PL_\#(\Gamma_h) &\to \mathbb{R} \\ v &\to \int_{\Gamma_h} \nabla_{\Gamma_h} \tilde{u}_h \cdot \nabla_{\Gamma_h} v, \end{aligned} \tag{3.15}$$

where $\nabla_{\Gamma_h} \tilde{u}_h$ is the weak gradient of $\tilde{u}_h$ on Γ_h (Definition 2.15).

We now express the coefficients of its matrix $A_{\triangle_{\Gamma_h}}$ using a nodal basis.
Indeed $PL_\#(\Gamma_h)$ is generated by the nodal functions $\phi_i^h : \Gamma_h \to \mathbb{R}, \quad i = 1, ..., n$, such that $\phi_i^h(x_j) = \delta_{ij}$.

A solution $\tilde{u}_h$ of the form $\tilde{u}_h = \sum_{i=1}^n u_i \phi_i^h$ of the discrete Poisson problem (3.14) must satisfy the following system of equations

$$\forall i \in \{1, ..., n\}, \qquad \int_{\Gamma_h} \nabla_{\Gamma_h} \tilde{u}_h \cdot \nabla_{\Gamma_h} \phi_i^h = \int_{\Gamma_h} f_h \phi_i^h. \tag{3.16}$$

which takes the algebraic form

$$A_{\triangle_{\Gamma_h}} X = b_h, \tag{3.17}$$

$A_{\triangle_{\Gamma_h}} = (a_{ij}^h)_{i,j=1,...,n}$, $X = {}^t(u_1, ..., u_n)$ and $b_h = {}^t(b_1^h, ..., b_n^h)$ with

$$a_{ij}^h = \int_{\Gamma_h} \nabla_{\Gamma_h} \phi_i^h \cdot \nabla_{\Gamma_h} \phi_j^h, \tag{3.18}$$

$$b_j^h = \int_{\Gamma_h} f \phi_j^h. \tag{3.19}$$

Theorem 3.1 (Properties of $A_{\triangle_{\Gamma_h}}$)**.**
Consider a polyhedral surface Γ_h composed of n nodes. The finite element matrix $A_{\triangle_{\Gamma_h}}$ satisfies

- $A_{\triangle_{\Gamma_h}}$ *is symmetric and positive*
- $\ker A_{\triangle_{\Gamma_h}}$ *is the set of constant functions*
- $ImA_{\triangle_{\Gamma_h}}$ *is the set of functions with zero mean*
- $A_{\triangle_{\Gamma_h}}$ *is sparse : it has a very few number of nonzero elements [43]*

Proof:

- $A_{\triangle_{\Gamma_h}}$ is symmetric

$$\forall i,j \in \{1,...,n\}, \qquad a_{ij}^h = \int_{\Gamma_h} \nabla_{\Gamma_h}\phi_i^h \cdot \nabla_{\Gamma_h}\phi_j^h = \int_{\Gamma_h} \nabla_{\Gamma_h}\phi_j^h \cdot \nabla_{\Gamma_h}\phi_i^h = a_{ji}^h.$$

  Using the definition of the discrete Laplace-Beltrami (3.15) and the expression of $\tilde{u}_h$ we obtain

$$ {}^tX A_{\triangle_{\Gamma_h}} X = \int_{\Gamma} ||\nabla_{\Gamma_h}\tilde{u}_h||^2_{L^2(\Gamma_h)} \geq 0. \tag{3.20}$$

  Hence $A_{\triangle_{\Gamma_h}}$ is positive.

- Let $\tilde{u}_h \in \ker A_{\triangle_{\Gamma_h}}$. (3.20) yields $||\nabla_{\Gamma_h}\tilde{u}_h||^2_{L^2(\Gamma_h)} = 0$. This implies $||\nabla_{\Gamma_h}\tilde{u}_h||_{L^2(\Gamma_h)} = 0$ Thus $\tilde{u}_h =$ constant on each simplex of triangulation separately. Since $\tilde{u}_h \in C^0(\Gamma_h)$ then $\tilde{u}_h$ is constant on Γ_h

- Because a_{ij}^h is non zero only when the nodes i et j are the vertices of a common triangle. Roughly speaking, for a given node i, the coefficient a_{ij}^h will be nonzero only when the node j is one of the nodes of a triangle that is adjacent to the node i

  The matrix $A_{\triangle_{\Gamma_h}}$ is built by summing up the contributions of all triangles by applying the formula

$$a(\phi_i^h, \phi_j^h) = \sum_{\mathcal{T}\in\mathcal{T}_h} a_{\mathcal{T}}(\phi_i^h, \phi_j^h) = \sum_{\mathcal{T}\in\mathcal{T}_h} \int_{\mathcal{T}} \nabla_\Gamma \phi_i^h . \nabla_\Gamma \phi_j^h.$$

  $a_{\mathcal{T}}(\phi_i^h, \phi_j^h)$ is zero unless the node i and j are both vertices of $\mathcal{T}$.

□

$A_{\triangle_{\Gamma_h}}$ is non invertible since constants are in its kernel (Theorem 3.1), hence the linear system (3.17) is singular. However it admits a unique solution with zero mean provided the right hand side has zero mean.

Definition 3.10 (M-matrix)**.**
A matrix $A = (a_{ij})_{i,j=1,...,n}$ is said to be a M-matrix if it satisfies the following properties (see Definition 1.4 in [43]):

1. $a_{ii} > o$, *for* $i = 1, ..., n$.
2. $a_{ij} \leq 0$ *for* $i \neq j$, *i,j= 1,...,n.*
3. A *is non singular.*
4. A^{-1} *is positive*

3.2.5 Filling of matrices and computation

For the filling of matrices $A_{\triangle_{\Gamma_h}}$ and b_h in the equality (3.17), each coefficient a_{ij}^h and b_j^h is computed as a sum of contributions coming from each triangle:

$$a_{ij}^h = \int_{\Gamma_h} \nabla_\Gamma \phi_i^h . \nabla_\Gamma \phi_j^h = \sum_k \int_{\mathcal{T}_{hk}} \nabla \phi_i^h . \nabla \phi_j^h = \sum_k a_{ij}^{hk}$$

$$b_i^h = \int_{\Gamma_h} f \phi_i = \sum_k \int_{\mathcal{T}_{hk}} f \phi_i^h = \sum_k b_i^{hk}.$$

where $a_{ij}^{hk} = \int_{\mathcal{T}_{hk}} \nabla \phi_i^h . \nabla \phi_j^h$ and $b_i^{hk} = \int_{\mathcal{T}_{hk}} f \phi_i^h$.

Setting $s_1^{\mathcal{T}}$, $s_2^{\mathcal{T}}$ and $s_3^{\mathcal{T}}$ the indices of three nodes of each triangle $\mathcal{T}_{hk}$, the contribution b_i^{hk} is obtained by a quadratic formula

$$b_i^{hk} \approx \frac{|\mathcal{T}_{hk}|}{3} \left(f(\vec{x}_{s_1^{\mathcal{T}}}) \phi_i^h(\vec{x}_{s_1^{\mathcal{T}}}) + f(\vec{x}_{s_2^{\mathcal{T}}}) \phi_i^h(\vec{x}_{s_2^{\mathcal{T}}}) + f(\vec{x}_{s_3^{\mathcal{T}}}) \phi_i^h(\vec{x}_{s_3^{\mathcal{T}}}) \right),$$

where only one of the three terms in the brackets is nonzero and $|\mathcal{T}_{hk}|$ is the area of the triangle $\mathcal{T}_{hk}$.

For computing a_{ij}^{hk}, we observe that the mesh $\mathcal{T}_h$ of the domain Γ_h is composed of triangular elements $(\mathcal{T}_{hk})$ having non zero area. The n vertices of Γ_h are denoted $\vec{x}_1,\ \vec{x}_2,\ \vec{x}_3,\ ...,\ \vec{x}_n$. To each vertex $\vec{x}_i$, we associate a nodal function, $\phi_i^h : \Gamma_h \to \mathbb{R}$ of $PL_\#(\Gamma_h)$ such that $\phi_i^h(x_j) = \delta_{ij}$.

In $3D$, functions $\phi \in PL_\#(\Gamma_h)$ take the following form on each triangle $\mathcal{T} \in \mathcal{T}_h$ $(\mathcal{T}_{hk} \equiv \mathcal{T})$

$$\phi(\vec{x}) = \alpha^{\mathcal{T}} x + \beta^{\mathcal{T}} y + \gamma^{\mathcal{T}} z + \zeta^{\mathcal{T}}. \tag{3.21}$$

The gradient of any function $\phi \in PL_\#(\Gamma_h)$ is thus constant on each triangle $\mathcal{T} \in \mathcal{T}_h$ and its components can be deduced from (3.21):

$$\forall \vec{x} \in \mathcal{T}, \quad \nabla_{\Gamma_h} \phi(\vec{x}) = \left(\alpha^{\mathcal{T}}, \beta^{\mathcal{T}}, \gamma^{\mathcal{T}} \right). \tag{3.22}$$

To determine the gradient of a function $\phi \in PL_\#(\Gamma_h)$ on a triangle $\mathcal{T}$, we must first consider that ϕ is invariant in the normal direction to cell $\mathcal{T}$. This can be expressed as a linear constraint on the gradient

$$n_x^{\mathcal{T}} \alpha^{\mathcal{T}} + n_y^{\mathcal{T}} \beta^{\mathcal{T}} + n_z^{\mathcal{T}} \gamma^{\mathcal{T}} = 0. \tag{3.23}$$

Secondly, we take the values of ϕ at each node of $\mathcal{T}$. Denoting $s_1^{\mathcal{T}}$, $s_2^{\mathcal{T}}$ and $s_3^{\mathcal{T}}$ the three nodes of $\mathcal{T}$, we have the following system:

$$\begin{pmatrix} x_{s_1^{\mathcal{T}}} & y_{s_1^{\mathcal{T}}} & z_{s_1^{\mathcal{T}}} & 1 \\ x_{s_2^{\mathcal{T}}} & y_{s_2^{\mathcal{T}}} & z_{s_2^{\mathcal{T}}} & 1 \\ x_{s_3^{\mathcal{T}}} & y_{s_3^{\mathcal{T}}} & z_{s_3^{\mathcal{T}}} & 1 \end{pmatrix} \begin{pmatrix} \alpha^{\mathcal{T}} \\ \beta^{\mathcal{T}} \\ \gamma^{\mathcal{T}} \\ \zeta^{\mathcal{T}} \end{pmatrix} = \begin{pmatrix} \phi(\vec{x}_{s_1^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_2^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_3^{\mathcal{T}}}) \end{pmatrix}. \tag{3.24}$$

From (3.23) and (3.24) we obtain the following system:

$$\begin{pmatrix} x_{s_1^{\mathcal{T}}} & y_{s_1^{\mathcal{T}}} & z_{s_1^{\mathcal{T}}} & 1 \\ x_{s_2^{\mathcal{T}}} & y_{s_2^{\mathcal{T}}} & z_{s_2^{\mathcal{T}}} & 1 \\ x_{s_3^{\mathcal{T}}} & y_{s_3^{\mathcal{T}}} & z_{s_3^{\mathcal{T}}} & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{pmatrix} \begin{pmatrix} \alpha^{\mathcal{T}} \\ \beta^{\mathcal{T}} \\ \gamma^{\mathcal{T}} \\ \zeta^{\mathcal{T}} \end{pmatrix} = \begin{pmatrix} \phi(\vec{x}_{s_1^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_2^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_3^{\mathcal{T}}}) \\ 0 \end{pmatrix}. \tag{3.25}$$

Taking in particular ϕ as a nodal functions ϕ_i, one has that $\phi_{s_i^{\mathcal{T}}}(\vec{x}_{s_j^{\mathcal{T}}}) = \delta_{ij}$, the system (3.25) has four unknowns $\alpha^{\mathcal{T}}$, $\beta^{\mathcal{T}}$, $\gamma^{\mathcal{T}}$ and $\zeta^{\mathcal{T}}$. The determinant of this system (3.25) is equal to $-2|\mathcal{T}|$,

where $|\mathcal{T}|$ is the area of the triangle $\mathcal{T}$ which is assumed to be nonzero.

We use the Cramer formulae to express the solution of (3.25) :

$$\nabla_{\Gamma_h}\phi(\vec{x})|_{\mathcal{T}} = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} \phi(\vec{x}_{s_1^{\mathcal{T}}}) & y_{s_1^{\mathcal{T}}} & z_{s_1^{\mathcal{T}}} & 1 \\ \phi(\vec{x}_{s_2^{\mathcal{T}}}) & y_{s_2^{\mathcal{T}}} & z_{s_2^{\mathcal{T}}} & 1 \\ \phi(\vec{x}_{s_3^{\mathcal{T}}}) & y_{s_3^{\mathcal{T}}} & z_{s_3^{\mathcal{T}}} & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & \phi(\vec{x}_{s_1^{\mathcal{T}}}) & z_{s_1^{\mathcal{T}}} & 1 \\ x_{s_2^{\mathcal{T}}} & \phi(\vec{x}_{s_2^{\mathcal{T}}}) & z_{s_2^{\mathcal{T}}} & 1 \\ x_{s_3^{\mathcal{T}}} & \phi(\vec{x}_{s_3^{\mathcal{T}}}) & z_{s_3^{\mathcal{T}}} & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & y_{s_1^{\mathcal{T}}} & \phi(\vec{x}_{s_1^{\mathcal{T}}}) & 1 \\ x_{s_2^{\mathcal{T}}} & y_{s_2^{\mathcal{T}}} & \phi(\vec{x}_{s_2^{\mathcal{T}}}) & 1 \\ x_{s_3^{\mathcal{T}}} & y_{s_3^{\mathcal{T}}} & \phi(\vec{x}_{s_3^{\mathcal{T}}}) & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}$$

We deduce the gradient of the nodal shape functions $\phi_{s_1^{\mathcal{T}}}$, $\phi_{s_2^{\mathcal{T}}}$ and $\phi_{s_3^{\mathcal{T}}}$ taking into account the fact that $\phi_i(\vec{x}_j) = \delta_{ij}^h \quad \forall i,j \in \{s_1^{\mathcal{T}}, s_2^{\mathcal{T}}, s_3^{\mathcal{T}}\}$:

$$\nabla_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\vec{x}) = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} 1 & y_{s_1^{\mathcal{T}}} & z_{s_1^{\mathcal{T}}} & 1 \\ 0 & y_{s_2^{\mathcal{T}}} & z_{s_2^{\mathcal{T}}} & 1 \\ 0 & y_{s_3^{\mathcal{T}}} & z_{s_3^{\mathcal{T}}} & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & 1 & z_{s_1^{\mathcal{T}}} & 1 \\ x_{s_2^{\mathcal{T}}} & 0 & z_{s_2^{\mathcal{T}}} & 1 \\ x_{s_3^{\mathcal{T}}} & 0 & z_{s_3^{\mathcal{T}}} & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & y_{s_1^{\mathcal{T}}} & 1 & 1 \\ x_{s_2^{\mathcal{T}}} & y_{s_2^{\mathcal{T}}} & 0 & 1 \\ x_{s_3^{\mathcal{T}}} & y_{s_3^{\mathcal{T}}} & 0 & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}$$

$$\nabla_{\Gamma_h}\phi_{s_2^{\mathcal{T}}}(\vec{x}) = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} 0 & y_{s_1^{\mathcal{T}}} & z_{s_1^{\mathcal{T}}} & 1 \\ 1 & y_{s_2^{\mathcal{T}}} & z_{s_2^{\mathcal{T}}} & 1 \\ 0 & y_{s_3^{\mathcal{T}}} & z_{s_3^{\mathcal{T}}} & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & 0 & z_{s_1^{\mathcal{T}}} & 1 \\ x_{s_2^{\mathcal{T}}} & 1 & z_{s_2^{\mathcal{T}}} & 1 \\ x_{s_3^{\mathcal{T}}} & 0 & z_{s_3^{\mathcal{T}}} & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & y_{s_1^{\mathcal{T}}} & 0 & 1 \\ x_{s_2^{\mathcal{T}}} & y_{s_2^{\mathcal{T}}} & 1 & 1 \\ x_{s_3^{\mathcal{T}}} & y_{s_3^{\mathcal{T}}} & 0 & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}$$

$$\nabla_{\Gamma_h}\phi_{s_3^{\mathcal{T}}}(\vec{x}) = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} 0 & y_{s_1^{\mathcal{T}}} & z_{s_1^{\mathcal{T}}} & 1 \\ 0 & y_{s_2^{\mathcal{T}}} & z_{s_2^{\mathcal{T}}} & 1 \\ 1 & y_{s_3^{\mathcal{T}}} & z_{s_3^{\mathcal{T}}} & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & 0 & z_{s_1^{\mathcal{T}}} & 1 \\ x_{s_2^{\mathcal{T}}} & 0 & z_{s_2^{\mathcal{T}}} & 1 \\ x_{s_3^{\mathcal{T}}} & 1 & z_{s_3^{\mathcal{T}}} & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} x_{s_1^{\mathcal{T}}} & y_{s_1^{\mathcal{T}}} & 0 & 1 \\ x_{s_2^{\mathcal{T}}} & y_{s_2^{\mathcal{T}}} & 0 & 1 \\ x_{s_3^{\mathcal{T}}} & y_{s_3^{\mathcal{T}}} & 1 & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}.$$

3.2.6 Convergence of discrete solution towards the continuous solution

In order to study the convergence of the finite element approximation, we need to compare $u \in H^1(\Gamma)$ with $\tilde{u}_h \in H^1(\Gamma_h)$ but don't share the same support. Hence we need to use the lift operator (3.9) which requires the assumption (3.11) that the triangulated surface Γ_h is closed enough to Γ.

As the parameter h goes to zero the distance between Γ_h and Γ converges to zero as expressed in the following theorem (See [16, Lemma 4.1]).

Theorem 3.2 (Convergence of Γ_h towards Γ)**.**
Let $\Gamma \in \mathbb{R}^3$ be an embedded C^2 hypersurface and h be the largest diameter of trianges in $\Gamma_h \subset U_{\epsilon_F,\Gamma}$ a piecewise linear surface. There exists a constant c such that

$$\forall x \in \Gamma_h, \quad dist(x,\Gamma) \leq ch^2.$$

Once proven that Γ_h converges towards Γ, we have the following convergence theorem which is a combination in [15] of Theorem 8 and Lemma 6, 7.

Theorem 3.3 (Convergence of $\tilde{u}_h$ towards u)**.**
Let $\Gamma \in \mathbb{R}^3$ be an embedded C^2 hyper surface and let h be the largest diameter of triangles in $\Gamma_h \subset U_{\epsilon_F,\Gamma}$.
If u is a continuous solution of the Poisson problem (2.38) *and $\tilde{u}_h$ is the discrete solution of* (3.14)*, then there exists $c > 0$ such that*

$$\|u - \tilde{u}_h \circ a_\epsilon^{-1}\|_{L^2(\Gamma)} \leq ch^2, \quad \|\nabla_\Gamma(u - \tilde{u}_h \circ a_\epsilon^{-1})\|_{L^2(\Gamma)} \leq ch. \tag{3.26}$$

Following [16], we define the interpolation operator $I_h : C(\Gamma) \to PL_\#(\Gamma_h)$ where $PL_\#(\Gamma_h)$ denotes the set of continuous piecewise affine functions on Γ_h. The idea is to interpolate the values of a function $u \in C(\Gamma)$ on each triangle $\mathcal{T}_{hk} \subset \Gamma_h$ from its values known on the three nodes belonging to Γ.
The following theorem proves the existence of the interpolation operator I_h and bounds the error between $u_h \in H^1_\#(\Gamma)$ and its lift on Γ.

Theorem 3.4 (Interpolation lemma)**.**
Let $u \in H^2(\Gamma)$, there exists a unique $I_h(u) \in PL_\#(\Gamma_h)$ such that

$$||u - a_\epsilon(I_h(u_h))||_{L^2(\Gamma)} + h||\nabla_\Gamma(u - a_\epsilon(I_h(u_h)))||_{L^2(\Gamma)} \leq ch^2(||\nabla_\Gamma u||_{L^2(\Gamma)} + h||\nabla_\Gamma u||_{L^2(\Gamma)}). \tag{3.27}$$

Proof: See Lemma 4.3 in [16]

□

In the following section we investigate the condition number of the finite element operator. We give an upper bound for the condition number that is similar to the Euclidean case.

3.3 Estimate of the condition number

The condition number of an invertible matrix A relative to the norm $||\cdot||$ is : $cond(A) = ||A|| \times ||A^{-1}||$. The Euclidean norm is quite often used in applications and the expression of the L^2-condition number for a symmetric matrix is

$$cond_2(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}, \tag{3.28}$$

where $\lambda_{max}(A)$ (resp. $\lambda_{min}(A)$) is the largest (resp. smallest) eigenvalue of A in absolute value.

The condition number plays an important role in the numerical linear algebra as it can be used to predict the convergence of iterative methods [43]. In the case of the finite element matrix for the Euclidean Laplace operator, the condition number has been evaluated and is proportional to h^{-2} [19]. In the case of a closed surface one has to deal firstly with the fact that $A_{\triangle_{\Gamma_h}}$ is non invertible, secondly with the presence of a curvature field.

We consider a piecewise triangular surface Γ_h and the finite element matrix $A_{\triangle_{\Gamma_h}}$ obtained from the discretization of the Laplace-Beltrami operator on Γ_h . A consequence of Theorem 3.1 is that $A_{\triangle_{\Gamma_h}}$ is an isomorphism of the space

$$\mathcal{V}_0 = \left\{ (u_1, \ldots, u_n) \in \mathbb{R}^d, s.t. \sum_{i=1}^{n} u_i \int_{\Gamma_h} \phi_i = 0 \right\}. \tag{3.29}$$

Here is our new result giving the upper bound of condition number of finite element operator. It is obtained by adapting to the curved surfaces the steps followed in the Euclidean case.

In the sequel, denoted by $\mathcal{T}$ a triangle of Γ_h, $|\mathcal{T}|$ its area and $h_{\mathcal{T}} = diam(\mathcal{T}) = \max_{x,y\in\mathcal{T}} ||x - y||$. To announce the result of the condition number of the finite element matrix (Theorem 3.5) we need the following 5 lemmas.

We start with a lemma regarding the affine geometry of 3D triangles.

Lemma 3.3.
Let $A_0, B_0, C_0, A, B, C \in \mathbb{R}^3$ such that A_0, B_0, C_0 as well as A, B, C are not aligned.
Let $\mathcal{T}_0$ (resp. $\mathcal{T}$) be the triangle formed by A_0, B_0 and C_0 (resp A, B and C). There exist an affine operator

$$\begin{array}{rcl} T_{\mathcal{T}} : \mathcal{T} & \rightarrow & \mathcal{T}_0 \\ x & \rightarrow & J_{\mathcal{T}} x + b_{\mathcal{T}} \end{array}$$

where $J_{\mathcal{T}}$ is a 3×3 matrix and $b_{\mathcal{T}} \in \mathbb{R}^3$.

Proof:
Since A, B, C are not aligned, they define a plane $(\mathcal{P})$ with unit normal $\overrightarrow{n}$ and A, B, C is an affine frame of reference of $(\mathcal{P})$.
Since A_0, B_0, C_0 are not aligned, they define a plane $(\mathcal{P}_0)$ with unit normal $\overrightarrow{n}_0$ and A_0, B_0, C_0 is an

affine frame of reference of $(\mathcal{P}_0)$.
Since $(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{n})$ and $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n}_0)$ form a basis of $\mathbb{R}^3$, $J_\mathcal{T}$ is defined as the transition matrix from a basis $(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{n})$ to a basis $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n}_0)$. We have

$$\begin{aligned} J_\mathcal{T}\overrightarrow{AB} &= \overrightarrow{A_0B_0} \\ J_\mathcal{T}\overrightarrow{AC} &= \overrightarrow{A_0C_0} \\ J_\mathcal{T}\overrightarrow{n} &= \overrightarrow{n}_0 \end{aligned}$$

defining

$$b_\mathcal{T} = A_0 - J_\mathcal{T}A,$$

the affine transformation $T_\mathcal{T}(x) = J_\mathcal{T}x + b_\mathcal{T}$ satisfies

$$\begin{aligned} T_\mathcal{T}(A) &= A_0 \\ T_\mathcal{T}(B) &= B_0 \\ T_\mathcal{T}(C) &= C_0 \end{aligned}$$

Let $P \in \mathcal{T}$, since $(\overrightarrow{AB}, \overrightarrow{AC})$ is a basis of $(\mathcal{P})$ there are coefficients $\alpha, \beta \in \mathbb{R}$ such that

$$P = A + \alpha\overrightarrow{AB} + \beta\overrightarrow{AC}$$

hence

$$\begin{aligned} T_\mathcal{T}(P) &= T_\mathcal{T}(A) + \alpha J_\mathcal{T}\overrightarrow{AB} + \beta J_\mathcal{T}\overrightarrow{AC} \\ &= A_0 + \alpha\overrightarrow{A_0B_0} + \beta\overrightarrow{A_0C_0} \\ &= P_0 \in \mathcal{T}_0. \end{aligned}$$

Hence $T_\mathcal{T} : \mathcal{T} \to \mathcal{T}_0$, and the lemma is proved.

□

Lemma 3.4.
Under the hypotheses of Lemma 3.3 and letting $M_\mathcal{T} = [\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{n}]$ the transition matrix from the canonical basis to the basis $(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{n})$ and $M_{\mathcal{T}_0} = [\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n_0}]$ the transition matrix from the canonical basis to the basis $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n}_0)$, $J_\mathcal{T}$ is of the form

$$J_\mathcal{T} = M_{\mathcal{T}_0}M_\mathcal{T}^{-1}$$

and furthermore

$$det(J_\mathcal{T}) = \frac{|\mathcal{T}_0|}{|\mathcal{T}|}.$$

Proof:
Since $J_\mathcal{T}$ is transition matrix from basis $(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{n})$ to basis $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n_0})$, we have

$$J_\mathcal{T}M_\mathcal{T} = M_{\mathcal{T}_0}$$

since $M_{\mathcal{T}}$ is invertible, multiplying the previous relation by $M_{\mathcal{T}}^{-1}$, we obtain $J_{\mathcal{T}} = M_{\mathcal{T}_0} M_{\mathcal{T}}^{-1}$. Taking the determinant yields $det(J_{\mathcal{T}}) = det(M_{\mathcal{T}_0}) \times det(M_{\mathcal{T}}^{-1})$ then

$$\begin{aligned} det(J_{\mathcal{T}}) &= \frac{det(M_{\mathcal{T}_0})}{det(M_{\mathcal{T}}^{-1})} \\ &= \frac{|\mathcal{T}_0|}{|\mathcal{T}|} \end{aligned}$$

since

$$\begin{aligned} \det(M_{\mathcal{T}}) = det(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{n}) = (\overrightarrow{AB} \wedge \overrightarrow{AC}).\overrightarrow{n} = |\mathcal{T}| \\ \det(M_{\mathcal{T}_0}) = \det(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n_0}) = (\overrightarrow{A_0B_0} \wedge \overrightarrow{A_0C_0}).\overrightarrow{n_0} = |\mathcal{T}_0| \end{aligned}$$

□

Let $\mathcal{T}_0$ be a reference triangle in $\mathbb{R}^3$ with three non aligned vertices x_1, x_2, x_3. ($\mathcal{T}_0$ is independent from the surface Γ_h).

Lemma 3.5.
We write $\phi_i^0, i \in \{1, 2, \cdots, n\}$ the shape functions associated to each node of $\mathcal{T}_0$ and define the function

$$\begin{aligned} f_0 : \mathbb{R}^n &\rightarrow \mathbb{R} \\ (u_1, u_2, \cdots, u_n) &\rightarrow \int_{\mathcal{T}_0} |\sum_{i=1}^{n} u_i \phi_i^0|^2 \end{aligned}$$

f_0 is a strictly positive quadratic form on $\mathbb{R}^n$.
Furthermore, there exist $C > c > 0$, such that

$$\forall (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \quad c \sum_{i=1}^{n} u_i^2 \leq \int_{\mathcal{T}_0} |\sum_{i=1}^{n} u_i \phi_i^0|^2 \leq C \sum_{i=1}^{n} u_i^2.$$

Proof:
f_0 is a positive quadratic form since

$$\begin{aligned} \forall U = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \ |\sum_{i=1}^{n} u_i \phi_i^0|^2 &\geq 0. \\ \forall U = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \ \int_{\mathcal{T}_0} |\sum_{i=1}^{n} u_i \phi_i^0|^2 &\geq 0 \\ \text{Finally}, \quad \forall U = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \ f_0(U) &\geq 0 \ . \end{aligned}$$

Let show that $ker f_0 = \{0_{\mathbb{R}^n}\}$, where $ker f_0$ is the kernel of f_0. First, we remark that

$$\begin{aligned}
\forall U = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, U \in ker f_0 &\iff f_0(U) = 0 \\
&\iff \int_{\mathcal{T}_0} |\sum_{i=1}^{n} u_i \phi_i^0|^2 = 0 \\
&\iff |\sum_{i=1}^{n} u_i \phi_i^0|^2 = 0 \\
&\iff \sum_{i=1}^{n} u_i \phi_i^0 = 0 \\
&\iff \forall x \in \mathcal{T}_0, \sum_{i=1}^{n} u_i \phi_i^0(x) = 0
\end{aligned}$$

since $\phi_i^0(x)$ are basis functions, we deduce that

$$f_0(U) = 0 \Longleftrightarrow u_i = 0 \quad \forall i \in \{1, 2, \cdots, n\}$$

That is f_0 is a strictly positive quadratic form.

Secondly, we seek $c > 0$ and $C > 0$ such that

$$\forall U = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, c \sum_{i=1}^{n} u_i^2 \leq f_0(U) \leq C \sum_{i=1}^{n} u_i^2$$

Since f_0 is a strictly positive quadratic form on $\mathbb{R}^n$, its matrix M is symmetric and positive definite. M thus admits a minimum eigenvalue $\lambda_{\min} > 0$ and a maximum eigenvalue $\lambda_{\max} > 0$.

We have $\forall U \in \mathbb{R}^n$, $f_0(U) = {}^tUMU$

Since M is symmetric and positive defined, we deduce

$$\forall U = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \quad \lambda_{\min} \sum_{i=1}^{n} u_i^2 \leq f_0(U) \leq \lambda_{\max} \sum_{i=1}^{n} u_i^2$$

Taking $c = \lambda_{\min} > 0$ and $C = \lambda_{\max} > 0$ we obtain the desired result.

□

In the following two lemmas (Lemma 3.6 and Lemma 3.7) the lower bound and the upper bound of the following bilinear form a_h

$$a_h(\tilde{u}_h, v) = \int_{\Gamma_h} \nabla_{\Gamma_h} \tilde{u}_h \cdot \nabla_{\Gamma_h} v. \tag{3.30}$$

on $PL_{\#}(\Gamma_h) \times PL_{\#}(\Gamma_h)$ are given, where $\tilde{u}_h$ still the discrete solution of our discrete variational problem.

Lemma 3.6 (Lower bound of a_h)**.**
We have for the solution $\tilde{u}_h$

$$\tilde{u}_h = \sum_{i=1}^{n} u_i \phi_i \in PL_{\#}(\Gamma_h), \quad a_h(\tilde{u}_h, \tilde{u}_h) \geq m(\Gamma_h) c n_c \frac{\min_{\mathcal{T} \in \mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_{i=1}^{n} u_i^2$$

Where $m(\Gamma_h)$ *is a Poincaré's constant,* $n(i)$ *is the number of cells surrounding the node* i *and* $n_c = \min_i n(i)$

Proof:
Using Poincaré's inequality we have

$$a_h(\tilde{u}_h, \tilde{u}_h) \geq m(\Gamma_h) \int_{\Gamma_h} |\tilde{u}_h|^2.$$

We then decompose a_h as a sum over the mesh cells $\mathcal{T}$.

$$a_h(\tilde{u}_h, \tilde{u}_h) \quad \geq \quad m(\Gamma_h) \sum_{\mathcal{T} \in \mathcal{T}_h} \int_{\mathcal{T}} |\tilde{u}_h|^2.$$

Let $N_{\mathcal{T}}$ denotes the set of nodes surrounding the cell $\mathcal{T}$. Since $\phi_i(x) = 0$ when $x \in \mathcal{T}$ and $i \notin N_{\mathcal{T}}$, we have

$$a_h(\tilde{u}_h, \tilde{u}_h) \quad \geq \quad m(\Gamma_h) \sum_{\mathcal{T} \in \mathcal{T}_h} \int_{\mathcal{T}} |\sum_{i \in N_{\mathcal{T}}} u_i \phi_i|^2.$$

Using a change of variable we obtain

$$a_h(\tilde{u}_h, \tilde{u}_h) \quad \geq \quad m(\Gamma_h) \sum_{\mathcal{T} \in \mathcal{T}_h} |\det(J_{\mathcal{T}})|^{-2} \int_{\mathcal{T}_0} |\sum_{i \in N_{\mathcal{T}}} u_i \phi^0_{T_{\mathcal{T}}(i)}|^2,$$

where $T_{\mathcal{T}}(i)$ represents the node of $\mathcal{T}_0$ that is the image of the node i by $T_{\mathcal{T}}$. From Lemma 3.5 we have

$$a_h(\tilde{u}_h, \tilde{u}_h) \quad \geq \quad m(\Gamma_h) c \sum_{\mathcal{T} \in \mathcal{T}_h} |\det(J_{\mathcal{T}})|^{-2} \sum_{i \in N_{\mathcal{T}}} u_i^2.$$

and Lemma 3.4 leads to

$$\begin{aligned}
a_h(\tilde{u}_h, \tilde{u}_h) \quad &\geq \quad m(\Gamma_h) c \sum_{\mathcal{T} \in \mathcal{T}_h} \frac{|\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_{i \in N_{\mathcal{T}}} u_i^2 \\
&\geq \quad m(\Gamma_h) c \frac{\min_{\mathcal{T} \in \mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_{\mathcal{T} \in \mathcal{T}_h} \sum_{i \in N_{\mathcal{T}}} u_i^2 \\
&\geq \quad m(\Gamma_h) c \frac{\min_{\mathcal{T} \in \mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_i n(i) u_i^2 \\
&\geq \quad m(\Gamma_h) c n_c \frac{\min_{\mathcal{T} \in \mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_i u_i^2,
\end{aligned}$$

Hence we finally have

$$a_h(\tilde{u}_h, \tilde{u}_h) \quad \geq \quad m(\Gamma_h) c n_c \frac{\min_{\mathcal{T} \in \mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_i u_i^2, \tag{3.31}$$

where $n(i)$ is the number of cells surrounding the node i and

$$n_c = \min_i n(i).$$

□

Lemma 3.7 (Upper bound of a_h)**.**
Under the hypotheses of Lemma 3.6, we have

$$\forall \tilde{u}_h = \sum_{i=1}^{n} u_i \phi_i \in PL_{\#}(\Gamma_h), \quad a_h(\tilde{u}_h, \tilde{u}_h) \leq n n_c \frac{h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2} \sum_i u_i^2.$$

where $h_{\mathcal{T}} = diam(\mathcal{T}) = \max_{x,y\in\mathcal{T}} ||x-y||$

Proof:
We decompose a_h as a sum over the mesh cells $\mathcal{T}_h$.

$$\begin{aligned}
\forall \tilde{u}_h \in PL_{\#}(\Gamma_h), a_h(\tilde{u}_h, \tilde{u}_h) &= \sum_{\mathcal{T}\in\mathcal{T}_h} \int_{\mathcal{T}} |\nabla_{\Gamma_h} \tilde{u}_h|^2 && (3.32)\\
&= \sum_{\mathcal{T}\in\mathcal{T}_h} \int_{\mathcal{T}} \left| \sum_{i\in N_{\mathcal{T}}} u_i \nabla_{\Gamma_h} \phi_i(\overrightarrow{x}) \right|^2 && (3.33)\\
&\leq 3 \sum_{\mathcal{T}\in\mathcal{T}_h} \sum_{i\in N_{\mathcal{T}}} u_i^2 \int_{\mathcal{T}} |\nabla_{\Gamma_h} \phi_i(\overrightarrow{x})|^2. && (3.34)
\end{aligned}$$

The next step consists in the explicit calculation of $\nabla_{\Gamma_h}\phi_i(\overrightarrow{x})$ in each triangle $\mathcal{T}$. For more detail we refer to Section 3.2.5.

Given a triangle $\mathcal{T}$ having nodes $s_1^{\mathcal{T}}$, $s_2^{\mathcal{T}}$ and $s_3^{\mathcal{T}}$, the gradient of the nodal function associated to $s_1^{\mathcal{T}}$, is

$$\forall \overrightarrow{x} \in \mathcal{T}, \quad \nabla_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\overrightarrow{x}) = \frac{1}{-2|\mathcal{T}|} \begin{pmatrix} a_1 \\ \\ b_1 \\ c_1 \end{pmatrix},$$

where

$$\begin{aligned}
a_1 &= (y_{s_3^{\mathcal{T}}} - y_{s_2^{\mathcal{T}}}) n_z^{\mathcal{T}} + (z_{s_2^{\mathcal{T}}} - z_{s_3^{\mathcal{T}}}) n_y^{\mathcal{T}} \\
b_1 &= (x_{s_2^{\mathcal{T}}} - x_{s_3^{\mathcal{T}}}) n_z^{\mathcal{T}} + (z_{s_3^{\mathcal{T}}} - z_{s_2^{\mathcal{T}}}) n_x^{\mathcal{T}} \\
c_1 &= (y_{s_2^{\mathcal{T}}} - y_{s_3^{\mathcal{T}}}) n_x^{\mathcal{T}} + (x_{s_3^{\mathcal{T}}} - x_{s_2^{\mathcal{T}}}) n_y^{\mathcal{T}}.
\end{aligned}$$

We then have the following, where $||.||$ denotes de Euclidean norm

$$||\nabla_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\overrightarrow{x})|| = \frac{1}{2|\mathcal{T}|}\sqrt{a_1^2 + b_1^2 + c_1^2},$$

and using the inequality $\forall r, s \in \mathbb{R}, (r+s)^2 \leq 2(r^2+s^2)$, we have

$$\begin{aligned}
||\nabla_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\overrightarrow{x})|| &\leq \frac{1}{2|\mathcal{T}|}\sqrt{a_1^2+b_1^2+c_1^2} \\
&\leq \frac{1}{2|\mathcal{T}|}\sqrt{(h_{\mathcal{T}}n_z^{\mathcal{T}}+h_{\mathcal{T}}n_y^{\mathcal{T}})^2+(h_{\mathcal{T}}n_x^{\mathcal{T}}+h_{\mathcal{T}}n_y^{\mathcal{T}})^2+(h_{\mathcal{T}}n_z^{\mathcal{T}}+h_{\mathcal{T}}n_x^{\mathcal{T}})^2} \\
&\leq \frac{1}{2|\mathcal{T}|}\sqrt{2((h_{\mathcal{T}}n_z^{\mathcal{T}})^2+(h_{\mathcal{T}}n_y^{\mathcal{T}})^2)+2((h_{\mathcal{T}}n_x^{\mathcal{T}})^2+(h_{\mathcal{T}}n_y^{\mathcal{T}})^2)+2((h_{\mathcal{T}}n_z^{\mathcal{T}})^2+(h_{\mathcal{T}}n_x^{\mathcal{T}})^2)} \\
&\leq \frac{1}{2|\mathcal{T}|}\sqrt{4((h_{\mathcal{T}}n_z^{\mathcal{T}})^2+(h_{\mathcal{T}}n_y^{\mathcal{T}})^2+(h_{\mathcal{T}}n_x^{\mathcal{T}})^2)} \\
&\leq \frac{1}{2|\mathcal{T}|}2h_{\mathcal{T}}\sqrt{(n_z^{\mathcal{T}})^2+(n_y^{\mathcal{T}})^2+(n_x^{\mathcal{T}})^2} \\
&\leq \frac{1}{2|\mathcal{T}|}2h_{\mathcal{T}}.
\end{aligned}$$

Hence

$$||\nabla_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\overrightarrow{x})|| \leq \frac{h_{\mathcal{T}}}{|\mathcal{T}|}.$$

Using the same approach we obtain an estimate of the gradient of the two remaining nodal functions

$$||\nabla_{\Gamma_h}\phi_{s_2^{\mathcal{T}}}(\overrightarrow{x})|| \leq \frac{h_{\mathcal{T}}}{|\mathcal{T}|}, \quad ||\nabla_{\Gamma_h}\phi_{s_3^{\mathcal{T}}}(\overrightarrow{x})|| \leq \tfrac{h_{\mathcal{T}}}{|\mathcal{T}|}.$$

Finally, for any shape function ϕ_i, we have

$$\forall i \in \{s_1^{\mathcal{T}}, s_2^{\mathcal{T}}, s_3^{\mathcal{T}}\}, ||\nabla_{\Gamma_h}\phi_i^h(\overrightarrow{x})|| \leq \frac{h_{\mathcal{T}}}{|\mathcal{T}|}$$

This allows us to deduce the final result from (3.34) :

$$a_h(\tilde{u}_h, \tilde{u}_h) \leq 3n_c(\sum_i u_i^2)\frac{\max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h}|\mathcal{T}|^2}. \tag{3.35}$$

□

Theorem 3.5 (Condition number of the finite element matrix)**.**
There exists a constant c such that for any closed piecewise triangular surface $\Gamma_h \subset \mathbb{R}^3$, the condition number of $A_{\triangle_{\Gamma_h}}$ (Equation (3.18)) satisfies

$$\mathcal{K}_h(A_{\triangle_{\Gamma_h}}) \leq \frac{n\max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{m(\Gamma_h)c\min_{\mathcal{T}\in\mathcal{T}_h}|\mathcal{T}|^2}$$

where $m(\Gamma_h)$ is related to the Poincaré's constant $c(\Gamma)$ by $m(\Gamma) = c(\Gamma)^{-1}$

Proof:
From Lemma 3.7, we have

$$\tilde{u}_h \in PL_{\#}(\Gamma_h), \frac{a_h(\tilde{u}_h, \tilde{u}_h)}{\sum_{i=1}^n u_i^2} \leq nn_c\frac{\max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h}|\mathcal{T}|^2}$$

and we deduce an upper bound of the spectrum of the finite element operator on $PL_{\#}(\Gamma_h)$.

$$\lambda_{\max} \leq 3n_c \frac{\max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2} \tag{3.36}$$

Lemma 3.6 leads to

$$\forall \tilde{u}_h \in PL_{\#}(\Gamma_h), \frac{a_h(\tilde{u}_h, \tilde{u}_h)}{\sum_{i=1}^n u_i^2} \geq m(\Gamma_h) c n_c \frac{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2}$$

and we deduce a lower bound of the spectrum of the finite element operator on $PL_{\#}(\Gamma_h)$.

$$\lambda_{\min} \geq m(\Gamma_h) c n_c \frac{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \tag{3.37}$$

From definition of

$$\mathcal{K}_h(A_{\triangle_{\Gamma_h}}) = \frac{\lambda_{\max}}{\lambda_{\min}},$$

combined with (3.37) and (3.36) yield the following inequalities

$$\begin{aligned} \forall \mathcal{T}_0, \mathcal{K}_h(A_{\triangle_{\Gamma_h}}) &\leq \frac{|\mathcal{T}_0|^2}{m(\Gamma_h) c n_c \min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2} \times \frac{n n_c \max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2} \\ &\leq \frac{|\mathcal{T}_0|^2}{m(\Gamma_h) c \min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2} \times \frac{n \max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2} \end{aligned}$$

Choosing $\mathcal{T}_0$ such that $|\mathcal{T}_0|^2 = \max_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2$, we finally have

$$\mathcal{K}_h(A_{\triangle_{\Gamma_h}}) \leq \frac{n \max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{m(\Gamma_h) c \min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}.$$

□

3.4 Conclusion

We have proven that the properties of the finite element methods on 3D surfaces are very similar to those in Euclidean spaces. Simulations are easy to perform. The present work can be extended for general manifolds of dimension $d > 2$.

Using similar techniques, it is possible to derive a lower bound of the condition number.

Since we have to solve a singular linear system , we didn't used a Gauss elimination method but an iterative method such as Conjugate Gradient or GMRES methods. However in order to improve the convergence of the iterative method and to reduce its computational time it is important to use a preconditioner. Classical preconditioners are based on incomplete Gauss elimination. For singular matrices the incomplete Gauss elimination is not possible. We therefore need to derive a new preconditioner.

⋆ ⋆ CHAPTER FOUR ⋆ ⋆

NUMERICAL RESULTS

Contents

4.1 Introduction

To validate our methods, we present two numerical examples. The first example is on the unit sphere and the second is on the torus; both examples are taken from [37]. For the numerical resolution of our discrete problem, we use an iterative solver because the stiffness matrix $A_{\triangle_{\Gamma_h}}$ is large and sparse (See [43]), it is the use of hollow storage (not dense) and an iterative solver that allows to use large meshes and therefore to consider **industrial applications**, since direct solvers saturate memory beyond 100,000 unknowns. Moreover in our case the matrix being singular, the use of a direct solver is very dangerous (division by zero). These calculations were performed with the iterative solver Gradient Conjugate (because the matrix is symmetric) and the preconditioner ICC (incomplete Cholesky) and a requested precision of 1e-4. We had to decrease the precision so that the calculations do not last too long, also because classical preconditioners are not effective on singular matrices. In our case, the size of the largest mesh is 431242 nodes (for the **Sphere**) and 481024 nodes (for the **Torus**), but this would not be possible with a dense matrix or a direct solver. The results were obtained sequentially on a simple laptop with 8GB of RAM. Computer speed could be improved on a parallel computer.

4.2 Finite element simulation of the Poisson problem for the Laplace-Beltrami operator on the 3D Sphere

4.2.1 Position of the Problem

We consider the unit sphere defined as follows

$$S = \{(x, y, z) \in \mathbb{R}^3, x^2 + y^2 + z^2 = 1\}.$$

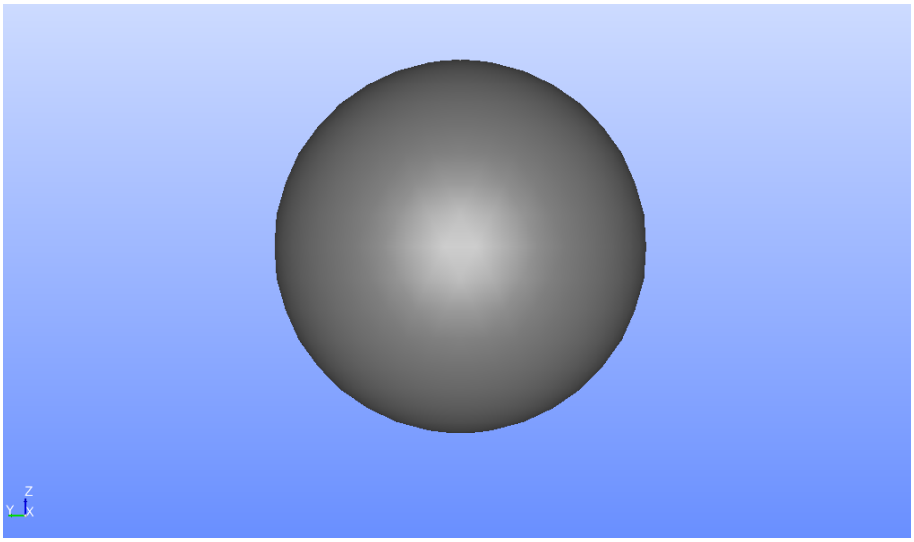


Figure 4.1: The unit sphere in SALOME CAO module

The sphere is a C^∞ manifold of dimension 2 embedded in $\mathbb{R}^3$. The **Laplace-Beltrami operator** $\triangle_\Gamma$, a generalisation of the Euclidean Laplacian, can be defined as the combination of a surface divergence $\nabla_\Gamma \cdot$, and of a surface gradient ∇_Γ (See Chapter 2) The points on the unit sphere can be

parametrised using the angles $\theta \in [0, \pi], \phi \in [0, 2\pi]$

$$\begin{cases} x = \sin\theta\cos\phi \\ y = \sin\theta\sin\phi \\ z = \cos\theta \end{cases} \tag{4.1}$$

We consider the following **Poisson problem** on the sphere S

$$\begin{cases} -\Delta_S u = f \text{ on } \Gamma \\ \int_S u = 0 \end{cases}, \tag{4.2}$$

where $f \in L^2(S)$ with zero mean, and u is the unknown function.

For the following choice of f :

$$f(x, y, z) = \frac{12}{(x^2 + y^2 + z^2)^{\frac{3}{2}}}(3x^2y - y^3), \tag{4.3}$$

the exact solution u of (4.2) is given by (see [37]):

$$u(x, y, z) = \frac{1}{(x^2 + y^2 + z^2)^{\frac{3}{2}}}(3x^2y - y^3) = \frac{1}{12}f.$$

Our objective is to solve numerically the Poisson problem (4.2) using the finite element method described in [3, 15, 16, 19].

Using the representation of u in spherical coordinates one can verify that u is an eigenfunction of $-\Delta_S$ and that the right hand side f satisfies the compatibility condition ($\int_\Gamma f = 0$).

Indeed, The Laplacian on the unit sphere is given by

$$\begin{aligned} \Delta_S &= \frac{1}{(\sin\theta)}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{(\sin\theta)^2}\frac{\partial^2}{\partial\phi^2} \\ &= \frac{\cos\theta}{\sin\theta}\frac{\partial}{\partial\theta} + \frac{\partial^2}{\partial\theta^2} + \frac{1}{(\sin\theta)^2}\frac{\partial^2}{\partial\phi^2} \\ &= \Delta_1 + \Delta_2 + \Delta_3 \end{aligned}$$

Let us find required derivatives,

$$\begin{aligned} \frac{\partial u}{\partial\theta} &= \frac{\partial(\sin(3\phi)\sin^3(\theta))}{\partial\theta} = 3\sin(3\phi)\sin^2(\theta)\cos(\theta) \\ \frac{\partial^2 u}{\partial\theta^2} &= \frac{\partial(3\sin(3\phi)\sin^2(\theta)\cos(\theta))}{\partial\theta} \\ &= 3\sin(3\phi)\left(-\sin^3(\theta) + 2\sin\theta\cos^2(\theta)\right) \\ &= 3\sin(3\phi)\left(-3\sin^3(\theta) + 2\sin\theta\right) \\ \frac{\partial u}{\partial\phi} &= \frac{\partial(\sin(3\phi)\sin^3(\theta))}{\partial\phi} = 3\cos(3\phi)\sin^3(\theta) \end{aligned}$$

and

$$\frac{\partial^2 u}{\partial\phi^2} = \frac{\partial(3\sin(3\phi)\sin^3(\theta))}{\partial\phi} = -9\sin(3\phi)\sin^3(\theta)$$

which leads to

$$\begin{aligned}
\Delta_1 u &= \frac{\cos\theta}{\sin\theta}(3\sin(3\phi)\sin^2(\theta)\cos(\theta)) = 3\sin(3\phi)\sin(\theta)\cos^2(\theta) \\
\Delta_2 u &= -9\sin(3\phi)\sin^3(\theta) + 6\sin(3\phi)\sin(\theta) \\
\Delta_3 u &= \frac{1}{(\sin\theta)^2}\frac{\partial^2}{\partial\phi^2} = -9\sin(3\phi)\sin(\theta)
\end{aligned}$$

We then have

$$\begin{aligned}
\Delta_S u &= \Delta_1 u + \Delta_2 u + \Delta_3 u \\
&= 3\sin(3\phi)\sin(\theta)\left(1-\sin^2(\theta)\right) - 9\sin(3\phi)\sin^3(\theta) - 3\sin(3\phi)\sin(\theta) \\
&= -12\sin^3\theta\sin(3\phi) \\
&= -12u
\end{aligned}$$

Thus $-\Delta_S u = 12u = f$ and u is the eigenvector of the Laplacian.

It is a spherical harmonics well recognized by mathematicians and physicists. The simplest example of using spherical harmonics in quantum mechanics is for example the description of the positions of electrons in the hydrogen atom. The angular motions of the electron in the hydrogen atom are described by a spherical harmonics.

Let us show that f is zero mean on the sphere.

$$\begin{aligned}
\int_\Gamma f ds = 12\int_0^{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin(3\phi)\sin^3(\theta)r^2\sin(\theta)d\theta d\phi &= 12r^2\int_0^{2\pi}\sin^4(\theta)d\theta\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\sin(3\phi)d\phi \\
&= 12r^2\int_0^{2\pi}\sin^4(\theta)d\theta\left[-\frac{1}{3}\cos(3\phi)\right]_{-\frac{\pi}{2}}^{\frac{\pi}{2}} = 0
\end{aligned}$$

Hence, f is zero mean on S.

For coding the finite element method, we use the Python language and the open-source Linux based library CDMATH [54] which is very simple for the manipulation of large matrices, vectors, meshes and fields. It (CDMATH) can handle finite element and finite volume discretization, read general $3D$ geometries and meshes generated by SALOME.

4.2.2 Meshing of the domain

For the design and meshing of the domain we use GEOMETRY and MESH modules of the software SALOME 9.5 (See [40, 41, 52]).

Below are the meshes used in our convergence analysis.

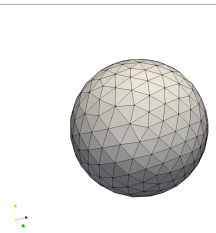
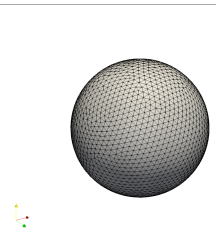
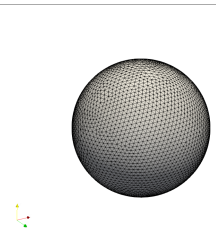
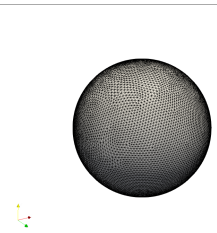
meshSphere 1	meshSphere 2	meshSphere 3	meshSphere 4
			
288 cells	2638 cells	4512 cells	10773 cells

Figure 4.2: Mesh of domain

4.2.3 Visualization of the results

For the visualization of the result, we use the PARAVIS module included in SALOME (See [52, 59]).

Below are visualizations of the numerical results obtained on the different meshes of Figure 4.2.

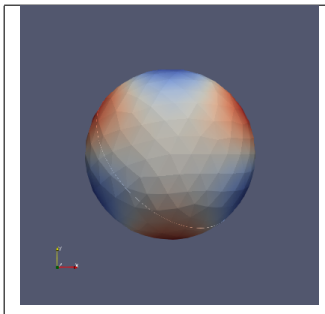
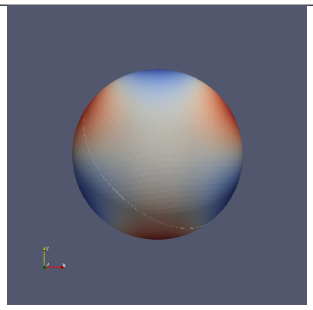
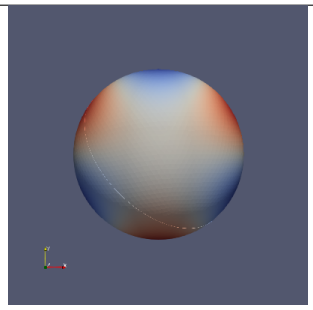
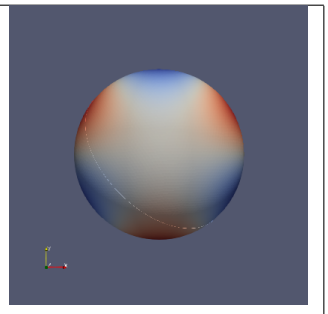
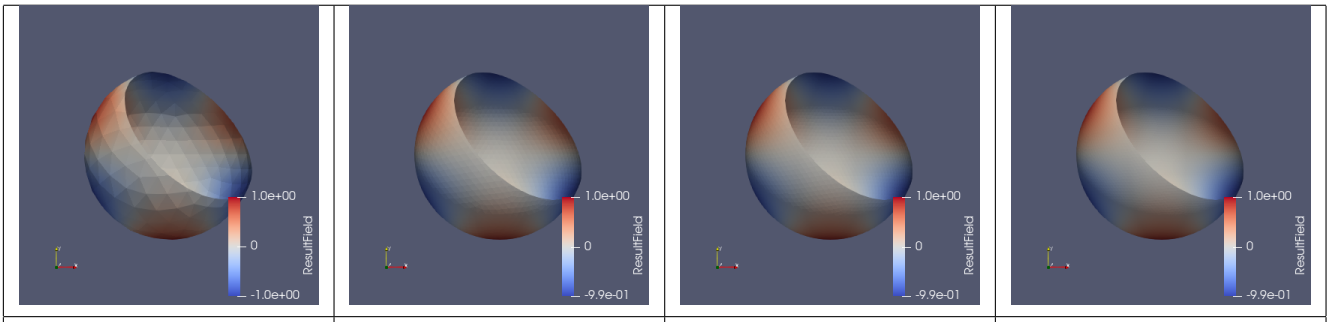
			
288 cells	2638 cells	4512 cells	10773 cells

Figure 4.3: Numerical results of the finite elements on the unit sphere with source f given by (4.3)

Below are clipings of the previous numerical results.



288 cells	2638 cells	4512 cells	10773 cells

Figure 4.4: Cliping of the numerical result on the unit sphere with source f given by (4.3)

4.2.4 Numerical convergence of the finite element method

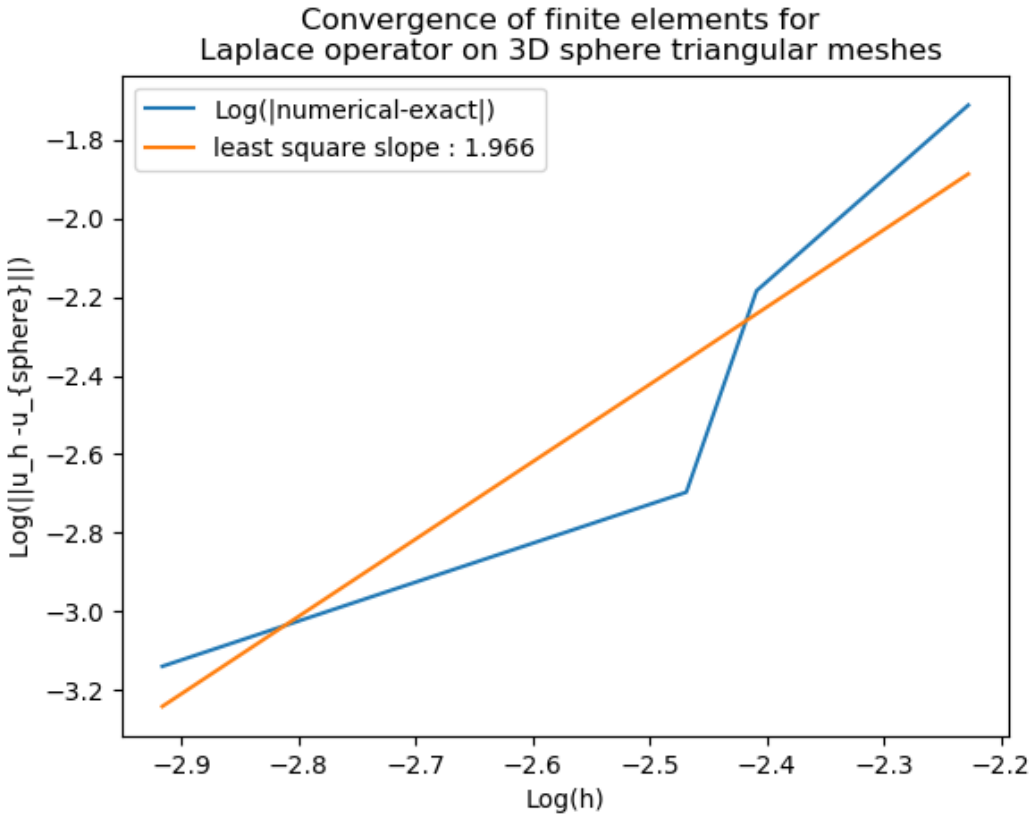


Figure 4.5: Convergence of the finite element method on the sphere

The method converges with a numerical order of approximately 1.966.

4.2.5 Computational time of the finite element method

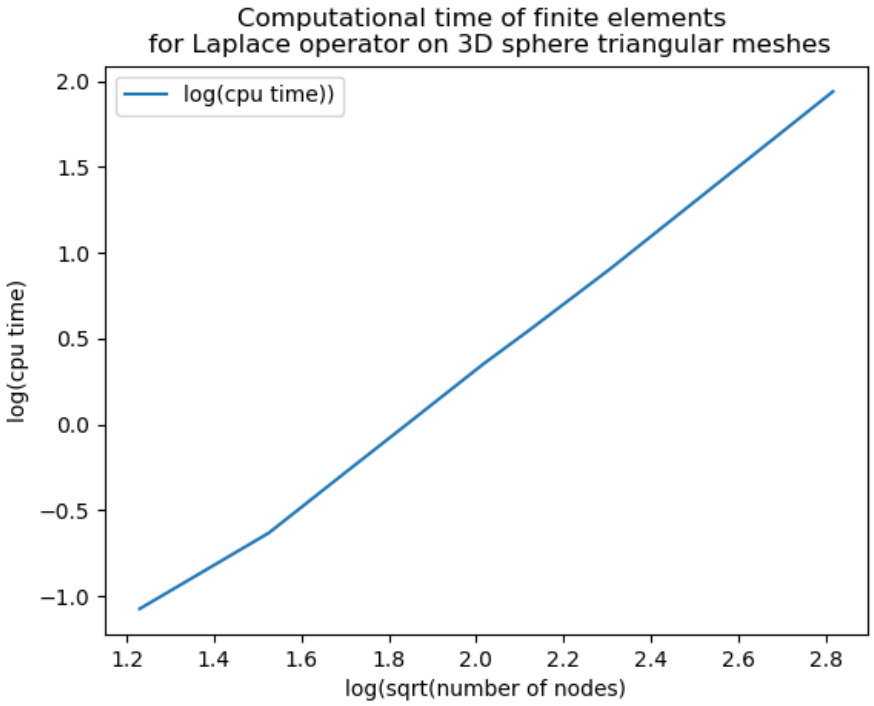


Figure 4.6: Computational time of the finite element method on the sphere

4.2.6 Ploting over slice circle

Here we have drawn a circle on each sphere to extract the values. This circle is visible on the spheres in Figure 4.17.

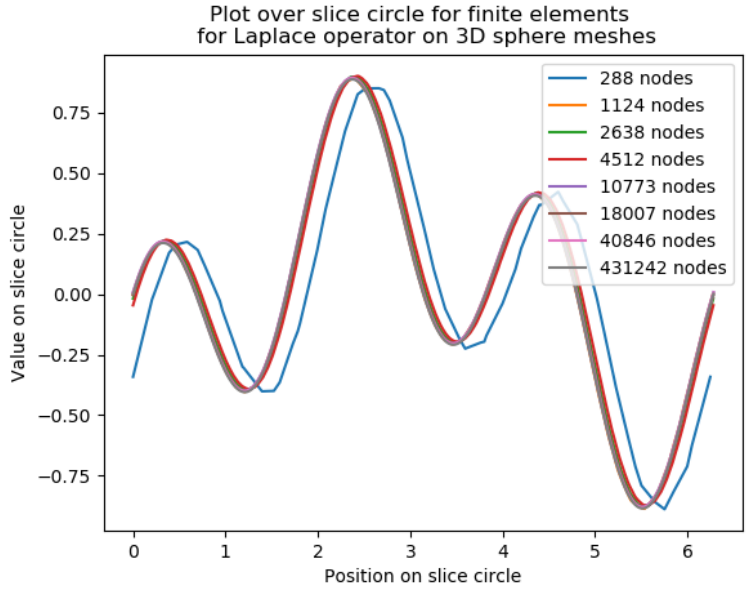


Figure 4.7: Convergence of the data ploted over a circle drawn on the sphere

4.3 Finite element simulation of the Poisson problem for the Laplace-Beltrami operator on the 3D Torus

4.3.1 Position of the Problem

We consider the torus defined as follow

$$\mathbb{T} = \{(x, y, z) \in \mathbb{R}^3, (\sqrt{x^2 + y^2} - R)^2 + z^2 - r^2 = 0\},$$

where $r > 0$ is the minor radius and $R > r$ is the major radius.

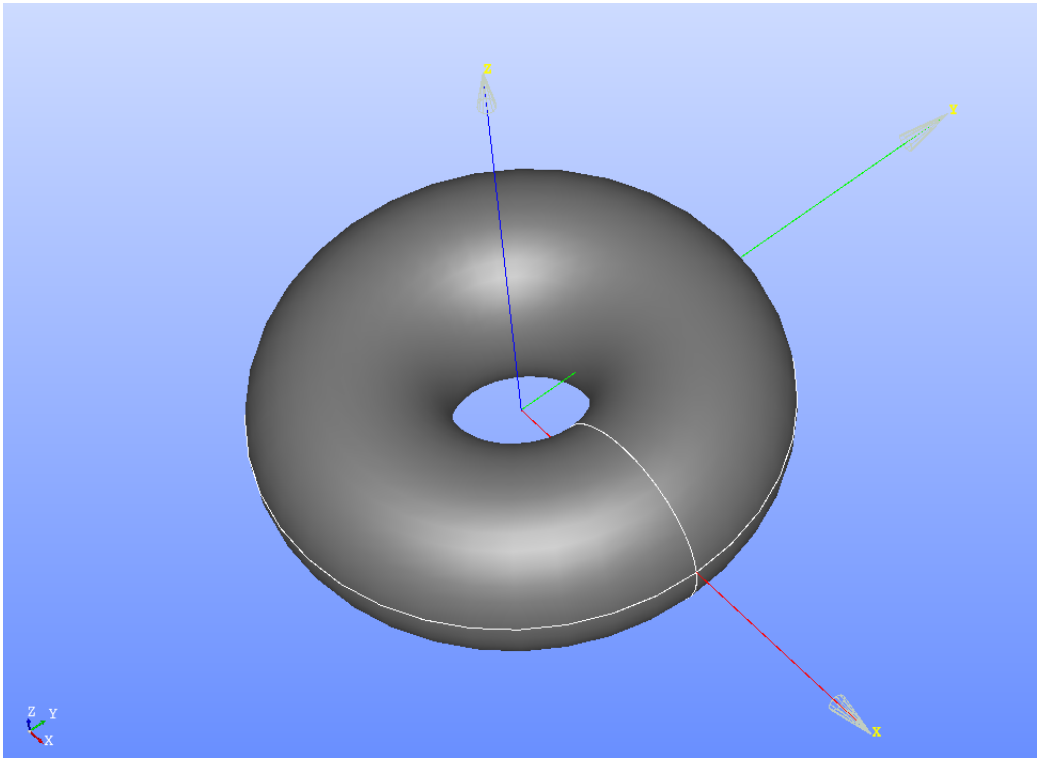


Figure 4.8: The torus in SALOME CAO module

The torus is a C^∞ manifold of dimension 2 embedded in $\mathbb{R}^3$. The **Laplace-Beltrami operator** $\triangle_{\mathbb{T}}$, a generalisation of the Euclidean Laplacian, can be defined on the torus as the combination of a surface divergence $\nabla_{\mathbb{T}}\cdot$, and of a surface gradient $\nabla_{\mathbb{T}}$ (See for instance [16, 38, 50] and Chapter 2).

To compute our exact solution of (4.6), we use the fact that the points of $\mathbb{T}$ can be parametrised using the angles $\theta, \phi \in [0, 2\pi]$ and the following relations

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} \cos\phi \\ \sin\phi \\ 0 \end{pmatrix} + r \begin{pmatrix} \cos\phi\cos\theta \\ \sin\phi\cos\theta \\ \sin\theta \end{pmatrix} \tag{4.4}$$

Using the torus coordinates (θ, ϕ) (4.4) where ϕ is an azimuthal angle on xy-plan, θ is the altitude angle at z-axis, $\theta, \phi \in [0, 2\pi]$, the Laplace-Beltrami operator takes the following form on the torus

$$\Delta_{\mathbb{T}} = \frac{1}{r^2(R + r\cos\theta)}\frac{\partial}{\partial\theta}\left((R + r\cos\theta)\frac{\partial}{\partial\theta}\right) + \frac{1}{(R + r\cos\theta)^2}\frac{\partial^2}{\partial\phi^2}. \tag{4.5}$$

We consider the following **Poisson problem** on the torus

$$\begin{cases} -\Delta_{\mathbb{T}} u = f \text{ on } \mathbb{T} \\ \int_{\mathbb{T}} u = 0 \end{cases}, \tag{4.6}$$

where $f \in L^2(\mathbb{T})$ with zero mean, and u the unknown function.

For the following choice of $u(\theta, \phi) = \sin(3\phi)\cos(3\theta + \phi)$ (see [37]), the right hand side is given by

$$\begin{aligned} f &= 9r^{-2}\sin(3\phi)\cos(3\theta + \phi) - (10\sin(3\phi)\cos(3\theta + \phi) && (4.7)\\ &\quad +6\cos(3\phi)\sin(3\theta + \phi))((R + rcos(\theta))^{-2} && (4.8)\\ &\quad -3\sin(\theta)\sin(3\phi)\sin(3\theta + \phi)(r(R + r\cos(\theta)))^{-1}. && (4.9) \end{aligned}$$

Our objective is to solve numerically the Poisson problem (4.6) using the finite element method described in [3, 15, 16, 19].

Let show that u is the exact solution of $-\Delta_{\mathbb{T}} = f$.

The Laplacian 4.5 can be rewritten as follows:

$$\begin{aligned} (4.5) \equiv \Delta_{\mathbb{T}} &= \frac{\sin\theta}{r(R + r\cos\theta)}\frac{\partial}{\partial\theta} + \frac{1}{r^2}\frac{\partial^2}{\partial\theta^2} + \frac{1}{(R + r\cos\theta)^2}\frac{\partial^2}{\partial\phi^2} \\ &= \Delta_1 + \Delta_2 + \Delta_3 \end{aligned}$$

$$\frac{\partial u}{\partial\theta} = \frac{\partial}{\partial\theta}(\sin(3\phi)\cos(3\theta + \phi)) = -3\sin(3\phi)\sin(3\theta + \phi)$$

$$\frac{\partial^2 u}{\partial\theta^2} = \frac{\partial}{\partial\theta}\left(\frac{\partial u}{\partial\theta}\right) = -9\sin(3\phi)\cos(3\theta + \phi)$$

$$\frac{\partial u}{\partial\phi} = 3\cos(3\phi)\cos(3\theta + \phi) - \sin(3\phi)\sin(3\theta + \phi)$$

$$\begin{aligned} \frac{\partial^2 u}{\partial\phi^2} = \frac{\partial}{\partial\phi}\left(\frac{\partial u}{\partial\phi}\right) &= -9\sin(3\phi)\cos(3\theta + \phi) - 3\cos(3\phi)\sin(3\theta + \phi) \\ &- 3\cos(3\phi)\sin(3\theta + \phi) - \sin(3\phi)\cos(3\theta + \phi) \\ &= -10\sin(3\phi)\cos(3\theta + \phi) - 6\cos(3\phi)\sin(3\theta + \phi) \end{aligned}$$

$$\begin{aligned} \Delta_1 u = \frac{\sin\theta}{r(R + r\cos\theta)}\frac{\partial u}{\partial\theta} &= \frac{\sin\theta}{r(R + r\cos\theta)}(-3\sin(3\phi)\sin(3\theta + \phi) && (4.10)\\ &= \frac{-3\sin\theta\sin(3\phi)\sin(3\theta + \phi)}{r(R + r\cos\theta)} && (4.11) \end{aligned}$$

$$\Delta_2 u = \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = -9r^{-2}\sin(3\phi)\cos(3\theta+\phi) \tag{4.12}$$

$$\begin{aligned} \Delta_3 u &= \frac{1}{(R+r\cos\theta)^2}\frac{\partial^2 u}{\partial \phi^2} && (4.13)\\ &= (R+r\cos\theta)^{-2}\left(-10\sin(3\phi)\cos(3\theta+\phi) - 6\cos(3\phi)\sin(3\theta+\phi)\right) && (4.14)\end{aligned}$$

After combining (4.10), (4.12) and (4.13), we obtain

$$\begin{aligned} \Delta_{\mathbb{T}} u &= \Delta_1 u + \Delta_2 u + \Delta_3 u && (4.15)\\ &= -3(r(R+r\cos\theta))^{-1}\sin\theta\sin(3\phi)\sin(3\theta+\phi) - 9r^{-2}\sin(3\phi)\cos(3\theta+\phi) && (4.16)\\ &+ (R+r\cos\theta)^{-2}\left(-10\sin(3\phi)\cos(3\theta+\phi) - 6\cos(3\phi)\sin(3\theta+\phi)\right) && (4.17)\end{aligned}$$

Hence, the source function $f = -\Delta_{\Pi} u$ and then u is the exact solution of (4.6)

For coding the finite element method, we use the Python language and the open-source Linux based library CDMATH [54] which is very simple for the manipulation of large matrices, vectors, meshes and fields. It (CDMATH) can handle finite element and finite volume discretizations, read general $3D$ geometries and meshes generated by SALOME.

4.3.2 Meshing of the domain

For the design and meshing of the domain we use GEOMETRY and MESH modules of the software SALOME 9.5 (See [40, 41, 52]).

Below are the meshes used in our convergence analysis.

meshTorus 1	meshTorus 2	meshTorus 3	meshTorus 4
			
1022 cells	6461 cells	20006 cells	43910 cells

Figure 4.9: Mesh of domain

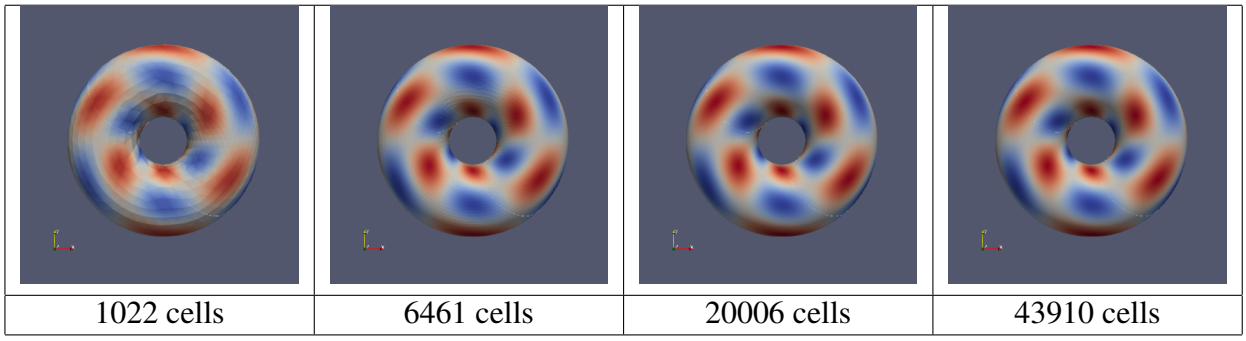
4.3.3 Visualization of the results

For the numerical resolution of our discrete problem, we use an iterative solver because the stiffness matrix $A_{\triangle_{\Gamma_h}}$ is large and sparse (See [43]) .

For the visualization of the result, we use the PARAVIS module included in SALOME (See [52, 59]).

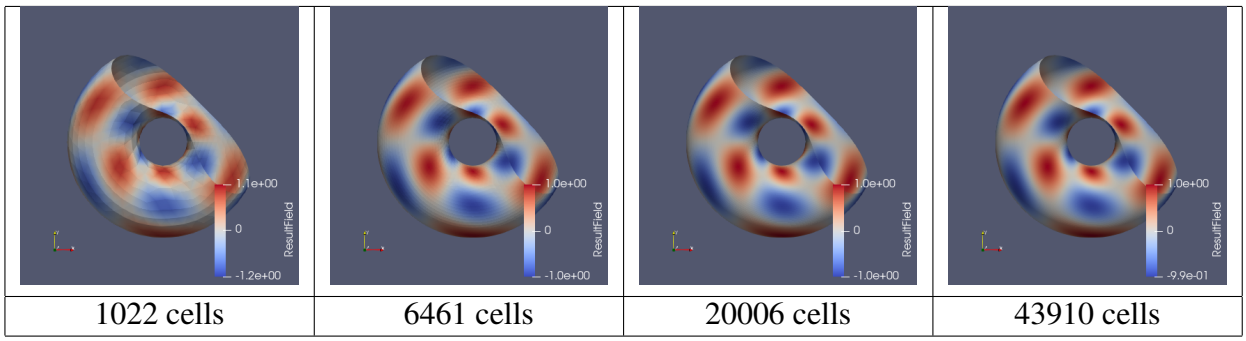
Below are visualizations of the numerical results obtained on the different meshes of Figure 4.9.

Below are clipings of the previous numerical results.



1022 cells	6461 cells	20006 cells	43910 cells

Figure 4.10: Numerical results of the finite elements on the torus with source f given by (4.7)



1022 cells	6461 cells	20006 cells	43910 cells

Figure 4.11: Cliping of the numerical result on the torus with source f given by (4.7)

4.3.4 Numerical convergence of the finite element method

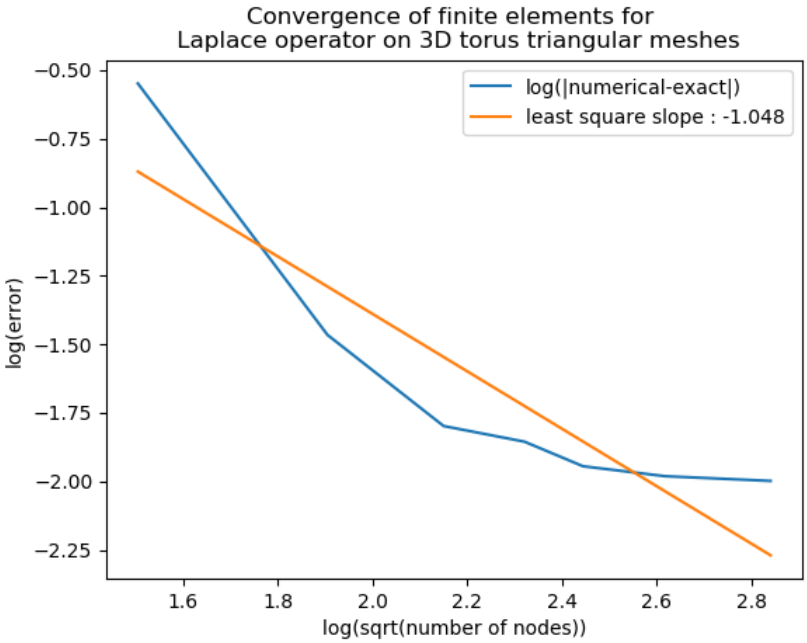


Figure 4.12: Convergence of the finite element method on the torus

The method converges with a numerical order of approximately 1.04.

4.3.5 Computational time of the finite element method

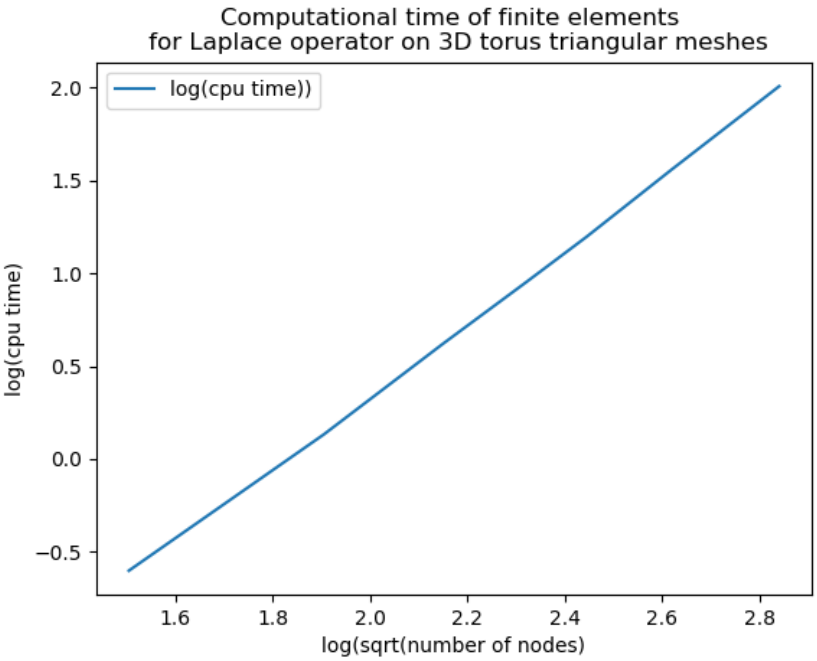


Figure 4.13: Computational time of the finite element method on the torus

4.3.6 Ploting over slice circle

Here we have drawn a circle on each torus to extract the values. This circle is visible on the torus in Figure 4.10.

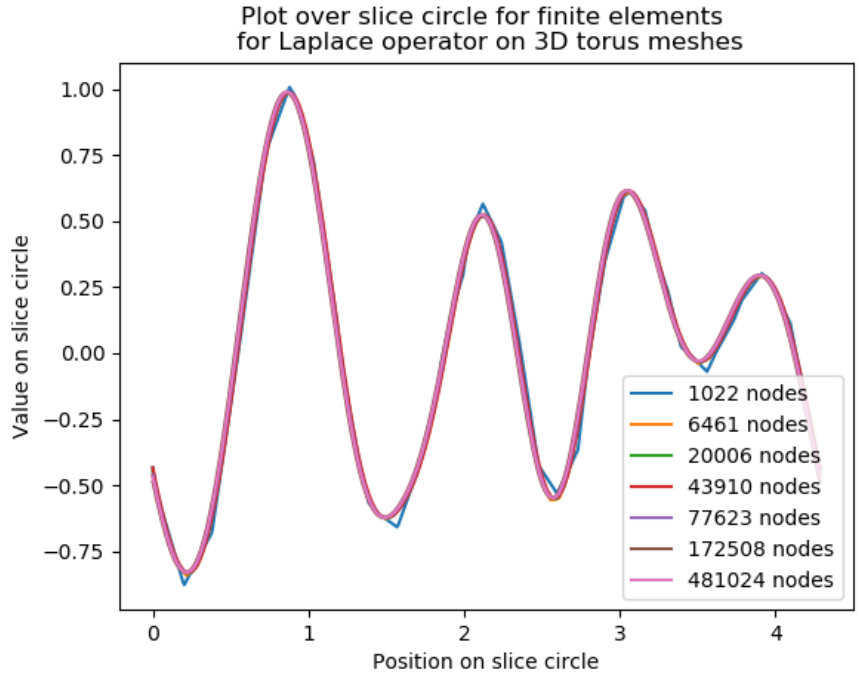


Figure 4.14: Convergence of the data ploted over a circle drawn on the torus

4.4 Finite element simulation of the Poisson problem on a cube skin

4.4.1 Introduction

We consider the unit cube $Y = (0,1) \times (0,1) \times (0,1) \subset \mathbb{R}^3$ and its boundary

$$\Gamma = \partial Y.$$

Γ is a topological manifold but not a differential manifold because of the presence of sharp edges where Γ admits no tangent space. Γ is however a Lipschitz manifold, an intermediate structure between topological and differentiable manifolds [22, 24, 32], which is only differentiable almost everywhere thanks to the Rademacher Theorem (Theorem 2.1). As a Lipschitz manifold, Γ can be endowed with a Laplace-Beltrami operator $\triangle_\Gamma = \nabla \cdot \nabla$, defined as the combination of a surface divergence $\nabla_\Gamma \cdot$, and of a surface gradient ∇_Γ (see [22]).

We consider f the restriction of the smooth function $\cos(2\pi x)\cos(2\pi y)\cos(2\pi z)$ to Γ. The explicit expressions of f on each face of Γ are

$$f(x,y,z) = \begin{cases} \cos(2\pi x)\cos(2\pi y) & \text{if} \quad z = 0 \text{ or } z = 1 \\ \cos(2\pi x)\cos(2\pi z) & \text{if} \quad y = 0 \text{ or } y = 1 \\ \cos(2\pi y)\cos(2\pi z) & \text{if} \quad x = 0 \text{ or } x = 1 \end{cases}. \tag{4.18}$$

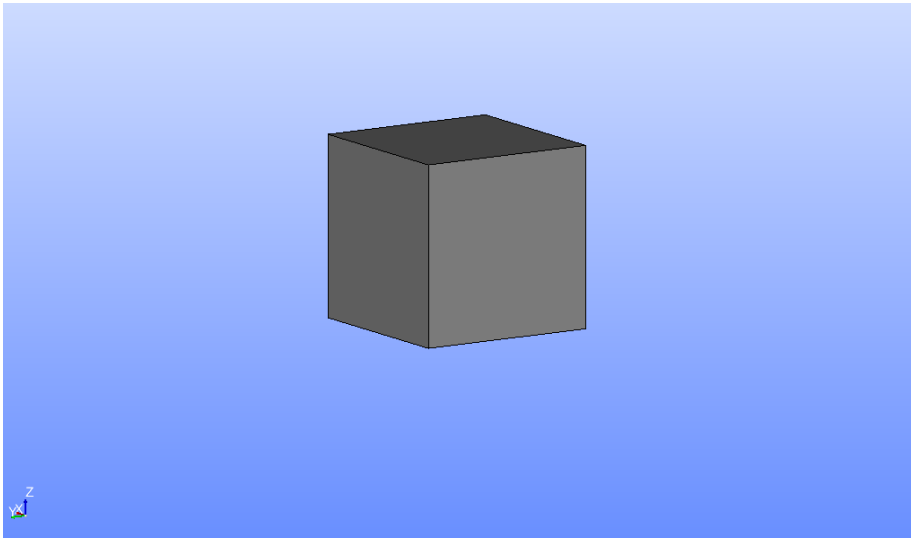


Figure 4.15: The unit cube in SALOME CAO module

f is an eigenfunction of the Laplace-Beltrami operator on Γ since

$$\begin{aligned}
\triangle_\Gamma f(x,y,z) &= \begin{cases} \partial_{xx}\cos(2\pi x)\cos(2\pi y) + \partial_{yy}\cos(2\pi x)\cos(2\pi y) & \text{if} \quad z=0 \text{ or } z=1 \\ \partial_{xx}\cos(2\pi x)\cos(2\pi z) + \partial_{zz}\cos(2\pi x)\cos(2\pi z) & \text{if} \quad y=0 \text{ or } y=1 \\ \partial_{yy}\cos(2\pi y)\cos(2\pi z) + \partial_{zz}\cos(2\pi y)\cos(2\pi z) & \text{if} \quad x=0 \text{ or } x=1 \end{cases} \\
&= \begin{cases} -2(2\pi)^2\cos(2\pi x)\cos(2\pi y) & \text{if} \quad z=0 \text{ or } z=1 \\ -2(2\pi)^2\cos(2\pi x)\cos(2\pi z) & \text{if} \quad y=0 \text{ or } y=1 \\ -2(2\pi)^2\cos(2\pi y)\cos(2\pi z) & \text{if} \quad x=0 \text{ or } x=1 \end{cases} \\
&= -8\pi^2 f.
\end{aligned}$$

We are going to solve the following Poisson problem on Γ :

$$\begin{cases} -\triangle_\Gamma u = f, \\ \int_\Gamma u = 0, \end{cases}$$

where the source $f \in L^2(\Gamma)$ with zero mean, and u is the unknown.

Our objective is to solve numerically the Poisson problem (4.4.1) using the finite element method described in [36].

For the numerical resolution of our discrete problem, we use an iterative solver because the stiffness matrix $A_{\triangle_{\Gamma_h}}$ is large and sparse (See [43]). The library PETSc [58] provides linear solvers for singular systems.

For the design and meshing of the domain we use GEOMETRY and MESH modules of the software SALOME 9.5 (See [41, 52]).

For the visualization of the result, we use the PARAVIS module included in SALOME (See [52, 59]).

For the coding of the script, we use Python with the open-source Linux based library SOLVERLAB [60] which is very practical for the manipulation of large matrices, vectors, meshes and fields. It (SOLVERLAB) can handle finite element and finite volume discretizations, read general $1D, 2D$ and $3D$ geometries and meshes generated by SALOME.

4.4.2 Meshing of the domain

Below are the meshes used in our convergence analysis.

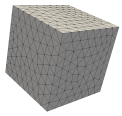
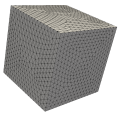
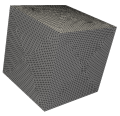
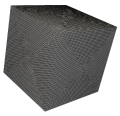
meshCubeSkin 1	meshCubeSkin 2	meshCubeSkin 3	meshCubeSkin 4
			
412 cells	1923 cells	7042 cells	13225 cells

Figure 4.16: Meshes of the unit cube skin

4.4.3 Visualization of the results

Below are visualizations of the numerical results obtained on the different meshes

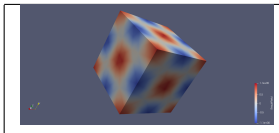
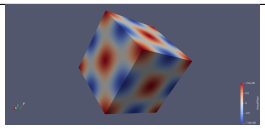
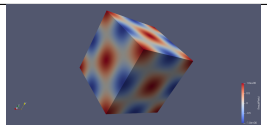
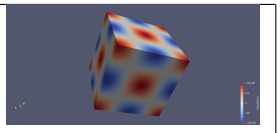
			
412 cells	1923 cells	7042 cells	13225 cells

Figure 4.17: Numerical results of the finite elements on the unit cube skin with source f given by (4.18)

Below are clippings of the previous numerical results.

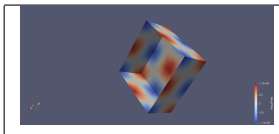
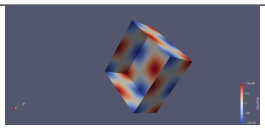
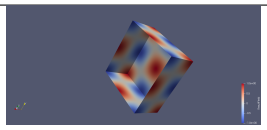
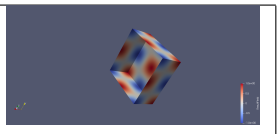
			
412 cells	1923 cells	7042 cells	13225 cells

Figure 4.18: Cliping of the numerical result on the unit cube skin with source f given by (4.18)

4.4.4 Numerical convergence of the finite element method

The method converges with a numerical order of approximately 1.91.

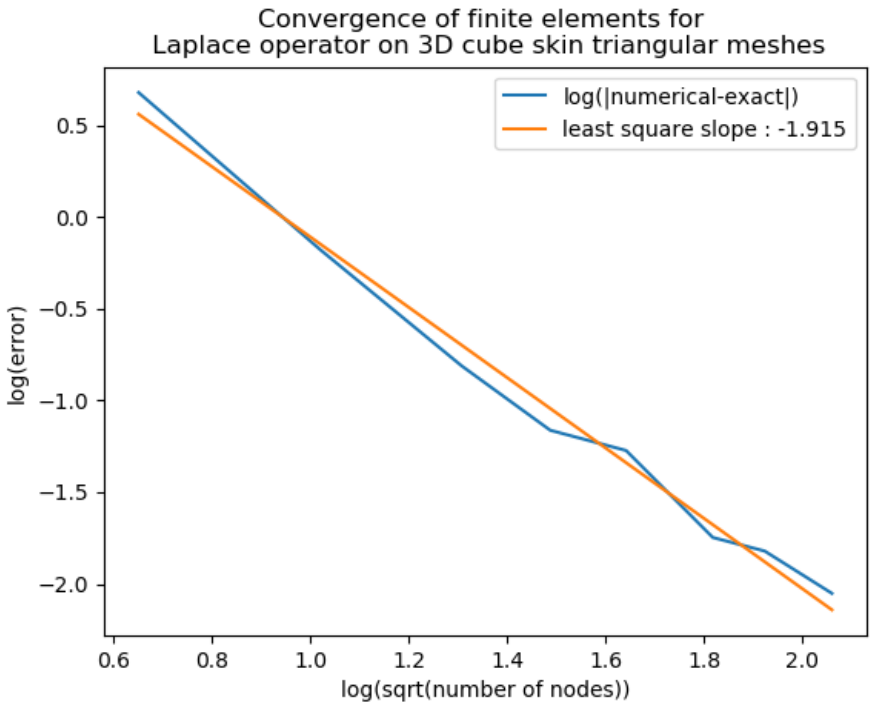


Figure 4.19: Convergence of the finite element method on the cube skin

4.4.5 Computational time of the finite element method

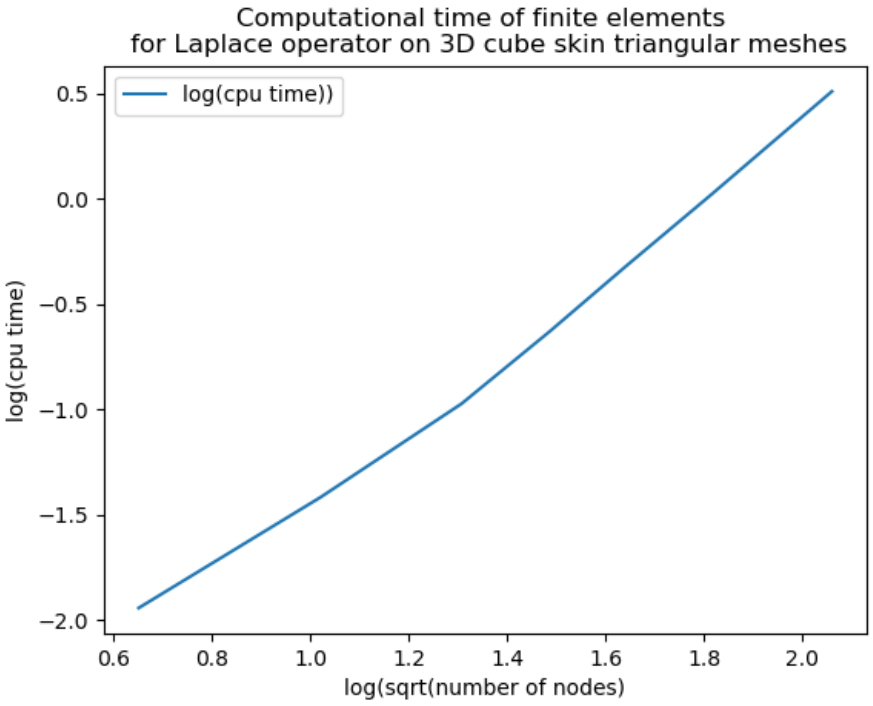


Figure 4.20: Computational time of the finite element method on the cube skin

4.5 Discussion of the numerical results for the Condition Number

4.5.1 Numerical convergence of the finite element method

Figure 4.21 displays the evolution of the numerical error $\|u_h - u_{sphere}\|$ with the cell minimal diameter h, in logarithmic scale. The numerical error $\|u_h - u_{sphere}\|$ is taken as the supremum of $|u_h(x_i) - u_{sphere}(x_i)|$ over all the nodes x_i.

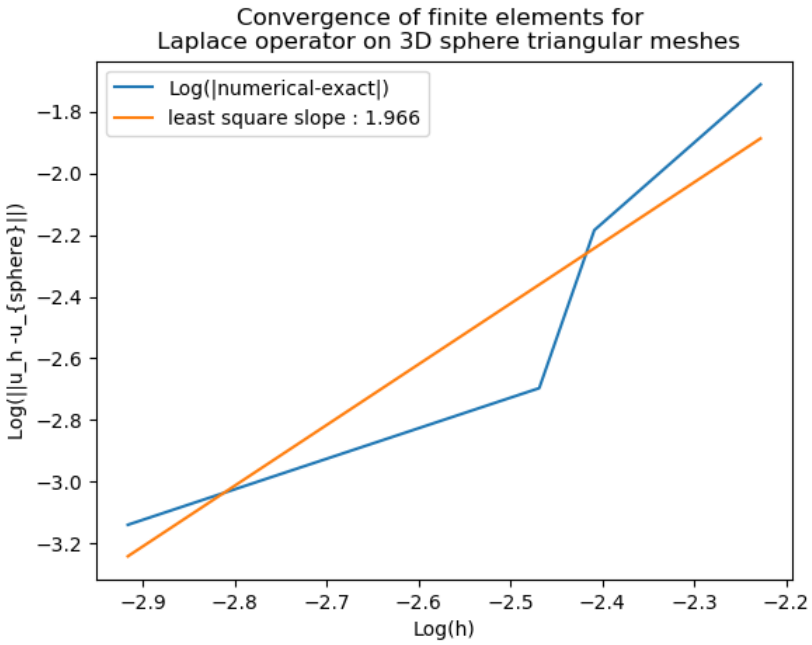


Figure 4.21: Convergence of the finite element method on the sphere

The theoretical error is composed of two contributions. The first is the interpolation error of u_{sphere} from the sphere Γ_{sphere} to a polyhedron Γ_h and is $\mathcal{O}(h^2)$. The second contribution is the approximation error coming from the finite element discretisation of the Poisson equation. This error is $\mathcal{O}(h^2)$. The total error is therefore $\mathcal{O}(h^2)$. See for example [13, Proposition 2.3] and [15, Theorem 8] for details concerning the errors.

We observe on Figure 4.21 that the method converges with a numerical order of approximately 1.966, which is very closed to the theoretical value of 2.

4.5.2 Number of CG iterations for the finite element method on the sphere

It is not possible to use a direct solver for the numerical resolution of the linear system $A_{\triangle\Gamma_h} X = b_h$, since they apply only to invertible matrices. We used instead the Conjugate Gradient (CG) [55] method with Incomplete LU factorisation [56] as preconditioner to solve our singular linear system thanks to PETSc algorithm [57, 58]. Figure 4.22 displays the evolution of the number of CG iterations with the number of nodes. The number of iterations increases linearly with the number of nodes of the mesh.

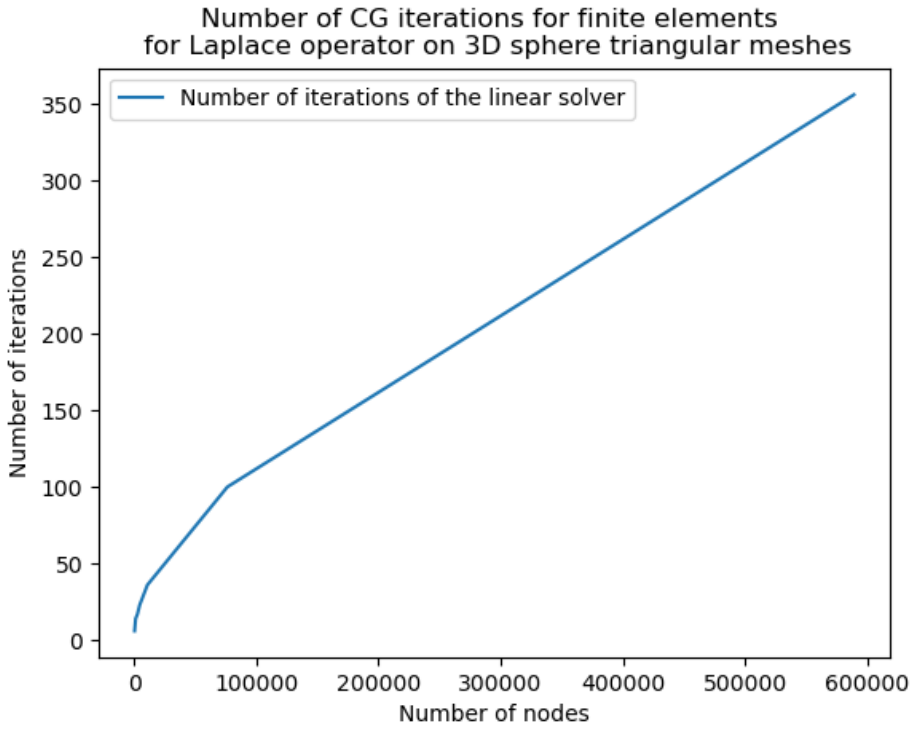


Figure 4.22: Number of CG iterations for the finite element method on the sphere

4.5.3 CG Residual for the finite elements methods on the sphere

Figure 4.23 displays the evolution in Logarithmic scale of the residual $\epsilon_h = \|A_{\triangle\Gamma_h} X - b_h\|$ with the number of nodes in the mesh $\mathcal{T}_h$. This residual ϵ_h evolves from about 10^{-2} to about 10^{-5} as the number of nodes evolves from 288 to 600000. This is because we had to adapt the precision of the linear solver to the number of nodes. Indeed the matrix of the linear system $A_{\triangle\Gamma_h} X = b_h$ is singular and thus b_h should belong to the range of $A_{\triangle\Gamma_h}$ up to machine precision. The theoretical range of $A_{\triangle\Gamma_h}$ consists in vectors of zero mean, but at the computer level zero becomes machine precision and thus b_h should have mean lower than machine precision. If the machine precision is too small (say 10^{-10}) then the linear solver will fail when the integral of b_h is larger than 10^{-10}.

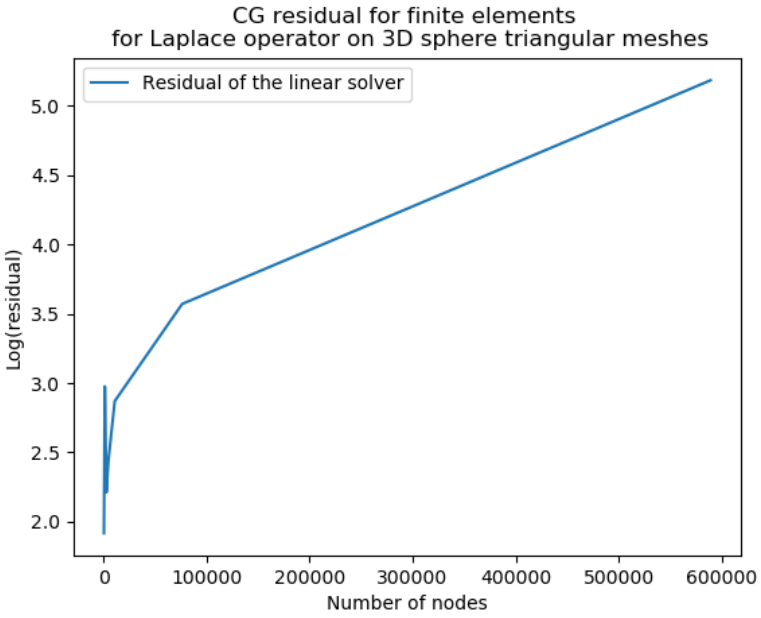


Figure 4.23: CG residual for the finite element method on the sphere

4.5.4 Condition number of the finite element matrix on the sphere

Figure 4.24 displays the evolution of the condition number with the cell minimal diameter h, in logarithmic scale. We observe first that the condition number increase as $h \to \infty$. Furthermore, the

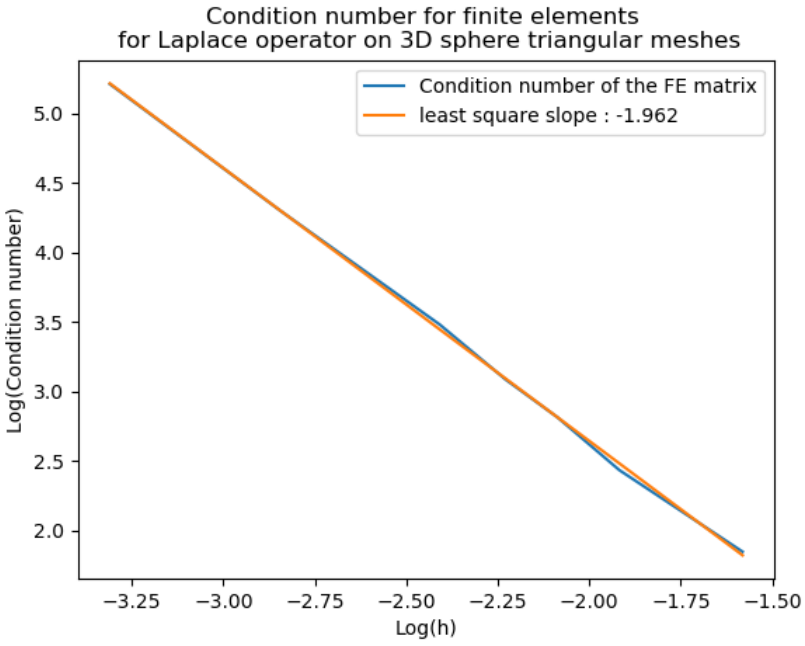


Figure 4.24: Condition number for the finite element method on the sphere

condition number decays as h to the power -1.962, which is very closed to the theoretical value of 2 given in our main result in Theorem 3.5.

♣ Conclusion and future work ♣

During this thesis, we have proven that properties of the finite element methods on 3D surfaces are very similar to that in Euclidean spaces. The finite element matrix is given (the coefficients are explicitly computed) in 3D space. The finite element matrix is not invertible because constant functions are in the kernel. The image of the finite element matrix is the space of functions with zero means[15]. The linear system $A_{\triangle_{\Gamma_h}} X = b_h$ thus admits a solution for b_h with zero means. The solution X has zero means. Simulations are easy to perform. For the coding of the script, we use Python programming Language with the open-source Linux based library SOLVERLAB which is very practical for the manipulation of large matrices, vectors, meshes and field.

Examples of the Poisson problem on the sphere and the torus with analytic solutions has been presented. These solutions were computed numerically for the validation of the numerical method and its coding. The visualization of the result after the computation is done by the Paraview software. We have seen that the properties of the finite element matrix on 3D surfaces are very similar to those in Euclidean spaces. The main difference when the surface is closed is that the matrix is not invertible, which makes the proofs more technical. However, adapting the technique used in the Euclidean context, we gave an upper bound for the condition number in $\mathcal{O}(h^{-2})$. Since we have to solve a singular linear system , we do not use a Gauss elimination method but instead an iterative method such as Conjugate Gradient or GMRES methods. However in order to improve the convergence of the iterative method and to reduce its computational time it is important to use a preconditioner. Classical preconditioners are based on incomplete Gauss elimination. For singular matrices the incomplete Gauss elimination is not possible. We therefore need to derive new preconditioner. In Section 4.5, we have given some numerical results of the simulation of the Poisson problem on a sphere. The results showed that the discretization converges with scheme order of approximately 1.966 and that the condition number increases nearly as $\mathcal{O}(h^{-2})$. Theoretical estimation corroborated by numerical evaluations of condition number from the finite element matrix spectrum, showing that our upper bound is practical. The present work can be extended for general manifolds in dimension n. Using similar techniques, it is possible to derive a lower bound of the condition number.
The numerical simulations performed in section 4.5 showed the number of iterations of the linear solver increases linearly with the mesh size. This yields a sharp increase of the computational time as the number of nodes increases. Our simulation used the Incomplete LU factorisation as preconditioner but more advanced techniques such as multi-grid perform better in the Euclidean context. Their adaptation to the curved would be based on a fine analysis of the finite element matrix taking into account its singularity. This work can therefore be seen as a first step in the design and analysis of advanced preconditioners for singular systems, particularly those arising from the discretisation of PDEs on closed surfaces.

The notion of Lipschitz manifold has been studied by some authors [22, 24] and is useful in laying the foundation of the finite element method on general compact manifolds. We showed

how it allows the proper definition of tangent spaces, gradient and divergence operators almost everywhere. Differentiability almost everywhere thanks to Rademacher's Theorem 2.1 is the key ingredient of the calculus on Lipschitz manifolds.

We have proposed proofs of the Stokes' Theorem 2.3 and Green's Theorem 2.4 as well as a Poincaré's inequality type. This enables the weak formulation of the Poisson problem. The well-posedness of the weak formulation of the Poisson problem is then obtained thanks to the Lax-Milgram's Theorem 1.6.

We gave an example of numerical simulation on a Lipschitz (non smooth) manifold to illustrate the fact that the lack of smoothness does not yield any numerical singularity. The numerical results of the Poisson problem on the unit cube boundary showed that the finite element approximation converges towards the exact solution.

This study has set the theoretical ground for further numerical analysis of more complex PDEs on Lipschitz manifolds. Note however that on Lipschitz manifolds the maximum order of differentiability is one which is enough for the classical weak formulation of second order elliptic PDEs. This is however a technical difficulty for general high order PDEs on Lipschitz manifolds. One workaround for PDEs of order three and more could however be to reset the PDE into a system of low order PDEs.

Yet, with this new approach we were not able to extend the regularity and convergence theorems of [15, 16] because the lack of smoothness of the Lipschitz manifold Γ prohibits the use of the lift operator. Further research should therefore be devoted to the theoretical analysis of the convergence of both the approximate manifold Γ_h and solution u_h towards the exact manifold Γ and solution u.

.

♣ Appendices ♣

A Matrix numerical analysis

This section is dedicated to the numerical analysis of matrix calculation and more precisely to algorithms used to solve linear systems and to calculating the eigenvalues and eigenvectors of a self-adjoint matrix.

A.1 Solution of linear systems

The linear system for the problem which consists of finding the solutions $x \in \mathbb{R}^n$ of the following algebraic equation

$$Ax = b \tag{4.19}$$

where A belongs to the set $\mathbf{M}_n(\mathbb{R})$ of real square matrices of order n, and $b \in \mathbb{R}^n$ is the right-hand side vector.

We shall see two types of methods for the solution of linear system: those called direct, that is to say which allow us to calculate the exact solution in a finite number of operations, and those called **iterative**,that is to say which calculate a sequence of approximate solution which converge to the exact solution.

A.2 Review of matrix norms

We start by recalling the idea of subordinate norm for matrices. We denote by $\mathcal{M}_n(\mathbb{R})$ (respectively $\mathcal{M}_n(\mathbb{C})$ the set of real (respectively complex) square matrices of order n. Even if we consider real matrices, it is necessary, for technical reasons which will be seen in remark 13.1.4, to consider complex matrices.

Definition 4.1. *Let $||.||$ be a vector norm on $\mathbb{C}^n$. We associate with it a matrix norm, called subordinate to this vector norm, defined by*

$$||A|| = \sup_{x \in \mathbb{C}^n, x \neq 0} \frac{||Ax||}{||x||} \tag{4.20}$$

By abuse of language we use the same notation for vector norms and subordinate matrix norms. We easily verify that a subordinate norm defined in this way is a matrix norm over $\mathcal{M}_n(\mathbb{C})$ or $\mathcal{M}_n(\mathbb{R})$.

A.3 Matrix condition number

Before describing the algorithms for the solution of linear systems, we must consider the problems of precision and stability due to rounding errors. Indeed, in a computer there are no exact calculations, and the precision is limited because of the number of bits used to represent real numbers: usually 32 or 64 bits (which makes about 8 or 16 significant figures). We must therefore pay great attention to the inevitable rounding errors and to their propagation throughout a calculation. Numerical methods for the solution of linear systems which do not amplify these errors are called stable. In practice, we shall therefore use algorithms which are both efficient and stable. This amplification of errors depends on the matrix considered. To quantify this phenomenon, we introduce the idea of the condition number of a matrix.

Definition 4.2. *Take a subordinate matrix norm that we denote by $||A||$ (see definition 8.4.1). We say the condition number of a matrix $A \in \mathcal{M}_n(\mathbb{C})$, relative to this norm, is the value defined by*

$$cond(A) = ||A||.||A^{-1}|| \tag{4.21}$$

This idea of condition number will allow us to measure the amplification of the errors in the data (right-hand side or matrix) which result.

Remark 4.1. *We shall say that a matrix is well conditioned if its condition number is close to 1 (its minimal value) and that it is ill conditioned if its condition number is large.*

B Important Linux Commands

1. **ls**: Directory listing
2. **cd dir** Change directory to dir
3. **cd** Change to home directory
4. **mv file1 file2** Rename or move file1 to file2,if file2 is an existing directory
5. **mkdir** Creating a directory dir
6. **rm-r dir**: Deleting the directory
7. **rm-f file**: Force to remove the file
8. **rm file**: Deleting the file
9. **pwd** Show current working directory
10. **cp -r dir1 dir2**: Copy dir1 to dir2;create dir2 if not present
11. **cp file1 file2**: Copy the contents of file1 to file2

C CDMATH Library

CDMATH is a CFD toolbox designed for numerical analysts who work on the representation of thermal-hydraulics and who would prefer to focus on high-level computation.

- Compilation, installation and tests sources
  - Cmake
  - CppUnit
- Download binary CDMATH from repositories
  - Ubuntu 14, 16
  - Fedora 20, 21, 22, 23, 24
- Documentations
  - Installation guides
  - Source code documentation with DOXYGEN
- External tools
  - SALOME PLATFORM: to build meshes and visualize data (MED format)
  - PARAVIEW : To analyze and visualize data (VTK format)
- PETSc is a suite of data structures and routines for the scalable(parallel) solution of scientific applications modeled by partial dif-ferential equations.http://www.mcs.anl.gov/petsc

D SALOME Simulation Platform

Salome is a free software that provides a generic platform for pre- and post-processing for numerical simulation. The salome source code and binaries may be downloaded from its official website.

1. Developer(s) Open cascade EDF CEA
2. Initial release: 2001; 18 years ago
3. Stable release: 9.2.0/December 21, 2018;
4. Written in C++; Python
5. Operating system Linux/ Windows/Unix-like
6. Licence GNU Lesser General Public Licence
7. Website www.salome-Platform.org

SALOME 9.2.0 supports the following platforms (64-bits architecture only):

- Linux Debian 8 and 9
- Linux CentOS 7

- Linux Fedora 24 and 26
- Linux Ubuntu LTS 16.04 and 18.04

The Main modules included in salome are the following:

- Kernel: distributed components management, study management, general services.
- GUI: framework implementing general graphical user interface's services.
- Geometry: create, modify, import/export, repair, measure CAD models, ...
- Mesh: generate a mesh from CAD model using different meshing algorithms or import mesh data from external files; modify mesh data; check quality of meshes, ...
- Med: management of mesh data in MED format.
- ParaVis: post-processing module based on KITWARE ParaView application.
- YACS: supervision module to create and manage distributed calculations.

D.1 Geometry Module

1. Creation of basic geometrical objects

   Point; Line; Circle; Ellipse; Arc; Curve; Vector; Plane; Working Plane; Local Coordinate System;

2. Creation of 3D primitives:

   Box; Cylinder; Sphere; Torus; Cone;

3. Creation of topological objects:

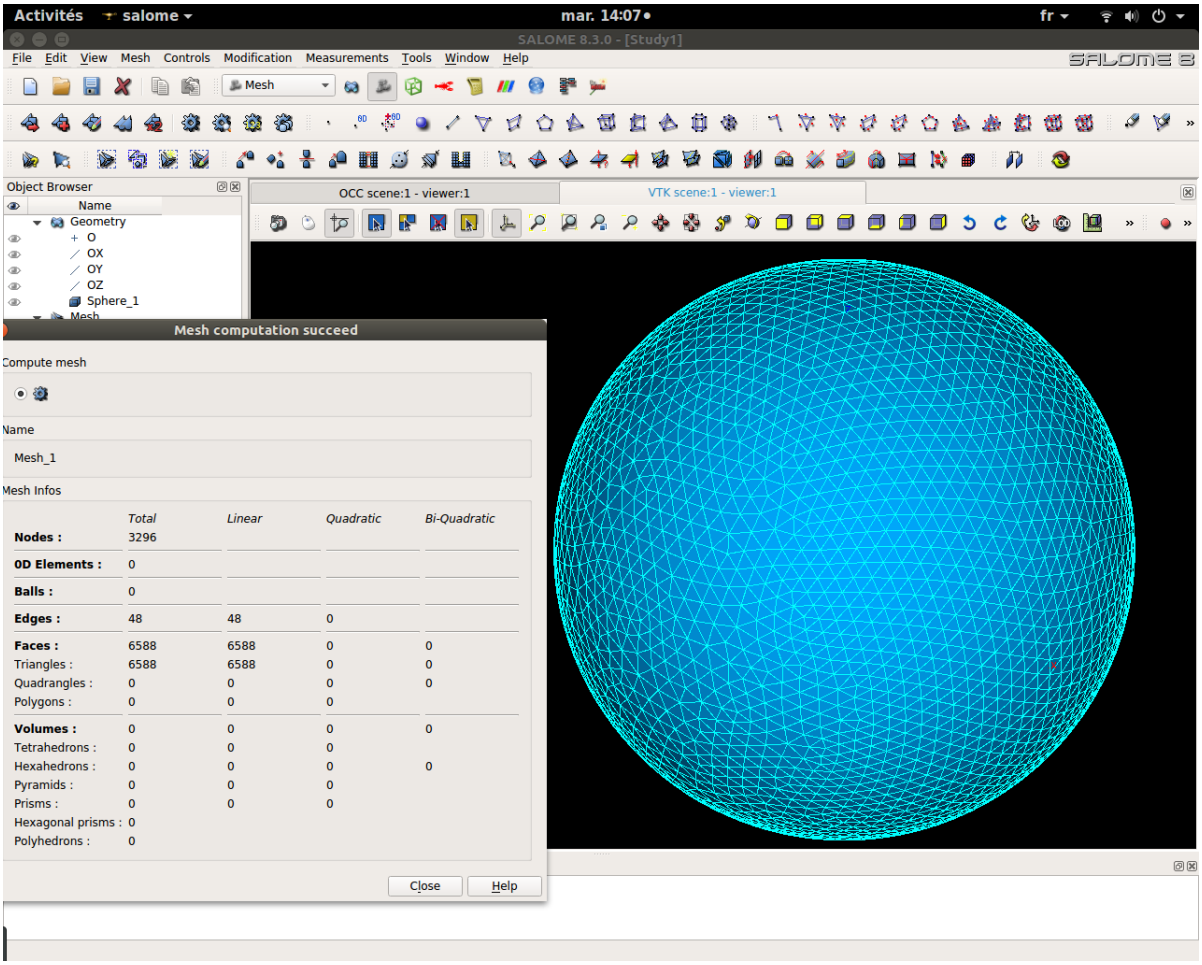
   Vertex; Edge; Wire; Face; Shell; Solid/CompSolid; Compound;

4. Explode topological objects

5. Boolean operations:

   Fuse; Common; Cut; Section;

D.2 Mesh Module(SMESH)



E Python Script

E.1 Python script to solve the Poisson problem on Sphere

```
# -*-coding:utf-8 -*
#
    ================================================================================

# Name        : Résolution EF de l'équation de Laplace-Beltrami -\triangle
    u = f sur une sphere
# Author      : Marcial Nguemfouo and Michael Ndjinga
# Copyright   : CEA Saclay 2017
# Description : Utilisation de la méthode des éléménts finis P1 avec champs
     u et f discrétisés aux noeuds d'un maillage triangulaire
#               Création et sauvegarde du champ résultant ainsi que du
    champ second membre en utilisant la librairie CDMATH
#               Référence : M. A. Olshanskii, A. Reusken, and J. Grande. A
    finite element method for elliptic equations
#                           on surfaces. SIAM J. Num. Anal., 47, p. 3355
#               Solution exacte = f/12 : il s'agit d'un vecteur propre du
    laplacien sur la sphère
#               Résolution d'un système linéaire à matrice singulière : les
     vecteurs constants sont dans le noyau
```

```
#
    ================================================================================


import cdmath
from math import pow
import numpy as np
import PV_routines
import VTK_routines
import paraview.simple as pvs

#Chargement du maillage triangulaire de la sphère
#
    ================================================================================

my_mesh = cdmath.Mesh("meshSphere.med")
if(not my_mesh.isTriangular()) :
  raise ValueError("Wrong cell types : mesh is not made of triangles")
if(my_mesh.getMeshDimension()!=2) :
  raise ValueError("Wrong mesh dimension : expected a surface of dimension
    2")
if(my_mesh.getSpaceDimension()!=3) :
  raise ValueError("Wrong space dimension : expected a space of dimension 3
    ")

nbNodes = my_mesh.getNumberOfNodes()
nbCells = my_mesh.getNumberOfCells()

print("Mesh building/loading done")
print("nb of nodes=", nbNodes)
print("nb of cells=", nbCells)

#Discrétisation du second membre et détermination des noeuds intérieurs
#=====================================================================
my_RHSfield = cdmath.Field("RHS field", cdmath.NODES, my_mesh, 1)
maxNbNeighbours = 0#This is to determine the number of non zero
    coefficients in the sparse finite element rigidity matrix

#parcours des noeuds pour discrétisation du second membre et extraction du
    nb max voisins d'un noeud
for i in range(nbNodes):
  Ni=my_mesh.getNode(i)
  x = Ni.x()
  y = Ni.y()
  z = Ni.z()

  my_RHSfield[i]=12*y*(3*x*x-y*y)/pow(x*x+y*y+z*z,3/2)#vecteur propre du
    laplacien sur la sphère
  if my_mesh.isBorderNode(i): # Détection des noeuds frontière
    raise ValueError("Mesh should not contain borders")
  else:
    maxNbNeighbours = max(1+Ni.getNumberOfCells(),maxNbNeighbours) #true
    only for planar cells, otherwise use function Ni.getNumberOfEdges()

print("Right hand side discretisation done")
print("Max nb of neighbours=", maxNbNeighbours)
print("Integral of the RHS", my_RHSfield.integral(0))

```

```
# Construction de la matrice de rigidité et du vecteur second membre du système linéaire
#
# ======================================================================================

Rigidite=cdmath.SparseMatrixPetsc(nbNodes,nbNodes,maxNbNeighbours)# warning : third argument is number of non zero coefficients per line
RHS=cdmath.Vector(nbNodes)

# Vecteurs gradient de la fonction de forme associée à chaque noeud d'un triangle
GradShapeFunc0=cdmath.Vector(3)
GradShapeFunc1=cdmath.Vector(3)
GradShapeFunc2=cdmath.Vector(3)

normalFace0=cdmath.Vector(3)
normalFace1=cdmath.Vector(3)

#On parcourt les triangles du domaine
for i in range(nbCells):

  Ci=my_mesh.getCell(i)

  #Contribution à la matrice de rigidité
  nodeId0=Ci.getNodeId(0)
  nodeId1=Ci.getNodeId(1)
  nodeId2=Ci.getNodeId(2)
  N0=my_mesh.getNode(nodeId0)
  N1=my_mesh.getNode(nodeId1)
  N2=my_mesh.getNode(nodeId2)

  #Build normal to cell Ci
  normalFace0[0]=Ci.getNormalVector(0,0)
  normalFace0[1]=Ci.getNormalVector(0,1)
  normalFace0[2]=Ci.getNormalVector(0,2)
  normalFace1[0]=Ci.getNormalVector(1,0)
  normalFace1[1]=Ci.getNormalVector(1,1)
  normalFace1[2]=Ci.getNormalVector(1,2)

  normalCell = normalFace0.crossProduct(normalFace1)
  normalCell = normalCell/normalCell.norm()

  cellMat=cdmath.Matrix(4)
  cellMat[0,0]=N0.x()
  cellMat[0,1]=N0.y()
  cellMat[0,2]=N0.z()
  cellMat[1,0]=N1.x()
  cellMat[1,1]=N1.y()
  cellMat[1,2]=N1.z()
  cellMat[2,0]=N2.x()
  cellMat[2,1]=N2.y()
  cellMat[2,2]=N2.z()
  cellMat[3,0]=normalCell[0]
  cellMat[3,1]=normalCell[1]
  cellMat[3,2]=normalCell[2]
  cellMat[0,3]=1
  cellMat[1,3]=1
  cellMat[2,3]=1
```

```
  cellMat[3,3]=0

  #Formule des gradients voir EF P1 -> calcul déterminants
  GradShapeFunc0[0]= cellMat.partMatrix(0,0).determinant()/2
  GradShapeFunc0[1]=-cellMat.partMatrix(0,1).determinant()/2
  GradShapeFunc0[2]= cellMat.partMatrix(0,2).determinant()/2
  GradShapeFunc1[0]=-cellMat.partMatrix(1,0).determinant()/2
  GradShapeFunc1[1]= cellMat.partMatrix(1,1).determinant()/2
  GradShapeFunc1[2]=-cellMat.partMatrix(1,2).determinant()/2
  GradShapeFunc2[0]= cellMat.partMatrix(2,0).determinant()/2
  GradShapeFunc2[1]=-cellMat.partMatrix(2,1).determinant()/2
  GradShapeFunc2[2]= cellMat.partMatrix(2,2).determinant()/2

  #Création d'un tableau (numéro du noeud, gradient de la fonction de forme
  GradShapeFuncs={nodeId0 : GradShapeFunc0}
  GradShapeFuncs[nodeId1]=GradShapeFunc1
  GradShapeFuncs[nodeId2]=GradShapeFunc2

  # Remplissage de  la matrice de rigidité et du second membre
  for j in [nodeId0,nodeId1,nodeId2] :
    #Ajout de la contribution de la cellule triangulaire i au second membre
    du noeud j
    RHS[j]=Ci.getMeasure()/3*my_RHSfield[j]+RHS[j] # intégrale dans le
   triangle du produit f x fonction de base
    #Contribution de la cellule triangulaire i à la ligne j du système liné
   aire
    for k in [nodeId0,nodeId1,nodeId2] :
      Rigidite.addValue(j,k,GradShapeFuncs[j]*GradShapeFuncs[k]/Ci.
   getMeasure())

print("Linear system matrix building done")

# Résolution du système linéaire
#================================
LS=cdmath.LinearSolver(Rigidite,RHS,100,1.E-6,"GMRES","ILU")
LS.setMatrixIsSingular()#En raison de l'absence de bord
SolSyst=LS.solve()
print "Preconditioner used : ", LS.getNameOfPc()
print "Number of iterations used : ", LS.getNumberOfIter()
print "Final residual : ", LS.getResidu()
print("Linear system solved")

# Création du champ résultat
#============================
my_ResultField = cdmath.Field("ResultField", cdmath.NODES, my_mesh, 1)
for j in range(nbNodes):
    my_ResultField[j]=SolSyst[j];#remplissage des valeurs pour les noeuds
   intérieurs
#sauvegarde sur le disque dur du résultat dans un fichier paraview
my_ResultField.writeVTK("FiniteElementsOnSpherePoisson")

#Postprocessing :
#================
# save 3D picture
PV_routines.Save_PV_data_to_picture_file("FiniteElementsOnSpherePoisson"+'
   _0.vtu','ResultField','NODES','FiniteElementsOnSpherePoisson")
resolution=100
```

```
VTK_routines.Clip_VTK_data_to_VTK("FiniteElementsOnSpherePoisson"+'_0.vtu', "Clip_VTK_data_to_VTK_"+ "FiniteElementsOnSpherePoisson"+'_0.vtu' ,[0.25,0.25,0.25], [-0.5,-0.5,-0.5],resolution )
PV_routines.Save_PV_data_to_picture_file("Clip_VTK_data_to_VTK_"+"FiniteElementsOnSpherePoisson"+'_0.vtu',"ResultField",'NODES',"Clip_VTK_data_to_VTK_"+"FiniteElementsOnSpherePoisson")

# Plot  over slice circle
finiteElementsOnSphere_0vtu = pvs.XMLUnstructuredGridReader(FileName=["FiniteElementsOnSpherePoisson"+'_0.vtu'])
slice1 = pvs.Slice(Input=finiteElementsOnSphere_0vtu)
slice1.SliceType.Normal = [0.5, 0.5, 0.5]
renderView1 = pvs.GetActiveViewOrCreate('RenderView')
finiteElementsOnSphere_0vtuDisplay = pvs.Show(finiteElementsOnSphere_0vtu, renderView1)
pvs.ColorBy(finiteElementsOnSphere_0vtuDisplay, ('POINTS', 'ResultField'))
slice1Display = pvs.Show(slice1, renderView1)
pvs.SaveScreenshot("./FiniteElementsOnSpherePoisson"+"_Slice"+'.png', magnification=1, quality=100, view=renderView1)
plotOnSortedLines1 = pvs.PlotOnSortedLines(Input=slice1)
lineChartView2 = pvs.CreateView('XYChartView')
plotOnSortedLines1Display = pvs.Show(plotOnSortedLines1, lineChartView2)
plotOnSortedLines1Display.UseIndexForXAxis = 0
plotOnSortedLines1Display.XArrayName = 'arc_length'
plotOnSortedLines1Display.SeriesVisibility = ['ResultField (1)']
pvs.SaveScreenshot("./FiniteElementsOnSpherePoisson"+"_PlotOnSortedLine_"+'.png', magnification=1, quality=100, view=lineChartView2)
pvs.Delete(lineChartView2)

print("Integral of the numerical solution", my_ResultField.integral(0))
print("Numerical solution of Poisson equation on a sphere using finite elements done")

#Calcul de l'erreur commise par rapport à la solution exacte
#============================================================
#The following formulas use the fact that the exact solution is equal the right hand side divided by 12
max_abs_sol_exacte=0
erreur_abs=0
max_sol_num=0
min_sol_num=0
for i in range(nbNodes) :
    if max_abs_sol_exacte < abs(my_RHSfield[i]) :
        max_abs_sol_exacte = abs(my_RHSfield[i])
    if erreur_abs < abs(my_RHSfield[i]/12 - my_ResultField[i]) :
        erreur_abs = abs(my_RHSfield[i]/12 - my_ResultField[i])
    if max_sol_num < my_ResultField[i] :
        max_sol_num = my_ResultField[i]
    if min_sol_num > my_ResultField[i] :
        min_sol_num = my_ResultField[i]
max_abs_sol_exacte = max_abs_sol_exacte/12

print("Absolute error = max(| exact solution - numerical solution |) = ", erreur_abs )
print("Relative error = max(| exact solution - numerical solution |)/max(| exact solution |) = ",erreur_abs/max_abs_sol_exacte)
print("Maximum numerical solution = ", max_sol_num, " Minimum numerical solution = ", min_sol_num)
```

```
print("Maximum exact solution = ", my_RHSfield.max()/12, " Minimum exact solution = ", my_RHSfield.min()/12 )

assert erreur_abs/max_abs_sol_exacte <1.
```

E.2 Python script to solve the Poisson problem on Torus

```
# -*-coding:utf-8 -*
#
    ================================================================================

# Name        : Résolution EF de l'équation de Laplace-Beltrami -\triangle u = f sur un tore
# Author      : Marcial Nguemfouo and Michael Ndjinga
# Copyright   : CEA Saclay 2018
# Description : Utilisation de la méthode des éléménts finis P1 avec champs u et f discrétisés aux noeuds d'un maillage triangulaire
#               Création et sauvegarde du champ résultant ainsi que du champ second membre en utilisant la librairie CDMATH
#               Référence : M. A. Olshanskii, A. Reusken, and J. Grande. A finite element method for elliptic equations
#                           on surfaces. SIAM J. Num. Anal., 47, p. 3355
#               Résolution d'un système linéaire à matrice singulière : les vecteurs constants sont dans le noyau
#
    ================================================================================


import cdmath
from math import sin, cos, atan2, sqrt
import PV_routines
import VTK_routines
import paraview.simple as pvs

#Chargement du maillage triangulaire du tore
#
    ================================================================================

my_mesh = cdmath.Mesh("meshTorus.med")
if(not my_mesh.isTriangular()) :
  raise ValueError("Wrong cell types : mesh is not made of triangles")
if(my_mesh.getMeshDimension()!=2) :
  raise ValueError("Wrong mesh dimension : expected a surface of dimension 2")
if(my_mesh.getSpaceDimension()!=3) :
  raise ValueError("Wrong space dimension : expected a space of dimension 3")
nbNodes = my_mesh.getNumberOfNodes()

nbCells = my_mesh.getNumberOfCells()

print("Mesh building/loading done")
print("nb of nodes=", nbNodes)
print("nb of cells=", nbCells)

# Torus radii (calculation  will fail if the mesh is not correct)
R=1 #Grand rayon
r=0.6 #Petit rayon
```

```
39
40 #Discrétisation du second membre, de la solution exacte et détermination
       des noeuds intérieurs
41 #=====================================================================
42 my_RHSfield = cdmath.Field("RHS field", cdmath.NODES, my_mesh, 1)
43 exactSolField = cdmath.Field("Exact solution field", cdmath.NODES, my_mesh,
       1)
44
45 maxNbNeighbours = 0#This is to determine the number of non zero
       coefficients in the sparse finite element rigidity matrix
46
47 #parcours des noeuds pour discrétisation du second membre et extraction du
       nb max voisins d'un noeud
48 for i in range(nbNodes):
49   Ni=my_mesh.getNode(i)
50   x = Ni.x()
51   y = Ni.y()
52   z = Ni.z()
53
54   theta=atan2(z,sqrt(x*x+y*y)-R)
55   phi=atan2(y,x)
56
57   exactSolField[i] = sin(3*phi)*cos(3*theta+ phi) # for the exact solution
       we use the funtion given in the article of Olshanskii, Reusken 2009,
       page 19
58   my_RHSfield[i] = 9*sin(3*phi)*cos(3*theta+ phi)/(r*r) + (10*sin(3*phi)*
       cos(3*theta+ phi) + 6*cos(3*phi)*sin(3*theta+ phi))/((R+r*cos(theta))*(R
       +r*cos(theta))) - 3*sin(theta)*sin(3*phi)*sin(3*theta+ phi)/(r*(R+r*cos(
       theta))) #for the right hand side we use the function given in the
       article of Olshanskii, Reusken 2009, page 19
59   if my_mesh.isBorderNode(i): # Détection des noeuds frontière
60     raise ValueError("Mesh should not contain borders")
61   else:
62     maxNbNeighbours = max(1+Ni.getNumberOfCells(),maxNbNeighbours)#true
       only for planar cells, otherwise use function Ni.getNumberOfEdges()
63
64 print("Right hand side discretisation done")
65 print("Max nb of neighbours=", maxNbNeighbours)
66 print("Integral of the RHS", my_RHSfield.integral(0))
67
68 # Construction de la matrice de rigidité et du vecteur second membre du
       système linéaire
69 #
       ================================================================================
70 Rigidite=cdmath.SparseMatrixPetsc(nbNodes,nbNodes,maxNbNeighbours)
71 RHS=cdmath.Vector(nbNodes)
72
73 # Vecteurs gradient de la fonction de forme associée à chaque noeud d'un
       triangle
74 GradShapeFunc0=cdmath.Vector(3)
75 GradShapeFunc1=cdmath.Vector(3)
76 GradShapeFunc2=cdmath.Vector(3)
77
78 normalFace0=cdmath.Vector(3)
79 normalFace1=cdmath.Vector(3)
80
81 #On parcourt les triangles du domaine
```

```
for i in range(nbCells):

  Ci=my_mesh.getCell(i)

  #Contribution à la matrice de rigidité
  nodeId0=Ci.getNodeId(0)
  nodeId1=Ci.getNodeId(1)
  nodeId2=Ci.getNodeId(2)
  N0=my_mesh.getNode(nodeId0)
  N1=my_mesh.getNode(nodeId1)
  N2=my_mesh.getNode(nodeId2)

  #Build normal to cell Ci
  normalFace0[0]=Ci.getNormalVector(0,0)
  normalFace0[1]=Ci.getNormalVector(0,1)
  normalFace0[2]=Ci.getNormalVector(0,2)
  normalFace1[0]=Ci.getNormalVector(1,0)
  normalFace1[1]=Ci.getNormalVector(1,1)
  normalFace1[2]=Ci.getNormalVector(1,2)

  normalCell = normalFace0.crossProduct(normalFace1)
  normalCell = normalCell/normalCell.norm()

  cellMat=cdmath.Matrix(4)
  cellMat[0,0]=N0.x()
  cellMat[0,1]=N0.y()
  cellMat[0,2]=N0.z()
  cellMat[1,0]=N1.x()
  cellMat[1,1]=N1.y()
  cellMat[1,2]=N1.z()
  cellMat[2,0]=N2.x()
  cellMat[2,1]=N2.y()
  cellMat[2,2]=N2.z()
  cellMat[3,0]=normalCell[0]
  cellMat[3,1]=normalCell[1]
  cellMat[3,2]=normalCell[2]
  cellMat[0,3]=1
  cellMat[1,3]=1
  cellMat[2,3]=1
  cellMat[3,3]=0

  #Formule des gradients voir EF P1 -> calcul déterminants
  GradShapeFunc0[0]= cellMat.partMatrix(0,0).determinant()/2
  GradShapeFunc0[1]=-cellMat.partMatrix(0,1).determinant()/2
  GradShapeFunc0[2]= cellMat.partMatrix(0,2).determinant()/2
  GradShapeFunc1[0]=-cellMat.partMatrix(1,0).determinant()/2
  GradShapeFunc1[1]= cellMat.partMatrix(1,1).determinant()/2
  GradShapeFunc1[2]=-cellMat.partMatrix(1,2).determinant()/2
  GradShapeFunc2[0]= cellMat.partMatrix(2,0).determinant()/2
  GradShapeFunc2[1]=-cellMat.partMatrix(2,1).determinant()/2
  GradShapeFunc2[2]= cellMat.partMatrix(2,2).determinant()/2

  #Création d'un tableau (numéro du noeud, gradient de la fonction de forme
  GradShapeFuncs={nodeId0 : GradShapeFunc0}
  GradShapeFuncs[nodeId1]=GradShapeFunc1
  GradShapeFuncs[nodeId2]=GradShapeFunc2

  # Remplissage de  la matrice de rigidité et du second membre
```

```
  for j in [nodeId0,nodeId1,nodeId2] :
    #Ajout de la contribution de la cellule triangulaire i au second membre du noeud j
    RHS[j]=Ci.getMeasure()/3*my_RHSfield[j]+RHS[j] # intégrale dans le triangle du produit f x fonction de base
    #Contribution de la cellule triangulaire i à la ligne j du système linéaire
    for k in [nodeId0,nodeId1,nodeId2] :
      Rigidite.addValue(j,k,GradShapeFuncs[j]*GradShapeFuncs[k]/Ci.getMeasure())

print("Linear system matrix building done")

# Résolution du système linéaire
#================================
LS=cdmath.LinearSolver(Rigidite,RHS,100,1.E-6,"CG","ILU")#Remplacer CG par CHOLESKY pour solveur direct
LS.setMatrixIsSingular()#En raison de l'absence de bord
SolSyst=LS.solve()
print "Preconditioner used : ", LS.getNameOfPc()
print "Number of iterations used : ", LS.getNumberOfIter()
print "Final residual : ", LS.getResidu()
print("Linear system solved")

# Création du champ résultat
#============================
my_ResultField = cdmath.Field("Numerical result field", cdmath.NODES, my_mesh, 1)
for j in range(nbNodes):
    my_ResultField[j]=SolSyst[j];#remplissage des valeurs pour les noeuds intérieurs
#sauvegarde sur le disque dur du résultat dans un fichier paraview
my_ResultField.writeVTK("FiniteElementsOnTorusPoisson")

print("Integral of the numerical solution", my_ResultField.integral(0))
print("Numerical solution of Poisson equation on a torus using finite elements done")

#Calcul de l'erreur commise par rapport à la solution exacte
#============================================================
max_sol_exacte=exactSolField.getNormEuclidean().max()
erreur_max=(exactSolField - my_ResultField).getNormEuclidean().max()
max_sol_num=my_ResultField.max()
min_sol_num=my_ResultField.min()

print("Absolute error =  max(| exact solution - numerical solution |)/max(| exact solution |) = ",erreur_max/max_sol_exacte)
print("Maximum numerical solution = ", max_sol_num, " Minimum numerical solution = ", min_sol_num)
print("Maximum exact solution = ", exactSolField.max(), " Minimum exact solution = ", exactSolField.min())

assert erreur_max/max_sol_exacte <1.

#Postprocessing :
#================
# Save 3D picture
PV_routines.Save_PV_data_to_picture_file("FiniteElementsOnTorusPoisson"+'_0
```

```
    .vtu','Numerical result field','NODES',"FiniteElementsOnTorusPoisson")
resolution=100
VTK_routines.Clip_VTK_data_to_VTK("FiniteElementsOnTorusPoisson"+'_0.vtu',"
    Clip_VTK_data_to_VTK_"+ "FiniteElementsOnTorusPoisson"+'_0.vtu'
    ,[0.25,0.25,0.25], [-0.5,-0.5,-0.5],resolution )
PV_routines.Save_PV_data_to_picture_file("Clip_VTK_data_to_VTK_"+"
    FiniteElementsOnTorusPoisson"+'_0.vtu',"Numerical result field",'NODES',
    "Clip_VTK_data_to_VTK_"+"FiniteElementsOnTorusPoisson")

# Plot  over slice circle
finiteElementsOnTorus_0vtu = pvs.XMLUnstructuredGridReader(FileName=["
    FiniteElementsOnTorusPoisson"+'_0.vtu'])
slice1 = pvs.Slice(Input=finiteElementsOnTorus_0vtu)
slice1.SliceType.Normal = [0.5, 0.5, 0.5]
renderView1 = pvs.GetActiveViewOrCreate('RenderView')
finiteElementsOnTorus_0vtuDisplay = pvs.Show(finiteElementsOnTorus_0vtu,
    renderView1)
pvs.ColorBy(finiteElementsOnTorus_0vtuDisplay, ('POINTS', 'Numerical result
     field'))
slice1Display = pvs.Show(slice1, renderView1)
pvs.SaveScreenshot("./FiniteElementsOnTorusPoisson"+"_Slice"+'.png',
    magnification=1, quality=100, view=renderView1)
plotOnSortedLines1 = pvs.PlotOnSortedLines(Input=slice1)
lineChartView2 = pvs.CreateView('XYChartView')
plotOnSortedLines1Display = pvs.Show(plotOnSortedLines1, lineChartView2)
plotOnSortedLines1Display.UseIndexForXAxis = 0
plotOnSortedLines1Display.XArrayName = 'arc_length'
plotOnSortedLines1Display.SeriesVisibility = ['Numerical result field (1)']
pvs.SaveScreenshot("./FiniteElementsOnTorusPoisson"+"_PlotOnSortedLine_"+'.
    png', magnification=1, quality=100, view=lineChartView2)
pvs.Delete(lineChartView2)
```

E.3 Python script to solve the Poisson problem on Cube skin

```
# -*-coding:utf-8 -*
#
    ================================================================================

# Name        : Résolution EF de l'équation de Laplace-Beltrami -\triangle
    u = f sur la frontière d'un cube
# Author      : Marcial Nguemfouo and Michael Ndjinga
# Copyright   : CEA Saclay 2021
# Description : Utilisation de la méthode des éléménts finis P1 avec champs
     u et f discrétisés aux noeuds d'un maillage triangulaire
#               Création et sauvegarde du champ résultant ainsi que du
    champ second membre en utilisant la librairie CDMATH
#               Résolution d'un système linéaire à matrice singulière : les
     vecteurs constants sont dans le noyau
#               Comparaison de la solution numérique avec la solution
    exacte définie face par face : u(x,y,z)= cos(2*pi*x)*cos(2*pi*y)*cos(2*
    pi*z)
#
    ================================================================================


import cdmath
from math import cos, pi
import numpy as np
```

```
import PV_routines
import VTK_routines
import paraview.simple as pvs

#Chargement du maillage triangulaire de la frontière du cube unité [0,1]x[0,1]x[0,1]
#============================================================================

my_mesh = cdmath.Mesh("meshCubeSkin.med")
if(not my_mesh.isTriangular()) :
  raise ValueError("Wrong cell types : mesh is not made of triangles")
if(my_mesh.getMeshDimension()!=2) :
  raise ValueError("Wrong mesh dimension : expected a surface of dimension 2")
if(my_mesh.getSpaceDimension()!=3) :
  raise ValueError("Wrong space dimension : expected a space of dimension 3")

nbNodes = my_mesh.getNumberOfNodes()
nbCells = my_mesh.getNumberOfCells()

print("Mesh building/loading done")
print("nb of nodes=", nbNodes)
print("nb of cells=", nbCells)

#Discrétisation du second membre et détermination des noeuds intérieurs
#======================================================================
my_RHSfield = cdmath.Field("RHS field", cdmath.NODES, my_mesh, 1)
maxNbNeighbours = 0#This is to determine the number of non zero coefficients in the sparse finite element rigidity matrix

eps=1e-6
#parcours des noeuds pour discrétisation du second membre et extraction du nb max voisins d'un noeud
for i in range(nbNodes):
  Ni=my_mesh.getNode(i)
  x = Ni.x()
  y = Ni.y()
  z = Ni.z()

  my_RHSfield[i]= 8*pi*pi*cos(2*pi*x)*cos(2*pi*y)*cos(2*pi*z)

  if my_mesh.isBorderNode(i): # Détection des noeuds frontière
    raise ValueError("Mesh should not contain borders")
  else:
    maxNbNeighbours = max(1+Ni.getNumberOfCells(),maxNbNeighbours) #true only for planar cells, otherwise use function Ni.getNumberOfEdges()

print("Right hand side discretisation done")
print("Max nb of neighbours=", maxNbNeighbours)
print("Integral of the RHS", my_RHSfield.integral(0))

# Construction de la matrice de rigidité et du vecteur second membre du système linéaire
#========================================================================
```

```
Rigidite=cdmath.SparseMatrixPetsc(nbNodes,nbNodes,maxNbNeighbours)# warning
    : third argument is number of non zero coefficients per line
RHS=cdmath.Vector(nbNodes)

# Vecteurs gradient de la fonction de forme associée à chaque noeud d'un
   triangle
GradShapeFunc0=cdmath.Vector(3)
GradShapeFunc1=cdmath.Vector(3)
GradShapeFunc2=cdmath.Vector(3)

normalFace0=cdmath.Vector(3)
normalFace1=cdmath.Vector(3)

#On parcourt les triangles du domaine
for i in range(nbCells):

  Ci=my_mesh.getCell(i)

  #Contribution à la matrice de rigidité
  nodeId0=Ci.getNodeId(0)
  nodeId1=Ci.getNodeId(1)
  nodeId2=Ci.getNodeId(2)
  N0=my_mesh.getNode(nodeId0)
  N1=my_mesh.getNode(nodeId1)
  N2=my_mesh.getNode(nodeId2)

  #Build normal to cell Ci
  normalFace0[0]=Ci.getNormalVector(0,0)
  normalFace0[1]=Ci.getNormalVector(0,1)
  normalFace0[2]=Ci.getNormalVector(0,2)
  normalFace1[0]=Ci.getNormalVector(1,0)
  normalFace1[1]=Ci.getNormalVector(1,1)
  normalFace1[2]=Ci.getNormalVector(1,2)

  normalCell = normalFace0.crossProduct(normalFace1)
  normalCell = normalCell*(1/normalCell.norm())

  cellMat=cdmath.Matrix(4)
  cellMat[0,0]=N0.x()
  cellMat[0,1]=N0.y()
  cellMat[0,2]=N0.z()
  cellMat[1,0]=N1.x()
  cellMat[1,1]=N1.y()
  cellMat[1,2]=N1.z()
  cellMat[2,0]=N2.x()
  cellMat[2,1]=N2.y()
  cellMat[2,2]=N2.z()
  cellMat[3,0]=normalCell[0]
  cellMat[3,1]=normalCell[1]
  cellMat[3,2]=normalCell[2]
  cellMat[0,3]=1
  cellMat[1,3]=1
  cellMat[2,3]=1
  cellMat[3,3]=0

  #Formule des gradients voir EF P1 -> calcul déterminants
  GradShapeFunc0[0]= cellMat.partMatrix(0,0).determinant()*0.5
  GradShapeFunc0[1]=-cellMat.partMatrix(0,1).determinant()*0.5
```

```
  GradShapeFunc0[2]= cellMat.partMatrix(0,2).determinant()*0.5
  GradShapeFunc1[0]=-cellMat.partMatrix(1,0).determinant()*0.5
  GradShapeFunc1[1]= cellMat.partMatrix(1,1).determinant()*0.5
  GradShapeFunc1[2]=-cellMat.partMatrix(1,2).determinant()*0.5
  GradShapeFunc2[0]= cellMat.partMatrix(2,0).determinant()*0.5
  GradShapeFunc2[1]=-cellMat.partMatrix(2,1).determinant()*0.5
  GradShapeFunc2[2]= cellMat.partMatrix(2,2).determinant()*0.5

  #Création d'un tableau (numéro du noeud, gradient de la fonction de forme
  GradShapeFuncs={nodeId0 : GradShapeFunc0}
  GradShapeFuncs[nodeId1]=GradShapeFunc1
  GradShapeFuncs[nodeId2]=GradShapeFunc2

  # Remplissage de  la matrice de rigidité et du second membre
  for j in [nodeId0,nodeId1,nodeId2] :
    #Ajout de la contribution de la cellule triangulaire i au second membre
     du noeud j
    RHS[j]=Ci.getMeasure()/3*my_RHSfield[j]+RHS[j] # intégrale dans le
    triangle du produit f x fonction de base
    #Contribution de la cellule triangulaire i à la ligne j du système liné
    aire
    for k in [nodeId0,nodeId1,nodeId2] :
      Rigidite.addValue(j,k,GradShapeFuncs[j]*GradShapeFuncs[k]/Ci.
    getMeasure())

print("Linear system matrix building done")

# Conditionnement de la matrice de rigidité
#==================================
cond = Rigidite.getConditionNumber(True)
print("Condition number is ",cond)

# Résolution du système linéaire
#==================================
LS=cdmath.LinearSolver(Rigidite,RHS,100,1.E-6,"GMRES","ILU")
LS.setMatrixIsSingular()#En raison de l'absence de bord
SolSyst=LS.solve()
print("Preconditioner used : ", LS.getNameOfPc() )
print("Number of iterations used : ", LS.getNumberOfIter() )
print("Final residual : ", LS.getResidu() )
print("Linear system solved")

# Création du champ résultat
#============================
my_ResultField = cdmath.Field("ResultField", cdmath.NODES, my_mesh, 1)
for j in range(nbNodes):
    my_ResultField[j]=SolSyst[j];#remplissage des valeurs issues du système
     linéaire dans le champs résultat
#sauvegarde sur le disque dur du résultat dans un fichier paraview
my_ResultField.writeVTK("FiniteElementsOnCubeSkinPoisson")
my_RHSfield.writeVTK("RHS_CubeSkinPoisson")

print("Integral of the numerical solution", my_ResultField.integral(0))
print("Numerical solution of Poisson equation on a cube skin using finite
   elements done")

#Calcul de l'erreur commise par rapport à la solution exacte
#============================================================
```

```
#The following formulas use the fact that the exact solution is equal the right hand side divided by 8*pi*pi
max_abs_sol_exacte=0
erreur_abs=0
max_sol_num=0
min_sol_num=0
for i in range(nbNodes) :
    if max_abs_sol_exacte < abs(my_RHSfield[i]) :
        max_abs_sol_exacte = abs(my_RHSfield[i])
    if erreur_abs  < abs(my_RHSfield[i]/(8*pi*pi) - my_ResultField[i]) :
        erreur_abs = abs(my_RHSfield[i]/(8*pi*pi) - my_ResultField[i])
    if max_sol_num  < my_ResultField[i] :
        max_sol_num = my_ResultField[i]
    if min_sol_num  > my_ResultField[i] :
        min_sol_num = my_ResultField[i]
max_abs_sol_exacte = max_abs_sol_exacte/(8*pi*pi)

print("Relative error = max(| exact solution - numerical solution |)/max(| exact solution |) = ",erreur_abs/max_abs_sol_exacte)
print("Maximum numerical solution = ", max_sol_num, " Minimum numerical solution = ", min_sol_num)
print("Maximum exact solution = ", my_RHSfield.max()/(8*pi*pi), " Minimum exact solution = ", my_RHSfield.min()/(8*pi*pi) )

#Postprocessing :
#================
# save 3D picture
PV_routines.Save_PV_data_to_picture_file("FiniteElementsOnCubeSkinPoisson"+'_0.vtu',"ResultField",'NODES',"FiniteElementsOnCubeSkinPoisson")
resolution=100
VTK_routines.Clip_VTK_data_to_VTK("FiniteElementsOnCubeSkinPoisson"+'_0.vtu','Clip_VTK_data_to_VTK_"+ "FiniteElementsOnCubeSkinPoisson"+'_0.vtu',[0.75,0.75,0.75], [0.,0.5,-0.5],resolution )
PV_routines.Save_PV_data_to_picture_file("Clip_VTK_data_to_VTK_"+"FiniteElementsOnCubeSkinPoisson"+'_0.vtu',"ResultField",'NODES',"Clip_VTK_data_to_VTK_"+"FiniteElementsOnCubeSkinPoisson")

# Plot  over slice circle
finiteElementsOnCubeSkin_0vtu = pvs.XMLUnstructuredGridReader(FileName=["FiniteElementsOnCubeSkinPoisson"+'_0.vtu'])
slice1 = pvs.Slice(Input=finiteElementsOnCubeSkin_0vtu)
slice1.SliceType.Normal = [0, 1, 0]
renderView1 = pvs.GetActiveViewOrCreate('RenderView')
finiteElementsOnCubeSkin_0vtuDisplay = pvs.Show(finiteElementsOnCubeSkin_0vtu, renderView1)
pvs.ColorBy(finiteElementsOnCubeSkin_0vtuDisplay, ('POINTS', 'ResultField'))
slice1Display = pvs.Show(slice1, renderView1)
pvs.SaveScreenshot("./FiniteElementsOnCubeSkinPoisson"+"_Slice"+'.png', magnification=1, quality=100, view=renderView1)
plotOnSortedLines1 = pvs.PlotOnSortedLines(Input=slice1)
lineChartView2 = pvs.CreateView('XYChartView')
plotOnSortedLines1Display = pvs.Show(plotOnSortedLines1, lineChartView2)
plotOnSortedLines1Display.UseIndexForXAxis = 0
plotOnSortedLines1Display.XArrayName = 'arc_length'
plotOnSortedLines1Display.SeriesVisibility = ['ResultField (1)']
pvs.SaveScreenshot("./FiniteElementsOnCubeSkinPoisson"+"_PlotOnSortedLine_"+'.png', magnification=1, quality=100, view=lineChartView2)
```

```
pvs.Delete(lineChartView2)

assert erreur_abs/max_abs_sol_exacte <1.
```

E.4 Python script to investigate the Condition number

```
# -*-coding:utf-8 -*
#
    ================================================================================

# Name        : Résolution EF de l'équation de Laplace-Beltrami -\triangle
    u = f sur une sphere
# Author      : Marcial Nguemfouo and Michael Ndjinga
# Copyright   : CEA Saclay 2017
# Description : Utilisation de la méthode des éléménts finis P1 avec champs
    u et f discrétisés aux noeuds d'un maillage triangulaire
#       Création et sauvegarde du champ résultant ainsi que du champ second
    membre en utilisant la librairie CDMATH
#
    ================================================================================


import cdmath
import time
from math import pow
import numpy as np
import PV_routines
import VTK_routines
import paraview.simple as pvs
from math import log10, sqrt


def solve_condNumber(filename,resolution):
    start = time.time()
    #Préprocessing optionnel: création du fichier my_mesh.med contenant la
    géométrie et le maillage du domaine de calcul à partir de commandes
    python (import salome)

    #Chargement du maillage triangulaire de la sphère
    #
    ================================================================================

    my_mesh = cdmath.Mesh(filename+".med")
    if(not my_mesh.isTriangular()) :
        raise ValueError("Wrong cell types : mesh is not made of triangles"
    )
    if(my_mesh.getMeshDimension()!=2) :
        raise ValueError("Wrong mesh dimension : expected a surface of
    dimension 2")
    if(my_mesh.getSpaceDimension()!=3) :
        raise ValueError("Wrong space dimension : expected a space of
    dimension 3")

    nbNodes = my_mesh.getNumberOfNodes()
    nbCells = my_mesh.getNumberOfCells()
    diameter= my_mesh.minRatioVolSurf()

    print("Mesh building/loading done")
```

```
    print("nb of nodes=", nbNodes)
    print("nb of cells=", nbCells)

    #Discrétisation du second membre et détermination des noeuds intérieurs
    #=========================================================================
    my_RHSfield = cdmath.Field("RHS field", cdmath.NODES, my_mesh, 1)
    maxNbNeighbours = 0#This is to determine the number of non zero coefficients in the sparse finite element rigidity matrix

    #parcours des noeuds pour discrétisation du second membre et extraction du nb max voisins d'un noeud
    for i in range(nbNodes):
        Ni=my_mesh.getNode(i)
        x = Ni.x()
        y = Ni.y()
        z = Ni.z()

        my_RHSfield[i]=12*y*(3*x*x-y*y)/pow(x*x+y*y+z*z,3/2)#vecteur propre du laplacien sur la sphère
        if my_mesh.isBorderNode(i): # Détection des noeuds frontière
            raise ValueError("Mesh should not contain borders")
        else:
            maxNbNeighbours = max(1+Ni.getNumberOfCells(),maxNbNeighbours)

    # sauvegarde sur le disque dur du second membre discrétisé dans un fichier paraview
    my_RHSfield.writeVTK("FiniteElementsOnSphereRHSField")
    intRHS = abs( my_RHSfield.integral(0) )#We adjust the precision because de RHS should be in the kernel of the singular matrix

    print("Right hand side discretisation done")
    print("Max nb of neighbours=", maxNbNeighbours)
    print("Integral of the RHS", intRHS )

    # Construction de la matrice de rigidité et du vecteur second membre du système linéaire
    #
    ================================================================================

    Rigidite=cdmath.SparseMatrixPetsc(nbNodes,nbNodes,maxNbNeighbours)# warning : third argument is number of non zero coefficients per line
    RHS=cdmath.Vector(nbNodes)

    # Vecteurs gradient de la fonction de forme associée à chaque noeud d'un triangle
    GradShapeFunc0=cdmath.Vector(3)
    GradShapeFunc1=cdmath.Vector(3)
    GradShapeFunc2=cdmath.Vector(3)

    normalFace0=cdmath.Vector(3)
    normalFace1=cdmath.Vector(3)

    #On parcourt les triangles du domaine
    for i in range(nbCells):

        Ci=my_mesh.getCell(i)

        #Contribution à la matrice de rigidité
```

```
        nodeId0=Ci.getNodeId(0)
        nodeId1=Ci.getNodeId(1)
        nodeId2=Ci.getNodeId(2)
        N0=my_mesh.getNode(nodeId0)
        N1=my_mesh.getNode(nodeId1)
        N2=my_mesh.getNode(nodeId2)

        #Build normal to cell Ci
        normalFace0[0]=Ci.getNormalVector(0,0)
        normalFace0[1]=Ci.getNormalVector(0,1)
        normalFace0[2]=Ci.getNormalVector(0,2)
        normalFace1[0]=Ci.getNormalVector(1,0)
        normalFace1[1]=Ci.getNormalVector(1,1)
        normalFace1[2]=Ci.getNormalVector(1,2)

        normalCell = normalFace0.crossProduct(normalFace1)
        test = normalFace0.tensProduct(normalFace1)
        normalCell = normalCell*(1./normalCell.norm())

        cellMat=cdmath.Matrix(4)
        cellMat[0,0]=N0.x()
        cellMat[0,1]=N0.y()
        cellMat[0,2]=N0.z()
        cellMat[1,0]=N1.x()
        cellMat[1,1]=N1.y()
        cellMat[1,2]=N1.z()
        cellMat[2,0]=N2.x()
        cellMat[2,1]=N2.y()
        cellMat[2,2]=N2.z()
        cellMat[3,0]=normalCell[0]
        cellMat[3,1]=normalCell[1]
        cellMat[3,2]=normalCell[2]
        cellMat[0,3]=1
        cellMat[1,3]=1
        cellMat[2,3]=1
        cellMat[3,3]=0

        #Formule des gradients voir EF P1 -> calcul déterminants
        GradShapeFunc0[0]= cellMat.partMatrix(0,0).determinant()/2
        GradShapeFunc0[1]=-cellMat.partMatrix(0,1).determinant()/2
        GradShapeFunc0[2]= cellMat.partMatrix(0,2).determinant()/2
        GradShapeFunc1[0]=-cellMat.partMatrix(1,0).determinant()/2
        GradShapeFunc1[1]= cellMat.partMatrix(1,1).determinant()/2
        GradShapeFunc1[2]=-cellMat.partMatrix(1,2).determinant()/2
        GradShapeFunc2[0]= cellMat.partMatrix(2,0).determinant()/2
        GradShapeFunc2[1]=-cellMat.partMatrix(2,1).determinant()/2
        GradShapeFunc2[2]= cellMat.partMatrix(2,2).determinant()/2

        #Création d'un tableau (numéro du noeud, gradient de la fonction de
    forme
        GradShapeFuncs={nodeId0 : GradShapeFunc0}
        GradShapeFuncs[nodeId1]=GradShapeFunc1
        GradShapeFuncs[nodeId2]=GradShapeFunc2

        # Remplissage de  la matrice de rigidité et du second membre
        for j in [nodeId0,nodeId1,nodeId2] :
            #Ajout de la contribution de la cellule triangulaire i au
  second membre du noeud j
```

```
            RHS[j]=Ci.getMeasure()/3*my_RHSfield[j]+RHS[j] # intégrale dans le triangle du produit f x fonction de base
            #Contribution de la cellule triangulaire i à la ligne j du système linéaire
            for k in [nodeId0,nodeId1,nodeId2] :
                Rigidite.addValue(j,k,GradShapeFuncs[j]*GradShapeFuncs[k]/Ci.getMeasure())

    print("Linear system matrix building done")

    CondNumb=Rigidite.getConditionNumber(True, 1.e-3)
    print( "Condition number : ", log10(CondNumb))
   # print(np.format_float_scientific(CondNumb, precision = 1, exp_digits=1))

    return CondNumb, diameter

if __name__ == """__main__""":
   solve_condNumber("meshSphere152516",100)
```

```
import cdmath
import FiniteElementsOnSphereWithConditionNumber
import matplotlib.pyplot as plt
import numpy as np
from math import log10, sqrt

def test_validation3DSphereEF():
    #### 3D sphere FE triangle mesh
    meshList=['meshSphere_1','meshSphere_2','meshSphere_3','meshSphere_4','meshSphere_5','meshSphere152516','meshSphere874742','meshSphere1178302']
    nbMeshes=len(meshList)
    error_tab=[0]*nbMeshes
    mesh_size_tab=[0]*nbMeshes
    mesh_size=[0]*nbMeshes
    CondNumb_tab=[0]*nbMeshes
    Condition=[0]*nbMeshes
    diameter_tab=[0]*nbMeshes
    mesh_path='./'
    mesh_name='meshSphereWithTrianglesFE'
    diag_data=[0]*nbMeshes
    resolution=100
    i=0
    # Storing of numerical errors and mesh sizes
    for filename in meshList:
        CondNumb_tab[i], diameter_tab[i] = FiniteElementsOnSphereWithConditionNumber.solve_condNumber(mesh_path+filename, resolution)
        Condition[i]=log10(CondNumb_tab[i])
        # mesh_size[i]=np.format_float_scientific(mesh_size_tab[i], precision=0, exp_digits=1)
        diameter_tab[i]=log10(diameter_tab[i])
        i=i+1
    # Least square linear regression
    # Find the best a,b such that f(x)=ax+b best approximates the convergence curve
    # The vector X=(a,b) solves a symmetric linear system AX=B with A=(a1, a2\\a2,a3), B=(b1,b2)
    a1=np.dot(diameter_tab,diameter_tab)
```

```
    a2=np.sum(diameter_tab)
    a3=nbMeshes
    b1=np.dot(Condition,diameter_tab)
    b2=np.sum(Condition)

    det=a1*a3-a2*a2

    a=( a3*b1-a2*b2)/det
    b=(-a2*b1+a1*b2)/det

    print( "FE on 3D sphere triangle mesh : scheme order is ", -a)


    # Plot of iteration number
    plt.close()
    plt.plot(diameter_tab, Condition, label='Condition number of the FE matrix')
    plt.plot(diameter_tab, a*np.array(diameter_tab)+b,label='least square slope : '+'%.3f' % a)
    plt.legend()
    plt.xlabel('Log(h)')
    # plt.xlabel('Number of nodes')
    plt.ylabel('Log(Condition number)')
    plt.title('Condition number for finite elements \n for Laplace operator on 3D sphere triangular meshes')
    plt.savefig(mesh_name+"_3DSpherePoissonFE_ConditionNumber.png")



if __name__ == """__main__""":

    test_validation3DSphereEF()
```

♣ Bibliography ♣

[1] R. A. Adams, *Sobolev Spaces*, Academic Press, New York, 1975.

[2] R. A. Adams, J. J. F. Fournier, *Sobolev Spaces*, Pure and Applied Mathematics series, Elsevier, Boston, **140**, 2003.

[3] G. Allaire, *Numerical analysis and optimization an introduction to mathematical modeling and numerical simulation*, Oxford University Press, New York, 2007.

[4] T. Aubin, *Some nonlinear problems in Riemannian geometry*, Springer Monographs in Mathematics, 1998.

[5] J.A. Baerentzen et al., *Guide to computational geometry processing*, DOI 10.1007/978-1-4471-4075-7.

[6] Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W., Karpeyev, D., Kaushik, D., Knepley, M., May, D., Curfman McInnes, L., Mills, R., Munson, T., Rupp, K., Sanan, P., Smith, B., Zampini, S., Zhang, H.: PETSc/TAO Users Manual, Argonne National Laboratory, ANL-21/39 - Revision 3.17, (2021), https://petsc.org/release/docs/manual/manual.pdf.

[7] V. Bergeaud and V. Lefebvre. *SALOME. A software integration platform for multi-physics, preprocessing and visualisation*, Proceedings of SNA + MC2010: international conference on supercomputing in nuclear applications + Monte Carlo, Tokyo 2010.

[8] F. Boyer, P. Fabrie, *Mathematical tools for the study of the incompressible Navier-Stokes equations and related models,* Applied Mathematical Sciences, Springer, New York, **183**, 2013.

[9] D. Braess, *Finite Elements, Theory, Fast Solvers, and Applications in Elasticity Theory*, Cambridge University Press, New York, 2007.

[10] H. Brezis, *Functional Analysis, Sobolev Spaces and PDEs*, Springer, 2010.

[11] I. Chavel, *Eigenvalues in Riemannian geometry*, 2nd ed., Academic Press, 1984.

[12] P. Ciarlet, Basic Error Estimates for Elliptic Problems, Val. II: Finite Element Methods, ch. 2. Handbook of Numerical Analysis, P. G. Ciarlet and J-L. Lions, editors. North-Holland, Amsterdam, 1991.

[13] A. Demlow, *Higher order finite element methods and pointwise error estimates for elliptic problems on surfaces*. *SIAM Journal on Numerical Analysis*, **47** (2009), 805-827. https://doi.org/10.1137/070708135.

[14] A. Demlow, M. A. Olshanskii, *An adaptive surface finite element method based on volume meshes, SIAM J. Numer. Anal.*, **50** (2012), 1624-1647.

[15] G. Dziuk, *Finite elements for the Beltrami operator on arbitrary surfaces.* Partial Differential Equations and Calculus Of Variations, Lecture Notes in Math., Springer, Berlin, **1357** (1988), 142-155.

[16] G. Dziuk and C. M. Elliott, *Finite element methods for surface PDEs*, Acta Numerica, **22** (2013), 289-396.

[17] M. Elliot, B. Stinner, *Computation of two-phase biomembranes with phase dependent material parameters using surface finite elements*, Commun. Comput. Phys., **13** (2010), 325-360.

[18] H. Edelsbrunner, *Geometry and Topology for Mesh Generation.* Cambridge Monographs on Applied and Computational Mathematics, *Cambridge University Press, Cambridge*, **7** (2006), 978-0-521-68207-7.

[19] A. Ern, J.-L. Guermond. *Theory and practice of finite elements*, Springer Science & Business Media, New York, **159**, 2013.

[20] A. Ern, and J-L. Guermond, *Evaluation for the condition number in linear system arising in finite element approximations,* ESAIM: Mathematical Modelling and Numerical Analysis, **40** (2006), 29-48.

[21] L. C. Evans, *Partial Differential Equations,* first edition, Graduate Studies in Mathematics, AMS, 1998.

[22] F. Gesztesy, I. Mitrea, D. Mitrea, M. Mitrea, *On the nature of the Laplace-Beltrami operator on Lipschitz manifolds,* Journal of Mathematical Sciences, **172** (2011), 279–346.

[23] C. Giovanni, *Oscillator spacetimes are Ricci solutons*, Nonlinear Analysis, **140** (2016), 254-269.

[24] V. Gol'dshtein, I. Mitrea, M. Mitrea, *Hodge Decompositions with mixed boundary conditions and application to partial differential equations on Lipschitz manifolds,* Journal of Mathematical Sciences, **172** (2011), 347–400.

[25] C. Gordon, D. Webb and S. Wolpert, *Isospectral plane domains and surfaces via Riemannian orbifolds*, Inventiones Mathematicae, **110** (1992), 1–22.

[26] C.C. Green, J.S. Marshall, *From Green's function for the Laplace–Beltrami operator on a toroidal surface*, Proc R Soc A, **469** 2013. DOI:http://dx.doi.org/10.1098/rspa.2012.0479.

[27] J. Guzman, A. Madureira, M. Sarkis, S. Walker, *Analysis of the finite element method for the Laplace–Beltrami equation on surfaces with regions of high curvature using graded meshes*, arXiv preprint arXiv:1705.04369, 2017.

[28] E. Hebey, *Nonlinear analysis on manifolds ; Sobolev spaces and inequalities*, Courant Lecture Notes, **5,** 2000.

[29] K. Hildebrandt, K. Polthier, On approximation of the Laplace-Beltrami operator and the Willmore energy of surfaces, *Eurographics Symposium on Geometry Processing*, **30,** 2011.

[30] T. Karkanis, A. J. Stewart, High Quality, Curvature Dependent Triangulation of Implicit Surfaces, *IEEE Computer Graphics and Applications*, **22,** 2001.

[31] J. Lafontaine, An introduction to differentiable manifolds, Springer, 2015.

[32] J. Luukkainen, J. Väisälä, Elements of Lipschitz Topology, Annales Academire Scientiarum Fennicre, Series A. I. Mathematica, **3** (1977), 85-122.

[33] J. Nedelec, *Curved finite element methods for the solution of singular integral equations on surfaces in* $\mathbb{R}^3$*,* Comput. Methods Appl. Mech. Engng, **8** (1976), 61-80.

[34] M. Nica, *Eigenvalues and eigenfunctions of the Laplacian,* The Waterloo Mathematics Review, **1** (2011) 23-34.

[35] M. Ndjinga, **M. Nguemfouo**, *Well-posedness for the Poisson problem on closed Lipschitz manifolds*, Partial Differ. Equ. Appl., **4** (2023). https://doi.org/10.1007/s42985-023-00263-x.

[36] **M. Nguemfouo**, M. Ndjinga, *On the condition number of the finite element method for Laplace-Beltrami operator*, J. Elliptic. Parabol. Equ., (2023).https://doi.org/10.100/s41808-023-00251-7.

[37] M. A. Olshanskii, A. Reusken and J. Grande, *A finite element method for elliptic equations on surfaces,* SIAM J. Numer. Anal., **47** (2009), 3339-3358.

[38] L. Qin, S. Zhang and Z. Zhang, *Finite Element formulation in flat coordinate spaces to solve elliptic problems in general Riemmannian manifolds,* SIAM J. S ci. Comput., **36** (2014), 2149-2165.

[39] P. Reuss, *Précis de Neutronique*, EDP Sciences, Les Ulis Cedex A, France, 2003.

[40] A. Ribes, A. Bruneton, A. Geay, *SALOME: an Open-Source simulation platform integrating ParaView,* 10.13140/RG.2.2.12107.08485, 2017.

[41] Ribes, Andre, and Christian Caremoli. *Salome platform component model for numerical simulation* Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, IEEE., **2**, 2007.

[42] S. Rosenberg, *The Laplacian on a Riemannian manifold*, London mathematical society student texts, **31,** 1997.

[43] Y. Saad. *Iterative Methods for Sparse Linear Systems*.PWS Publishing Company, Boston, 1996.

[44] W. A. Strauss, *Partial differential equations*, New York: Wiley., 2008.

[45] M. Taylor, *Measure Theory and Integration,* Am. Math. Soc., Providence, RI, 2006.

[46] M. E. Taylor, *Partial differential equations III - Nonlinear equations*, Applied Mathematical Sciences, Springer-Verlag, New York, 1997.

[47] L. N. Trefethen and D. Bau, III *Numerical linear algebra*, Siam, Philadelphia, 1997.

[48] T.J. Willmore, *Riemannian Geometry*, Oxford University press Inc., New York, 1993.

[49] J. Wloka, Partial Differential equations, Cambridge University Press, Cambridge, 1987.

[50] C. Yaiza, *Analysis on Manifolds via the Laplacian*, Math 253, Fall, Harvard University, 2013.

[51] http://www.bcamath.org.

[52] http://www.salome-platform.org/downloads/current-version, accessed August 2017.

[53] http://cs.queensu.ca/ jstewart/papers/cga01.pdf accessed August 2019.

[54] https://github.com/ndjinga/CDMATH accessed August 2018.

[55] PETSc Conjugate Gradient method, https://petsc.org/release/docs/manualpages/KSP/KSPCG.html accessed August 2017.

[56] PETSc ILU preconditioner, https://petsc.org/release/docs/manualpages/PC/PCILU.html accessed August 2017.

[57] PETSc linear solver, https://petsc.org/release/docs/manualpages/KSP/KSPSolve.html accessed August 2017.

[58] PETSc matrix nullspace, https://petsc.org/release/docs/manualpages/Mat/MatSetNullSpace.html accessed August 2017.

[59] http://www.salome-platform.org/

[60] https://github.com/ndjinga/SOLVERLAB accessed August 2021.

♣ ANNEX: PUBLISHED PAPERS ♣

F FIRST PAPER

Marcial Nguemfouo, Michael Ndjinga: On the condition number of the finite element method for Laplace Beltrami operator. *J Elliptic Parabol Equ (2023).https://doi.org/10.100/s41808-023-00251-7*[36]

G SECOND PAPER

Michael Ndjinga, Marcial Nguemfouo: Well posedness for the Poisson problem on closed Lipschitz manifolds. *Partial Differ. Equ. Appl. 4, 44 (2023). https://doi.org/10.1007/s42985-023-00263-x*[35]

Journal of Elliptic and Parabolic Equations
https://doi.org/10.1007/s41808-023-00251-7

On the condition number of the finite element method for the Laplace–Beltrami operator

Marcial Nguemfouo[1] · **Michaël Ndjinga**[2]

Received: 24 December 2022 / Accepted: 9 October 2023

Abstract
We give an upper bound for the condition number of the finite element operator for the Laplace-Beltrami operator on closed surfaces immersed in $\mathbb{R}^3$. The expression is similar to the condition number of the Laplace operator in the Euclidean case, with the curvature affecting the condition number through the Poincaré constant. However in the case of closed surfaces the finite element matrix is singular and the linear system is solved for a unique solution with zero mean. As an application, numerical simulation of the Poisson problem on a sphere is presented in this paper, and motivate the search for efficient preconditioners.

Keywords Condition number · Laplace-Beltrami operator · Finite element method · Elliptic equation · closed surfaces

Mathematics Subject Classification 58J05 · 65F35 · 35J05 · 65N30

1 Introduction

The discretisation of surface partial differential equations using finite element methods is motivated by important applications related to physical and biological phenomena [12]. It is also used to answer theoretical questions in geometry [2] and to model complex systems on computers [14]. As is the case in the Euclidean context, the use

Marcial Nguemfouo and Michaël Ndjinga have contributed equally to this work.

Marcial Nguemfouo
marcial.nguemfouo@facsciences-uy1.cm

Michaël Ndjinga
michael.ndjinga@cea.fr

[1] Department of Mathematics, Faculty of Sciences, University of Yaounde 1, P.O. Box 812 Yaounde, Cameroon

[2] CEA-Saclay, Université Paris-Saclay, DES, ISAS, DM2S, STMF, 91191 Gif-sur-Yvette, France

Published online: 30 October 2023

of the finite element method on curved surfaces requires in practice the resolution of potentially large linear systems [8]. The precision and convergence of this resolution depends highly on the condition number of the finite element matrix [22].

The condition number of the finite element matrix has been extensively studied in the Euclidean case [10, 11]. However few works exist in the case of the Laplace-Beltrami operator on curved surfaces[6, 18]., particularly when they are closed.

Moreover, classical preconditioners are based on Incomplete LU factorisations (see section 10.3.1 in [22]), which fails if the matrix is not an M-matrix (see theorem 10.2 in [22]), in particular if the matrix is singular (see definition 1.4 in [22]). Yet, the discretisation of PDEs on closed surfaces yields singular matrices hence, the resolution of the associated linear systems is more technical. There is therefore an important need for *ad hoc* preconditioners for the numerical simulation of PDEs on closed surfaces (see,e.g., [17]). A first step in this project is to give an estimate of the condition number of the finite element matrix.

We are interested in the finite element approximation of the Poisson problem

$$-\Delta_\Gamma u = f \text{ on } \Gamma, \tag{1}$$

where $\Gamma \subset \mathbb{R}^3$ is a closed C^2 manifold and the weak solution u is sought for in $H^1(\Gamma)$. Γ being a closed manifold, u and f must have zero mean for the problem to admit a unique solution ($\int_\Gamma u = \int_\Gamma f = 0$).

[8] adapted the classical Euclidean finite element approach in order to deal with curved surfaces. Existence theorems and asymptotic error estimates are proven in [8, 9]. An important feature of the method is the avoidance of charts both in the problem formulation and the numerical method. The surface finite element method is based simply on triangulated surfaces and requires the geometry solely through knowledge of the vertices and normal vector of each triangle.

Following [8], we first approximate the domain Γ by the surface of a polyhedron Γ_h with triangular faces. The finite elements (i.e. the 3D triangles) are planar as in the Euclidean case, but they are no longer share the same normal vector. The right hand side f is approximated by a function $f_h \in L^2(\Gamma_h)$ with zero mean.

The solution u of (1) on Γ is then approximated by the solution u_h of the following Poisson problem on Γ_h:

$$-\Delta_{\Gamma_h} u_h = f_h \text{ on } \Gamma_h, \tag{2}$$

where $\Gamma_h \subset \mathbb{R}^3$ is a closed piecewise triangular surface and the weak solution u_h is sought for in $H^1(\Gamma_h)$. Γ_h being a closed manifold, u_h and f_h must have zero mean for (2) to admit a unique solution ($\int_{\Gamma_h} u_h = \int_{\Gamma_h} f_h = 0$).

The finite element method proposed in [8] then consists in approximating $u_h \in H^1(\Gamma_h)$ by its projection $\tilde{u}_h$ on $PL(\Gamma_h) \subset H^1(\Gamma_h)$, the subspace of continuous piecewise linear functions. Doing so, the partial differential operator Δ_{Γ_h} is approximated by a finite dimension linear operator with matrix $A_{\Delta_{\Gamma_h}}$. This approximation uses the variational formulation of (2) and reduces the Poisson problem (1) to the

resolution of a linear system

$$A_{\Delta\Gamma_h} X = b, \tag{3}$$

where X and b are vectors with zero mean.

The linear systems obtained using this technique are sparse and can be very large. The most practical way to solve such very large linear systems is to resort to an iterative method. Since the convergence rate of such methods is strongly affected by the condition number $\mathcal{K}_h(A_{\Delta\Gamma_h})$ of the finite element matrix $A_{\Delta\Gamma_h}$, studying the condition number $\mathcal{K}_h(A_{\Delta\Gamma_h})$ has both a theoretical and practical importance in the study of the finite element method. Well-conditioned systems have a smaller propagation of errors in $A_{\Delta\Gamma_h}$ or b, converge faster when solved iteratively, and have a smaller uncertainty when the error is estimated using the residual and ill-conditioned leads to slow convergence and an inaccurate solution; ill-conditioning can be remedied by a modification of the basis through a preconditioning matrix. In [17], J Maes and al., studied two Bramble-Pasciak-Xu-type preconditioners for second and fourth order elliptic problems on the surface of 2-sphere, using spherical triangulation and spherical basis functions. Kornhuber and Yserentant in [15] construct and analyzed multigrid method for discretized self-adjoint elliptic problems on triangular surfaces in $\mathbb{R}^3$. In [5] Bonito and Pasciak design and analyze certain natural multigrid algorithms for the Laplace-Beltrami operator on surface. Multigrid can be carried out on triangular surfaces in the same way as on planar triangulations. Our approach is based on the technique employed by Ern and Guermond [10] in the Euclidean frame to the curved frame.

One technical difficulty is that in the case of closed surfaces, the finite element matrix is not invertible on $\mathbb{R}^d$ since the domain has no boundary. The finite element operator is however invertible on the space of zero mean vectors. The other technical difficulty is the handling of different normal vectors arising from the non zero curvature. We are still able to adapt the Euclidean approach following [10] to derive an upper bound of the condition number.

The article is organised as follows. In Sect. 2 we recall the definition and properties of the Laplace-Beltrami operator. In Sect. 3 we recall the existence and uniqueness of solutions to the Poisson problem for the Laplace-Beltrami operator on closed surfaces. Section 4 is devoted to the finite element discretisation. Section 5 is concerned with the upper bound of the condition number of the finite element operator and Sect. 6 is devoted to numerical simulation of the Poisson problem on a sphere in order to illustrate the numerical method presented in this paper.

and motivate the search for efficient preconditioners;(see, e.g., [17] for more detail concerning the study of preconditioners on surfaces).

2 Calculus on hypersurfaces

In standard calculus, the gradient, divergence and Laplace operators are classically defined on the Euclidean space $\mathbb{R}^d$. They can however also be defined on a smooth manifold Γ in an intrinsic way (no immersion into $\mathbb{R}^d$) using a Riemannian metric (see

for example [13], section 1.2) or the Hodge operator (see [2] section 11.2). We chose instead for simplicity and pedagogy to define the differential operators on embedded manifolds $\Gamma \subset \mathbb{R}^3$ following [8] (Definition 1). This approach is also easier to handle from a practical point of view in numerical methods. For instance Lemma 1 is used in the determination of the finite element coefficient from each triangle (A).

The definition of differential operators on hypersurfaces can be performed without local charts, using rather the so-called Fermi coordinates [9]. The Fermi coordinates are also useful in the proof of the Poincaré inequality (see theorems 2.8 and 2.12 in [9]).

We define embedded surfaces in Sect. 2.1, the tangential gradient $\overrightarrow{\nabla}_\Gamma$ in Sect. 2.2, the tangential divergence $\overrightarrow{\nabla}_\Gamma \cdot$ in Sect. 2.3, and then the Laplace-Beltrami operator $\triangle_\Gamma$ as the composition of divergence and gradient in Sect. 2.4.

2.1 Embedded surfaces and Fermi coordinates

Following [9], we start with the definition of embedded surfaces and their normals.

Definition 1 (Embedded C^k-hypersurface)
$\Gamma \subset \mathbb{R}^3$ is called a C^k-hypersurface if for each point $x_0 \in \Gamma$, there exists an open set $U_{x_0} \subset \mathbb{R}^3$ containing x_0 and a function $\phi_{x_0} \in C^k(U_{x_0})$ with the following properties

$$\begin{aligned} &\vec{\nabla}\phi_{x_0} \neq 0 \text{ on } \Gamma \cap U_{x_0} \\ &\Gamma \cap U_{x_0} = \{x \in U_{x_0} | \phi_{x_0}(x) = 0\}. \end{aligned} \tag{4}$$

For any C^1-hypersurface $\Gamma \subset \mathbb{R}^3$, one can define the **unit normal** at any point $x \in U_{x_0}$ as

$$\vec{n}(x) = \frac{\overrightarrow{\nabla}\phi_{x_0}(x)}{\|\overrightarrow{\nabla}\phi_{x_0}(x)\|}, \tag{5}$$

where $\overrightarrow{\nabla}$ denotes the classical $\mathbb{R}^3$ gradient.

We define the δ**-strip around** Γ as

$$U_{\delta,\Gamma} = \{x \in \mathbb{R}^3, dist(x, \Gamma) < \delta\}. \tag{6}$$

For δ small enough it is possible to define a projection operator $a_\Gamma : U_\delta \to \Gamma$ onto Γ and a signed distance function $d_\Gamma : U_\delta \to \mathbb{R}$. $a_\Gamma(x)$ and $d_\Gamma(x)$ are called the Fermi coordinates of x and their existence is given by the following theorem.

Theorem 1 (Fermi coordinates) *Let Γ be an embedded C^2 hypersurface. There exists $\delta_{Fermi} > 0$ such that for every point $x \in U_{\delta_{Fermi}}$, there exists a unique point $a_\Gamma(x) \in \Gamma$ such that*

$$\forall x \in U_{\delta_{Fermi}}, \quad x = a_\Gamma(x) + d_\Gamma(x)\vec{n}(x). \tag{7}$$

where $d_\Gamma \in C^2(U_{\delta_{Fermi}})$ is the signed distance function.

Proof see Lemma 2.8 in [9]. □

In the following, the Fermi-coordinates will allow us to extend a function defined on Γ to a neighbourhood of Γ, which will prove useful in the definitions of the tangential gradient and the tangential divergence (Sects. 2.2 and 2.3).

2.2 The tangential gradient

The notions of hypersurface (Definition 1) and associated normal vectors (5) defined in the previous section are useful in the following definition of the projected gradient.

Definition 2 (Projected $\mathbb{R}^3$ gradient)
Let Γ be an embedded C^1 hypersurface, U_Γ a neighbourhood of Γ in $\mathbb{R}^3$, and $\bar{u} \in C^1(U_\Gamma)$. The projected gradient of $\bar{u}$ on Γ is

$$\overrightarrow{\nabla}_{P\Gamma}\bar{u} = \overrightarrow{\nabla}\bar{u} - (\vec{n} \cdot \overrightarrow{\nabla}\bar{u})\vec{n}. \tag{8}$$

From the projected gradient of a function $\bar{u} \in C^1(U_\Gamma)$ on a C^1 hypersurface Γ, one can define the tangential gradient of a function $u \in C^1(\Gamma)$ on a C^2 hypersurface Γ as follows.

Definition 3 (Tangential gradient) Let Γ be an embedded C^2 hypersurface, $u \in C^1(\Gamma)$. Let $\bar{u} \in C^1(U_{\delta_{Fermi},\Gamma})$ such that $\bar{u}|_\Gamma = u$. The tangential gradient of u is

$$\overrightarrow{\nabla}_\Gamma u = \overrightarrow{\nabla}_{P\Gamma}\bar{u}. \tag{9}$$

The assumption that Γ be C^2 in definition 5 comes from the fact that the existence of an extension $\bar{u}$ of u on a neighbourhood of Γ requires Fermi coordinates since $\bar{u}(x) = u(a_\Gamma(x))$.

The value of the tangential gradient of $u \in C^1(U_{\delta_{Fermi},\Gamma})$ on an embedded hypersurface Γ does not depend on the choice of $\bar{u}$. It depends only on the values taken by u on Γ as expressed in the following lemma (see [9] Lemma 2.4).

One key property of the tangential gradient is stated in Lemma 1. It will be useful in section A to calculate the coefficients of the finite element matrix.

Lemma 1 *The tangential gradient belongs to the tangent plane and is orthogonal to the normal vector*

$$(\overrightarrow{\nabla}_\Gamma u) \cdot \vec{n} = 0.$$

As for the classical gradient, there is a Poincaré's inequality involving the tangential gradient.

Theorem 2 (Poincaré's inequality) *Assume that Γ is an embedded C^3 hypersurface. There exists a constant c such that, for every function $f \in H^1(\Gamma)$ with $\int_\Gamma f = 0$, we have the inequality*

$$\|f - |\Gamma|^{-1}\int_\Gamma f\|_{L^2(\Gamma)} \leq c\|\nabla_\Gamma f\|_{L^2(\Gamma)}. \tag{10}$$

Proof see Theorem 2.12 in [9]. □

2.3 The tangential divergence

The tangential divergence operator on Γ is defined as the contribution to the full divergence arising from the tangent space to Γ. We first define the projected divergence on Γ as follows.

Definition 4 (Projected $\mathbb{R}^3$ divergence) Let Γ be an embedded C^1 hypersurface, U_Γ a neighbourhood of Γ in $\mathbb{R}^3$, and $\overline{\mathbf{V}} \in C^1(U_\Gamma)^3$ a vector field. The projected divergence of $\mathbf{V}$ is

$$\mathrm{div}_{P\Gamma}\overline{\mathbf{V}} = \overrightarrow{\nabla} \cdot \overline{\mathbf{V}} - {}^t\vec{n}(\overrightarrow{\nabla} \cdot \overline{\mathbf{V}})\vec{n}$$

From the projected divergence of a function $\bar{u} \in C^1(U_\Gamma)$, one can define the tangential divergence of a function $u \in C^1(\Gamma)$ as follows.

Definition 5 (Tangential divergence) Let Γ be an embedded C^2 hypersurface, $\mathbf{V} \in C^1(\Gamma)^3$ a vector field. Let $\overline{\mathbf{V}} \in C^1(U_{\delta_{Fermi},\Gamma})^3$ such that $\overline{\mathbf{V}}|_\Gamma = \mathbf{V}$. The tangential divergence of $\mathbf{V}$ is

$$\mathrm{div}_\Gamma \mathbf{V} = \mathrm{div}_{P\Gamma}\overline{\mathbf{V}}. \tag{11}$$

The assumption that Γ be C^2 in Definition 5 comes from the fact the existence of an extension $\overline{\mathbf{V}}$ of $\mathbf{V}$ on a neighbourhood of Γ requires Fermi coordinates since $\overline{\mathbf{V}}(x) = \overline{\mathbf{V}}(a_\Gamma(x))$.

2.4 The Laplace–Beltrami operator

Now that we have defined the tangential gradient and divergence in Sects. 2.2 and 2.3, the Laplace-Beltrami-operator on Γ, can be defined as the composition of the tangential divergence and gradient of a function $u \in C^2(\Gamma)$.

Definition 6 (Laplace-Beltrami operator) Let Γ be an embedded C^2 hypersurface. The Laplace-Beltrami operator applied to $u \in C^2(\Gamma)$ is

$$\Delta_\Gamma u = \mathrm{div}_\Gamma \overrightarrow{\nabla}_\Gamma u \tag{12}$$

For any smooth embedded hypersurface Γ, the following Green's formula is a consequence of the Stokes' theorem (see theorem 6.25 in [16]):

$$\forall v \in C^1(\Gamma), u \in C^2(\Gamma), \quad \int_\Gamma v\Delta_\Gamma u = -\int_\Gamma \overrightarrow{\nabla}_\Gamma u \cdot \overrightarrow{\nabla}_\Gamma v \tag{13}$$

A consequence of the Green's formula (13) is that the operator $-\Delta_\Gamma$ is symmetric and positive.

3 The Poisson problem on a closed surface

Now that we have defined the Laplace-Beltrami operator in Sect. 2.4, we can study the Poisson problem on closed hypersurfaces. We start by setting the problem and the relevant functional spaces in Sect. 3.1. We then give the weak formulation of the problem in Sect. 3.2. We end up by summarising the existence result in Sect. 3.3.

3.1 Definition and functional spaces

Let Γ be a closed C^2 hypersurface in $\mathbb{R}^3$. Since Γ is closed ($\partial\Gamma = \emptyset$), all the constant functions are in the kernel of Δ_Γ. Δ_Γ is therefore not invertible on the space of functions $u \in C^2(\Gamma)$.

We therefore have to impose the global condition $\int_\Gamma u = 0$ to guarantee the uniqueness of solutions. We define $L^2_0(\Gamma)$ (resp. $H^1_0(\Gamma)$) the space of measurable functions that are square integrable (resp. weakly differentiable with square integrable weak derivative) with zero mean on Γ.

$$L^2_0(\Gamma) = \left\{ f \in L^2(\Gamma), \int_\Gamma f = 0 \right\}, \quad H^1_0(\Gamma) = \left\{ f \in H^1(\Gamma), \int_\Gamma f = 0 \right\}. \tag{14}$$

Let $f \in L^2_0(\Gamma)$. $u \in C^2(\Gamma)$ is a **classical solution** of the Poisson problem provided that

$$-\Delta_\Gamma u = f \text{ on } \Gamma, \tag{15}$$

$$\int_\Gamma u = 0. \tag{16}$$

3.2 Weak form of the Poisson problem

The classical Laplace-Beltrami operator (Definition 6) acts on C^2 functions. A classical solution of (15) is therefore a function $u \in C^2(\Gamma)$. Unfortunately such a strong solution doesn't always exists even if f is assumed continuous (see [1] section 3.1.2 in the Euclidean case).

The variational formulation for (15) is the following:

$$\text{Find } u \in H^1_0(\Gamma) \text{ such that } \forall v \in H^1_0(\Gamma), \quad \int_\Gamma \overrightarrow{\nabla}_\Gamma u \cdot \overrightarrow{\nabla}_\Gamma v = \int_\Gamma f v. \tag{17}$$

We obtain (17) from (15) by applying the Green's formula (13). (17) implies (15) only if $u \in C^2(\Gamma)$. A solution $u \in H^1_0(\Gamma)$ of (17) is called **weak solution** for the Poisson problem. Indeed, it is only one time weakly differentiable whereas a strong (classical) solution is twice differentiable.

3.3 Existence result

The existence and uniqueness of weak solutions for the variational formulation (17) of problem (15) is a classical result for smooth manifolds Γ (see for instance [2] chapter 4 section 1.2). The following statement is taken from [8] Theorem 1 b).

Theorem 3 [Existence and uniqueness of weak solutions on a C^3 manifold] *Let Γ be a closed embedded C^3 hypersurface in $\mathbb{R}^3$. For every $f \in L^2(\Gamma)$ with $\int_\Gamma f = 0$, there exists a weak solution $u \in H_0^1(\Gamma)$ of $-\triangle u = f$ on Γ. Furthermore u is unique up to a constant.*

Proof see Theorem 1 b) in [8]. □

4 The finite element method

The finite element method (FEM), is a numerical method for solving problems of engineering and mathematical physics [1, 12]. We first approximate the closed hypersurface Γ by a closed polyhedral surface Γ_h with triangular faces. We approximate the right hand side function $f \in L_0^2(\Gamma)$ by a function $f_h \in L_0^2(\Gamma_h)$.

In Sect. 4.1, we look for $\tilde{u}_h \in PL_0(\Gamma_h)$ the projection of the solution $u_h \in H_0^1(\Gamma_h)$ of the Poisson problem $-\triangle_{\Gamma_h}\tilde{u}_h = f_h$, on the space of continuous piecewise linear functions with zero mean $PL_0(\Gamma_h)$.

The finite element matrix is symmetric and positive but not invertible (Sect. 4.2). However the finite element linear system admits a unique solution provided the right hand side has zero mean.

4.1 The finite element matrix

We consider a closed triangulated surface Γ_h having $n \in \mathbb{N}$ nodes. We define $PL_0(\Gamma_h) \subset H_0^1(\Gamma_h)$, the subspace of piecewise linear functions on Γ_h that have zero mean. Functions ϕ of $PL_0(\Gamma_h)$ are linear in the sense that they take the form $\phi(\vec{x}) = \alpha^\mathcal{T} x + \beta^\mathcal{T} y + \gamma^\mathcal{T} z + \zeta^\mathcal{T}$ on each of the triangles $\mathcal{T}$ composing Γ_h.

The discrete form of the variational formulation of the Poisson Eq. (15) is the following.

$$\text{Find } \tilde{u}_h \in PL_0(\Gamma_h) \text{ such that } \forall \tilde{v}_h \in PL_0(\Gamma_h), \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h}\tilde{u}_h \cdot \overrightarrow{\nabla}_{\Gamma_h}\tilde{v}_h = \int_{\Gamma_h} f_h\tilde{v}_h. \tag{18}$$

Let $\phi_i : \Gamma_h \to \mathbb{R}, \quad i = 1, ..., n$ be the standard piecewise linear basis functions associated with the nodes $\vec{x}_1, \ \vec{x}_2, \ \vec{x}_3, \ ..., \ \vec{x}_n$ such that $\phi_i(\vec{x}_j) = \delta_{ij}$. The solution $\tilde{u}_h$ of the discrete Poisson problem (18) must therefore satisfy the following system of equations

$$\forall i \in \{1, ..., n\}, \qquad \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h}\tilde{u}_h \cdot \overrightarrow{\nabla}_{\Gamma_h}\phi_i = \int_{\Gamma_h} f_h\phi_i, \tag{19}$$

which takes the algebraic form

$$A_{\triangle_{\Gamma_h}} X = b_h, \tag{20}$$

where the unknown vector $X = {}^t(u_1, ..., u_n)$ is the vector of components of $\tilde{u}_h$ on the nodal basis:

$$\tilde{u}_h = \sum_{i=1}^{n} u_i \phi_i. \quad \text{with} \sum_{i=1}^{n} u_i = 0 \tag{21}$$

The coefficients of the system matrix $A_{\triangle_{\Gamma_h}} = (a_{ij})_{i,j=1,...,n}$, and of the right hand side vector $b_h = {}^t(b_1, ..., b_n)$ are given by

$$a_{ij} = \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h} \phi_i \cdot \overrightarrow{\nabla}_{\Gamma_h} \phi_j, \quad b_j = \int_{\Gamma_h} f \phi_j. \tag{22}$$

Theorem 4 *(Properties of $A_{\triangle_{\Gamma_h}}$) Consider a polyhedral surface Γ_h. The finite element matrix $A_{\triangle_{\Gamma_h}}$ (Eq. 22) satisfies*

- *$A_{\triangle_{\Gamma_h}}$ is symmetric and positive*
- *$\ker A_{\triangle_{\Gamma_h}}$ is the set of constant functions*
- *$Im A_{\triangle_{\Gamma_h}}$ is the set of functions with zero mean*

$A_{\triangle_{\Gamma_h}}$ is not invertible since constants are in its kernel, hence the linear system (20) is singular. However it admits a unique solution with zero mean provided the right hand side has zero mean (see Theorem 5 in the next section).

4.2 Existence of a finite element approximation

Due to the absence of boundary, a technical difficulty in the numerical solution of linear systems arising from PDEs on closed surfaces is that the solution should be sought for in spaces of function with nil average.

First we note that since $A_{\triangle_{\Gamma_h}}$ is singular, we need to impose a condition on the right-hand side b_h for the solvability of the discrete system (20). Following the continuous setting (Theorem 3), we impose that the discrete right hand side function f_h must have zero mean:

$$\int_{\Gamma_h} f_h = \sum_{i=1}^{n} b_i \int_{\Gamma_h} \phi_i = 0. \tag{23}$$

With the latter condition on the right hand side, since $\triangle_{\Gamma_h}$ is an endomorphism of $PL_0(\Gamma_h)$, we can state the following existence theorem.

Theorem 5 *(Existence theorem for the linear system)*

Let $PL_0(\Gamma_h)$ be the space of piecewise linear finite elements on the discrete surface Γ_h. Let $f_h \in PL_0(\Gamma_h)$ with $\int_{\Gamma_h} f_h = 0$. Then there exists a unique discrete solution $\tilde{u}_h \in PL_0(\Gamma_h)$ to the discrete Poisson problem (18) with the property that $\int_{\Gamma_h} \tilde{u}_h = 0$.

4.3 Convergence of the numerical method

4.3.1 Fermi coordinates Lift operator

A function u defined on Γ can be extended to a neighborhood of Γ in $\mathbb{R}^3$ using a lift operator based on the Fermi coordinates around Γ.

Thanks to the **Fermi coordinates** defined in Theorem 1, we can define as in [9] (equation 4.2) a **lift operator** L such that

$$\begin{aligned} L : C(\Gamma_h) &\to C(\Gamma) \\ u_h &\to u_h \circ a_\Gamma^{-1}, \end{aligned} \tag{24}$$

provided

$$\Gamma_h \subset U_{\delta_{Fermi},\Gamma}. \tag{25}$$

4.3.2 Convergence theorems

In order to study the convergence of the finite element approximation, we need to compare $u \in H^1(\Gamma)$ with $\tilde{u}_h \in H^1(\Gamma_h)$ but don't share the same support. Hence we need to use the lift operator (24) which requires the assumption (25) that the triangulated surface Γ_h is close enough to Γ.

As the parameter h goes to zero the distance between Γ_h and Γ converges to zero as expressed in the following theorem taken from [9] Lemma 4.1.

Theorem 6 (Convergence of Γ_h towards Γ) *Let $\Gamma \in \mathbb{R}^3$ be an embedded C^2 hypersurface and $\Gamma_h \subset U_{\delta_{Fermi},\Gamma}$ a piecewise linear surface. Let h be the largest diameter of trianges in Γ_h. There exists a constant c such that*

$$\forall x \in \Gamma_h, \quad dist(x, \Gamma) \leq ch^2.$$

Once proven that Γ_h converges towards Γ, we can prove that $\tilde{u}_h$ converges to u using the lift operator (24). The following convergence theorem is taken from [8] Theorem 8, Lemma 6 and Lemma 7.

Theorem 7 (Convergence of $\tilde{u}_h$ towards u) *Let $\Gamma \in \mathbb{R}^3$ be an embedded C^2 hypersurface and $\Gamma_h \subset U_{\delta_{Fermi},\Gamma}$ a piecewise linear surface. Let h be the largest diameter of triangles in Γ_h.*

If u is a continuous solution of the Poisson problem (15) and $\tilde{u}_h$ is the discrete solution of (18), then there exists $c > 0$ such that

$$\|u - \tilde{u}_h \circ a_\Gamma^{-1}\|_{L^2(\Gamma)} \leq ch^2, \quad \|\nabla_\Gamma(u - \tilde{u}_h \circ a_\Gamma^{-1})\|_{L^2(\Gamma)} \leq ch. \tag{26}$$

5 Estimate of the condition number

The condition number of an invertible matrix A relative to the norm $\|\cdot\|$ is: $cond(A) = \|A\| \times \|A^{-1}\|$. The Euclidean norm is quite often used in applications and the expression of the L^2-condition number for a symmetric matrix is

$$cond_2(A) = \frac{\lambda_{\max}(|A|)}{\lambda_{\min}(|A|)}, \tag{27}$$

where $\lambda_{max}(|A|)$ (resp. $\lambda_{min}(|A|)$) is the largest (resp. smallest) eigenvalue of A in absolute value.

In the case of the finite element matrix $A_{\Delta_{\Gamma_h}}$, obtained from the discretisation of the Laplace-Beltrami operator on a closed piecewise triangular surface Γ_h (equation 22), one has to deal firstly with the fact that $A_{\Delta_{\Gamma_h}}$ is not invertible, secondly with the presence of a curvature field.

In the sequel, each triangle of Γ_h is denoted $\mathcal{T}$, its area is denoted $|\mathcal{T}|$ and its diameter is denoted

$$h_\mathcal{T} = diam(\mathcal{T}) = \max_{x,y\in\mathcal{T}} \|x - y\|. \tag{28}$$

As for the classical regular manifold, there is a Poincaré's inequality on Lipschitz manifold, we can state the following Lemma 2

Lemma 2 *(Poincaré's inequality on Lipschitz surface) Let Γ_h be a compact Lipschitz surface. There exists a constant $C(\Gamma_h)$ only depending on Γ_h such that*

$$\|v - |\Gamma_h|^{-1} \int_{\Gamma_h} v\|_{L_0^2(\Gamma_h)} \leq C(\Gamma_h)\|\nabla_{\Gamma_h} v\|_{L_0^2(\Gamma_h)} \quad \forall v \in H_0^1(\Gamma_h)$$

Proof see Lemma 2 in [4]. □

Theorem 8 (Condition number of the finite element matrix) *There exists a constant c such that for any closed piecewise triangular surface $\Gamma_h \subset \mathbb{R}^3$, the condition number of $A_{\Delta_{\Gamma_h}}$ (equation 22) satisfies*

$$\mathcal{K}_h(A_{\Delta_{\Gamma_h}}) \leq \frac{n \max_{\mathcal{T}\in\mathcal{T}_h} h_\mathcal{T}^2}{C(\Gamma_h)^{-1} c \min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2},$$

where $C(\Gamma_h)$ is the Poincaré's constant of Γ_h given by Lemma 2.

As is the case in the Euclidean context, the condition number of the finite element matrix for the Laplace-Beltrami operator is in $\mathcal{O}(h^{-2})$. The curvature of the surface affects the condition number through the Poincaré constant $C(\Gamma_h)$ from Lemma 2.

The proof of theorem 8 consists in five lemma adapting to the curved surfaces the steps followed in the Euclidean case (section 9.1.4 in [10]). We start with a lemma regarding the affine geometry of 3D triangles.

Lemma 3 *Let $A_0, B_0, C_0, A, B, C \in \mathbb{R}^3$ such that A_0, B_0, C_0 as well as A, B, C are not aligned.*

Let $\mathcal{T}_0$ (resp. $\mathcal{T}$) be the triangle formed by A_0, B_0 and C_0 (resp A, B and C). There exists an affine operator

$$\begin{aligned} T_{\mathcal{T}} : \mathcal{T} &\to \mathcal{T}_0 \\ x &\to J_{\mathcal{T}} x + b_{\mathcal{T}} \end{aligned}$$

where $J_{\mathcal{T}}$ is a 3×3 matrix and $b_{\mathcal{T}} \in \mathbb{R}^3$.

Proof Since A, B, C are not aligned, they define a plane $(\mathcal{P})$ with unit normal $\vec{n}$ and A, B, C is an affine frame of reference of $(\mathcal{P})$.

Since A_0, B_0, C_0 are not aligned, they define a plane $(\mathcal{P}_0)$ with unit normal $\vec{n}_0$ and A_0, B_0, C_0 is an affine frame of reference of $(\mathcal{P}_0)$.

Since $(\overrightarrow{AB}, \overrightarrow{AC}, \vec{n})$ and $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \vec{n}_0)$ form a basis of $\mathbb{R}^3$, $J_{\mathcal{T}}$ is defined as the transition matrix from the basis $(\overrightarrow{AB}, \overrightarrow{AC}, \vec{n})$ to the basis $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \vec{n}_0)$, and we have

$$\begin{aligned} J_{\mathcal{T}} \overrightarrow{AB} &= \overrightarrow{A_0B_0} \\ J_{\mathcal{T}} \overrightarrow{AC} &= \overrightarrow{A_0C_0} \\ J_{\mathcal{T}} \vec{n} &= \vec{n}_0. \end{aligned}$$

Defining

$$b_{\mathcal{T}} = A_0 - J_{\mathcal{T}} A,$$

the affine transformation $T_{\mathcal{T}}(x) = J_{\mathcal{T}} x + b_{\mathcal{T}}$ satisfies

$$\begin{aligned} T_{\mathcal{T}}(A) &= A_0 \\ T_{\mathcal{T}}(B) &= B_0 \\ T_{\mathcal{T}}(C) &= C_0. \end{aligned}$$

Let $P \in \mathcal{T}$, since $(\overrightarrow{AB}, \overrightarrow{AC})$ is a basis of $(\mathcal{P})$ there are coefficients $\alpha, \beta \in \mathbb{R}$ such that

$$P = A + \alpha \overrightarrow{AB} + \beta \overrightarrow{AC},$$

hence

$$T_{\mathcal{T}}(P) = T_{\mathcal{T}}(A) + \alpha J_{\mathcal{T}} \overrightarrow{AB} + \beta J_{\mathcal{T}} \overrightarrow{AC}$$

$$= A_0 + \alpha\overrightarrow{A_0B_0} + \beta\overrightarrow{A_0C_0}$$
$$= P_0 \in \mathcal{T}_0.$$

Hence $T_{\mathcal{T}} : \mathcal{T} \to \mathcal{T}_0$, and the theorem is proved. □

Lemma 4 *Under the hypotheses of Lemma 3, letting* $M_{\mathcal{T}} = [\overrightarrow{AB}, \overrightarrow{AC}, \vec{n}]$ *be the transition matrix from the canonical basis to the basis* $(\overrightarrow{AB}, \overrightarrow{AC}, \vec{n})$*, and* $M_{\mathcal{T}_0} = [\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n_0}]$ *be the transition matrix from the canonical basis to the basis* $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \vec{n}_0)$*,* $J_{\mathcal{T}}$ *takes the form*

$$J_{\mathcal{T}} = M_{\mathcal{T}_0} M_{\mathcal{T}}^{-1}$$

and furthermore

$$\det(J_{\mathcal{T}}) = \frac{|\mathcal{T}_0|}{|\mathcal{T}|}.$$

Proof Since $J_{\mathcal{T}}$ is the transition matrix from the basis $(\overrightarrow{AB}, \overrightarrow{AC}, \vec{n})$ to the basis $(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n_0})$, we have

$$J_{\mathcal{T}} M_{\mathcal{T}} = M_{\mathcal{T}_0}.$$

Since $M_{\mathcal{T}}$ is invertible, multiplying the previous relation by $M_{\mathcal{T}}^{-1}$ we obtain $J_{\mathcal{T}} = M_{\mathcal{T}_0} M_{\mathcal{T}}^{-1}$. Furthermore, taking the determinant yields

$$\det(J_{\mathcal{T}}) = \frac{|\mathcal{T}_0|}{|\mathcal{T}|},$$

since

$$\det(M_{\mathcal{T}}) = \det(\overrightarrow{AB}, \overrightarrow{AC}, \vec{n}) = (\overrightarrow{AB} \wedge \overrightarrow{AC}).\vec{n} = |\mathcal{T}|$$
$$\det(M_{\mathcal{T}_0}) = \det(\overrightarrow{A_0B_0}, \overrightarrow{A_0C_0}, \overrightarrow{n_0}) = (\overrightarrow{A_0B_0} \wedge \overrightarrow{A_0C_0}).\overrightarrow{n_0} = |\mathcal{T}_0|$$

□

Let $\mathcal{T}_0$ be a reference triangle in $\mathbb{R}^3$ with three non aligned vertices $\vec{x}_1, \vec{x}_2, \vec{x}_3$. In the proof of the main theorem, $\mathcal{T}_0$ will be chosen such that $|\mathcal{T}_0|^2 = \max_{\mathcal{T} \in \mathcal{T}_h} |\mathcal{T}|^2$.

The constant c in the main theorem is given by the following lemma.

Lemma 5 *Denote* ϕ_i^0*, the shape functions associated to each node of* $\mathcal{T}_0$*, and define the function*

$$f_0 : \mathbb{R}^n \to \mathbb{R}$$
$$(u_1, u_2, \cdots u_n) \to \int_{\mathcal{T}_0} |\sum_{i=1}^n u_i \phi_i^0|^2.$$

f_0 is a strictly positive quadratic form on $\mathbb{R}^n$.

Furthermore, there exist $c > 0$, $C > 0$ such that

$$\forall (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \quad c \sum_{i=1}^{n} u_i^2 \leq \int_{\mathcal{T}_0} |\sum_{i=1}^{n} u_i \phi_i^0|^2 \leq C \sum_{i=1}^{n} u_i^2.$$

Proof f_0 is a positive quadratic form since

$$\forall (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \quad \int_{\mathcal{T}_0} |\sum_{i=1}^{n} u_i \phi_i^0|^2 \geq 0.$$

First remark that:

$$\begin{aligned} f_0(U) = 0 &\Longleftrightarrow \int_{\mathcal{T}_0} |\sum_{i=1}^{n} u_i \phi_i^0|^2 = 0 \\ &\Longleftrightarrow |\sum_{i=1}^{n} u_i \phi_i^0|^2 = 0 \\ &\Longleftrightarrow \sum_{i=1}^{n} u_i \phi_i^0 = 0 \\ &\Longleftrightarrow \forall x \in \mathcal{T}_0, \sum_{i=1}^{n} u_i \phi_i^0(x) = 0. \end{aligned}$$

Since $\phi_i^0(x)$ are basis functions, we deduce that

$$f_0(U) = 0 \Leftrightarrow U = 0,$$

that is f_0 is a strictly positive quadratic form.

Secondly, we seek $c > 0$ and $C > 0$ such that

$$\forall U = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^n, \quad c \sum_{i=1}^{n} u_i^2 \leq f_0(U) \leq C \sum_{i=1}^{n} u_i^2.$$

Since f_0 is a strictly positive quadratic form on $\mathbb{R}^3$, its matrix M is symmetric and positive definite. M thus admits a minimum eigenvalue $\lambda_{\min} > 0$ and a maximum eigenvalue $\lambda_{\max} > 0$ (spectral theorem for symmetric matrices).

Since $f_0(U) = {}^tUMU$ we deduce

$$\forall U \in \mathbb{R}^n, \quad \lambda_{\min} \sum_{i=1}^{n} u_i^2 \leq f_0(U) \leq \lambda_{\max} \sum_{i=1}^{n} u_i^2.$$

Taking $c = \lambda_{\min} > 0$ and $C = \lambda_{\max} > 0$ we obtain the desired result. □

In the following two lemma (lemma 6 and Lemma 7) the lower bound and the upper bound of the following bilinear form a_h

$$a_h(\tilde{u}_h, \tilde{v}_h) = \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h} \tilde{u}_h \cdot \overrightarrow{\nabla}_{\Gamma_h} \tilde{v}_h, \tag{29}$$

on $PL_0(\Gamma_h)$ are given.

Lemma 6 *(Lower bound of a_h)*
We have

$$\forall \tilde{u}_h = \sum_{i=1}^{n} u_i \phi_i \in PL_0(\Gamma_h), \quad a_h(\tilde{u}_h, \tilde{u}_h) \geq C(\Gamma_h)^{-1} c n_c \frac{\min_{\mathcal{T} \in \mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_{i=1}^{n} u_i^2$$

Where $C(\Gamma_h)$ is a Poincaré's constant given by Lemma 2, $n(i)$ is the number of cells surrounding the node i and $n_c = \min_i n(i)$.

Proof Using Poincaré inequality we have

$$\begin{aligned} a_h(\tilde{u}_h, \tilde{u}_h) &\geq C(\Gamma_h)^{-1} \int_{\Gamma_h} |\tilde{u}_h|^2 \\ &\geq C(\Gamma_h)^{-1} \sum_{\mathcal{T} \in \mathcal{T}_h} \int_{\mathcal{T}} |\tilde{u}_h|^2. \end{aligned}$$

Let $N_{\mathcal{T}}$ denote the set of nodes surrounding the cell $\mathcal{T}$. Since $\phi_i(x) = 0$ when $x \in \mathcal{T}$ and $i \notin N_{\mathcal{T}}$, we have

$$a_h(\tilde{u}_h, \tilde{u}_h) \geq C(\Gamma_h)^{-1} \sum_{\mathcal{T} \in \mathcal{T}_h} \int_{\mathcal{T}} |\sum_{i \in N_{\mathcal{T}}} u_i \phi_i|^2.$$

Using a change of variable we obtain

$$a_h(\tilde{u}_h, \tilde{u}_h) \geq C(\Gamma_h)^{-1} \sum_{\mathcal{T} \in \mathcal{T}_h} |\det(J_{\mathcal{T}})|^{-2} \int_{\mathcal{T}_0} |\sum_{i \in N_{\mathcal{T}}} u_i \phi^0_{T_{\mathcal{T}}(i)}|^2,$$

where $T_{\mathcal{T}}(i)$ represents the node of $\mathcal{T}_0$ that is the image of the node i by $T_{\mathcal{T}}$. Using Lemma 5 we have

$$a_h(\tilde{u}_h, \tilde{u}_h) \geq C(\Gamma_h)^{-1} c \sum_{\mathcal{T} \in \mathcal{T}_h} |\det(J_{\mathcal{T}})|^{-2} \sum_{i \in N_{\mathcal{T}}} u_i^2.$$

Using Lemma 4 we have

$$a_h(\tilde{u}_h, \tilde{u}_h) \geq C(\Gamma_h)^{-1} c \sum_{\mathcal{T} \in \mathcal{T}_h} \frac{|\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_{i \in N_{\mathcal{T}}} u_i^2$$

$$\geq C(\Gamma_h)^{-1} c \frac{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_{\mathcal{T}\in\mathcal{T}_h} \sum_{i\in N_{\mathcal{T}}} u_i^2$$

$$\geq C(\Gamma_h)^{-1} c \frac{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_i n(i) u_i^2.$$

Hence we finally have

$$a_h(\tilde{u}_h, \tilde{u}_h) \geq C(\Gamma_h)^{-1} c n_c \frac{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2} \sum_i u_i^2,$$

where $n(i)$ is the number of cells surrounding the node i and

$$n_c = \min_i n(i).$$

□

Lemma 7 (Upper bound of a_h) *Under the hypotheses of Lemma 6, we have*

$$\forall \tilde{u}_h = \sum_{i=1}^{n} u_i \phi_i \in PL_0(\Gamma_h), \quad a_h(\tilde{u}_h, \tilde{u}_h) \leq n n_c \frac{h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|} \sum_i u_i^2.$$

Proof We decompose a_h over the mesh cells $\mathcal{T}_h$.

$$\begin{aligned} a_h(\tilde{u}_h, \tilde{u}_h) &= \sum_{\mathcal{T}\in\mathcal{T}_h} \int_{\mathcal{T}} |\overrightarrow{\nabla}_{\Gamma_h} \tilde{u}_h|^2 \\ &= \sum_{\mathcal{T}\in\mathcal{T}_h} \int_{\mathcal{T}} |\sum_{i\in N_{\mathcal{T}}} u_i \overrightarrow{\nabla}_{\Gamma_h} \phi_i(\vec{x})|^2 \\ &\leq n \sum_{\mathcal{T}\in\mathcal{T}_h} \sum_{i\in N_{\mathcal{T}}} u_i^2 \int_{\mathcal{T}} |\overrightarrow{\nabla}_{\Gamma_h} \phi_i(\vec{x})|^2. \end{aligned} \tag{30}$$

The next step consists in the explicit calculation of $\overrightarrow{\nabla}_{\Gamma_h} \phi_i(\vec{x})$ in each triangle $\mathcal{T}$. For more details we refer to A.

Given a triangle $\mathcal{T}$ having nodes $s_1^{\mathcal{T}}$, $s_2^{\mathcal{T}}$ and $s_3^{\mathcal{T}}$, the gradient of the nodal function associated to $s_1^{\mathcal{T}}$ is

$$\forall \vec{x} \in \mathcal{T}, \quad \overrightarrow{\nabla}_{\Gamma_h} \phi_{s_1^{\mathcal{T}}}(\vec{x}) = \frac{1}{-2|\mathcal{T}|} \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix},$$

where

$$a_1 = ((\vec{x}_{s_3^{\mathcal{T}}})_y - (\vec{x}_{s_2^{\mathcal{T}}})_y) n_z^{\mathcal{T}} + ((\vec{x}_{s_2^{\mathcal{T}}})_z - (\vec{x}_{s_3^{\mathcal{T}}})_z) n_y^{\mathcal{T}}$$

$$b_1 = ((\vec{x}_{s_2^{\mathcal{T}}})_x - (\vec{x}_{s_3^{\mathcal{T}}})_x)n_z^{\mathcal{T}} + ((\vec{x}_{s_3^{\mathcal{T}}})_z - (\vec{x}_{s_2^{\mathcal{T}}})_z)n_x^{\mathcal{T}}$$
$$c_1 = ((\vec{x}_{s_2^{\mathcal{T}}})_y - (\vec{x}_{s_3^{\mathcal{T}}})_y)n_x^{\mathcal{T}} + ((\vec{x}_{s_3^{\mathcal{T}}})_x - (\vec{x}_{s_2^{\mathcal{T}}})_x)n_y^{\mathcal{T}}.$$

Using the inequality $\forall r, s \in \mathbb{R}, (r+s)^2 \leq 2(r^2+s^2)$, we have:

$$\begin{aligned}
\|\overrightarrow{\nabla}_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\vec{x})\|_2 &\leq \frac{1}{2|\mathcal{T}|}\sqrt{a_1^2+b_1^2+c_1^2} \\
&\leq \frac{1}{2|\mathcal{T}|}\sqrt{(h_{\mathcal{T}}n_z^{\mathcal{T}}+h_{\mathcal{T}}n_y^{\mathcal{T}})^2+(h_{\mathcal{T}}n_x^{\mathcal{T}}+h_{\mathcal{T}}n_y^{\mathcal{T}})^2+(h_{\mathcal{T}}n_z^{\mathcal{T}}+h_{\mathcal{T}}n_x^{\mathcal{T}})^2} \\
&\leq \frac{1}{2|\mathcal{T}|}\sqrt{2((h_{\mathcal{T}}n_z^{\mathcal{T}})^2+(h_{\mathcal{T}}n_y^{\mathcal{T}})^2)+2((h_{\mathcal{T}}n_x^{\mathcal{T}})^2+(h_{\mathcal{T}}n_y^{\mathcal{T}})^2)+2((h_{\mathcal{T}}n_z^{\mathcal{T}})^2+(h_{\mathcal{T}}n_x^{\mathcal{T}})^2)} \\
&\leq \frac{1}{2|\mathcal{T}|}\sqrt{4((h_{\mathcal{T}}n_z^{\mathcal{T}})^2+(h_{\mathcal{T}}n_y^{\mathcal{T}})^2+(h_{\mathcal{T}}n_x^{\mathcal{T}})^2)} \\
&\leq \frac{1}{2|\mathcal{T}|}2h_{\mathcal{T}}\sqrt{(n_z^{\mathcal{T}})^2+(n_y^{\mathcal{T}})^2+(n_x^{\mathcal{T}})^2} \\
&\leq \frac{1}{2|\mathcal{T}|}2h_{\mathcal{T}}.
\end{aligned}$$

Hence

$$\|\overrightarrow{\nabla}_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\vec{x})\|_2 \leq \frac{h_{\mathcal{T}}}{|\mathcal{T}|}.$$

Using the same approach we obtain an estimate of the gradient of the two remaining nodal functions

$$\|\overrightarrow{\nabla}_{\Gamma_h}\phi_{s_2^{\mathcal{T}}}(\vec{x})\|_2 \leq \frac{h_{\mathcal{T}}}{|\mathcal{T}|}, \quad \|\overrightarrow{\nabla}_{\Gamma_h}\phi_{s_3^{\mathcal{T}}}(\vec{x})\|_2 \leq \frac{h_{\mathcal{T}}}{|\mathcal{T}|}.$$

Finally, for any shape function ϕ_i, we have

$$\forall i \in \{s_1^{\mathcal{T}}, s_2^{\mathcal{T}}, s_3^{\mathcal{T}}\}, \|\overrightarrow{\nabla}_{\Gamma_h}\phi_i(\vec{x})\| \leq \frac{h_{\mathcal{T}}}{|\mathcal{T}|}.$$

This allows us to deduce the final result from (30):

$$a_h(\tilde{u}_h, \tilde{u}_h) \leq nn_c \frac{\max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h}|\mathcal{T}|^2}\sum_i u_i^2. \tag{31}$$

□

Proof of the main theorem
Using Lemma 7, we have

$$\forall \tilde{u}_h \in PL_0(\Gamma_h), \qquad \frac{a_h(\tilde{u}_h, \tilde{u}_h)}{\sum_{i=1}^n u_i^2} = \frac{{}^tU_h A_{\Delta_{\Gamma_h}} U_h}{\|U_h\|^2} \leq nn_c \frac{\max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h}|\mathcal{T}|^2},$$

and we deduce an upper bound of the spectrum of the finite element operator on $PL_0(\Gamma_h)$

$$\lambda_{\max} \leq nn_c \frac{\max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}. \tag{32}$$

Using Lemma 6, we have

$$\forall \tilde{u}_h \in PL_0(\Gamma_h),\ \frac{a_h(\tilde{u}_h,\tilde{u}_h)}{\sum_{i=1}^n u_i^2} = \frac{{}^tU_h A_{\Delta\Gamma_h} U_h}{\|U_h\|^2} \geq C(\Gamma_h)^{-1} cn_c \frac{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2},$$

and we deduce a lower bound of the spectrum of the finite element operator on $PL_0(\Gamma_h)$

$$\lambda_{\min} \geq C(\Gamma_h)^{-1} cn_c \frac{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{|\mathcal{T}_0|^2}. \tag{33}$$

Using the definition of $\mathcal{K}_h(A_{\Delta\Gamma_h}) = \frac{\lambda_{\max}}{\lambda_{\min}}$, (33) and (32) yield the following inequality

$$\begin{aligned}\forall \mathcal{T}_0,\quad \mathcal{K}_h(A_{\Delta\Gamma_h}) &\leq \frac{\max_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{C(\Gamma_h)^{-1} cn_c |\mathcal{T}_0|^2} \times \frac{nn_c \max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|} \\ &\leq \frac{\max_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}{C(\Gamma_h)^{-1} c |\mathcal{T}_0|^2} \times \frac{n \max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{\min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|}.\end{aligned}$$

Choosing $\mathcal{T}_0$ such that $|\mathcal{T}_0|^2 = \max_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2$, we finally have

$$\mathcal{K}_h(A_{\Delta\Gamma_h}) \leq \frac{n \max_{\mathcal{T}\in\mathcal{T}_h} h_{\mathcal{T}}^2}{C(\Gamma_h)^{-1} c \min_{\mathcal{T}\in\mathcal{T}_h} |\mathcal{T}|^2}.$$

□

6 Some numerical results

In order to illustrate the numerical method presented in this paper, and motivate the search for efficient preconditioners, we present the numerical simulation of the Poisson problem on a sphere. The continuous problem is presented in Sect. 6.1. Then in Sect. 6.2 we present a sequence of refined meshes of the sphere. The results obtained from the finite element simulation of the Poisson problem on the mesh sequence are given in Sect. 6.3. We analyse these results in Sect. 6.4 and give the convergence curve (picture 4), iteration curve (picture 5), the residual curve (picture 6), and the condition number curve (picture 7).

For the design and meshing of the domain we use GEOMETRY and MESH modules of the software SALOME (see [21, 27]).

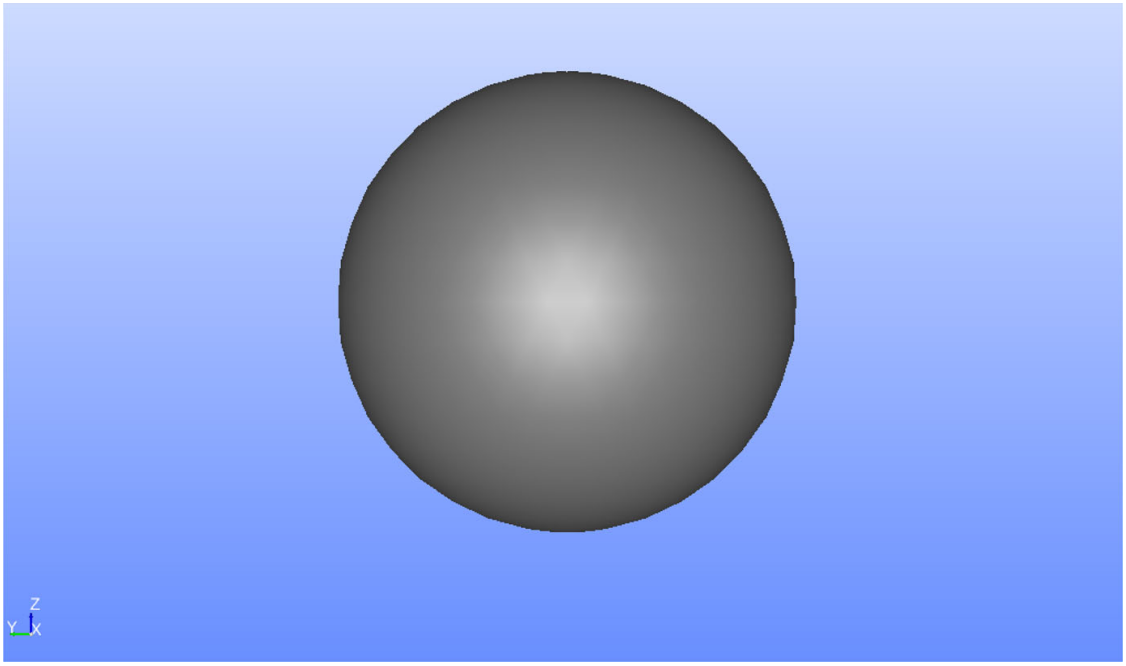


Fig. 1 The unit sphere

For the coding of the script, we use Python with the open-source Linux based library SOLVERLAB [28] which is very practical for the manipulation of large matrices, vectors, meshes and fields. It (SOLVERLAB) can handle finite element and finite volume discretizations, read general $1D$, $2D$ and $3D$ geometries and meshes generated by SALOME.
For the numerical resolution of our discrete problem, we use an iterative solver because the stiffness matrix $A_{\triangle_{\Gamma_h}}$ is large, sparse (see [22]) and singular. The library PETSc [3], encapsulated in SOLVERLAB, provides linear solvers for singular systems.
For the visualization of the result, we use the PARAVIS module included in SALOME (see [27]).

6.1 The Poisson problem on the unit sphere

We consider the unit sphere defined as follows

$$\Gamma_{sphere} = \{(x, y, z) \in \mathbb{R}^3, x^2 + y^2 + z^2 = 1\}.$$

The sphere is a C^∞ manifold of dimension 2 embedded in $\mathbb{R}^3$, hence the Laplace-Beltrami operator $\triangle_{\Gamma_{sphere}}$ is well defined on Γ_{sphere}.

Using the spherical coordinates (θ, ϕ) where θ is the longitude and ϕ the latitude, the Laplace-Beltrami operator takes the following form for any function $u \in C^2(\Gamma_{sphere})$:

$$\triangle_{\Gamma_{sphere}} u = \frac{1}{\sin(\phi)} \frac{\partial}{\partial \phi} \left(\sin \phi \frac{\partial u}{\partial \phi} \right) + \frac{1}{\sin(\phi)^2} \frac{\partial^2 u}{\partial \theta^2}.$$

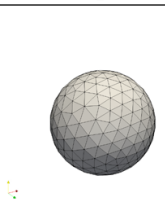
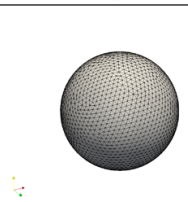
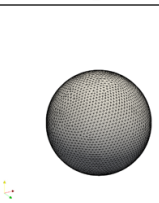
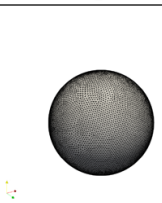
meshSphere 1	meshSphere 2	meshSphere 3	meshSphere 4
			
288 cells	2638 cells	4512 cells	10773 cells

Fig. 2 Mesh of domain

We consider the following **Poisson problem** on the sphere Γ_{sphere}

$$\begin{cases} -\Delta_{\Gamma_{sphere}} u = f_{sphere} \text{ on } \Gamma_{sphere} \\ \int_{\Gamma_{sphere}} u = 0 \end{cases}, \tag{34}$$

With the following choice for f_{sphere}:

$$f_{sphere}(x, y, z) = \frac{12}{(x^2 + y^2 + z^2)^{\frac{3}{2}}}(3x^2 y - y^3).$$

The exact solution u of (34) is given by (see [19]):

$$u_{sphere}(x, y, z) = \frac{1}{(x^2 + y^2 + z^2)^{\frac{3}{2}}}(3x^2 y - y^3) = \frac{1}{12} f_{sphere}.$$

One can check that f_{sphere} and u_{sphere} are **zero mean functions**.

Our objective is to solve numerically the Poisson problem (34) using the finite element method described in Sect. 4. We first build a sequence of refined meshes in Sect. 6.2. Then we run the simulation on each mesh. Some results are displayed in Sect. 6.3, and an analysis of the results is performed in Sect. 6.4.

6.2 Meshing of the sphere

In order to assess the Finite Element discretisation of the Poisson problem on the unit sphere, we build a sequence of refined meshes that will enable us to measure the convergence and computational time of the numerical method. The CAD model of the sphere (picture 1) was done with in the GEOMETRY module of the platform SALOME. For the design meshing of the sphere we use the MESH module of the platform SALOME (see [20, 21, 27]).

Below are screenshots of the meshes used in our convergence and computational time analysis.

The meshes are generated by a Delaunay type triangulation of the surface with all the nodes belonging to the sphere.

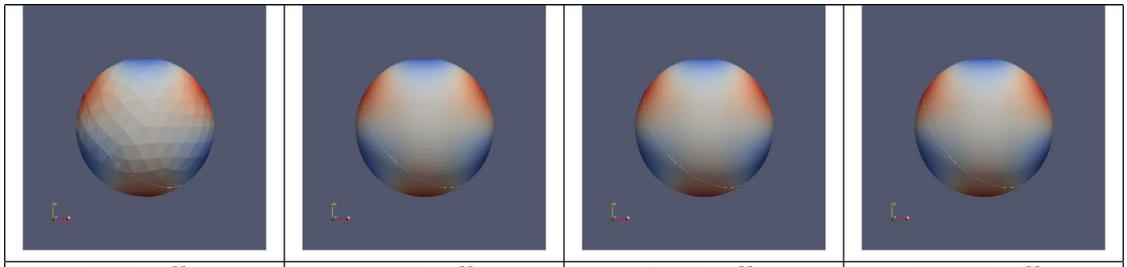


Fig. 3 Numerical results of the finite elements on the unit sphere

Using the SOLVERLAB python module, the right hand side of the Poisson problem (34) is interpolated on the meshes, and the rigidity matrices are filled with a spare matrix structured encapsulated from PETSc [3]. The linear systems are then solved using a conjugate gradient algorithm [22, 23] after the setting of a non zero nullspace [26].

6.3 Visualization of the results

For the numerical resolution of our discrete problem, we use an iterative solver because the stiffness matrix $A_{\triangle\Gamma_h}$ is large and sparse (see [22]).

For the visualization of the result, we use the PARAVIS module of the platform SALOME (see [27]).

Below are visualizations of the numerical results obtained on the different meshes of picture 2.

6.4 Discussion of the numerical results

Numerical convergence of the finite element method The picture 4 displays the evolution of the numerical error $\|u_h - u_{sphere}\|$ with the cell minimal diameter h, in logarithmic scale. The numerical error $\|u_h - u_{sphere}\|$ is taken as the supremum of $|u_h(x_i) - u_{sphere}(x_i)|$ over all the nodes x_i.

The theoretical error is composed of two contributions. The first is the interpolation error of u_{sphere} from the sphere Γ_{sphere} to a polyhedron Γ_h and is $\mathcal{O}(h^2)$. The second contribution is the approximation error coming from the finite element discretisation of the Poisson equation. This error is $\mathcal{O}(h^2)$. The total error is therefore $\mathcal{O}(h^2)$. See for example [7, Proposition 2.3] and [8, Theorem 8] for details concerning the errors.

We observe on picture 4 that the method converges with a numerical order of approximately 1.966, which is very close to the theoretical value of 2.

Number of CG iterations for the finite element method on the sphere It is not possible to use a direct solvers for the numerical resolution of the linear system $A_{\triangle\Gamma_h} X = b_h$, since they apply only to invertible matrices. We used instead the Conjugate Gradient (CG) [23] method with Incomplete LU factorisation [24] as preconditioner to solve our singular linear system thanks to PETSc algorithm [25, 26]. The picture 5 displays

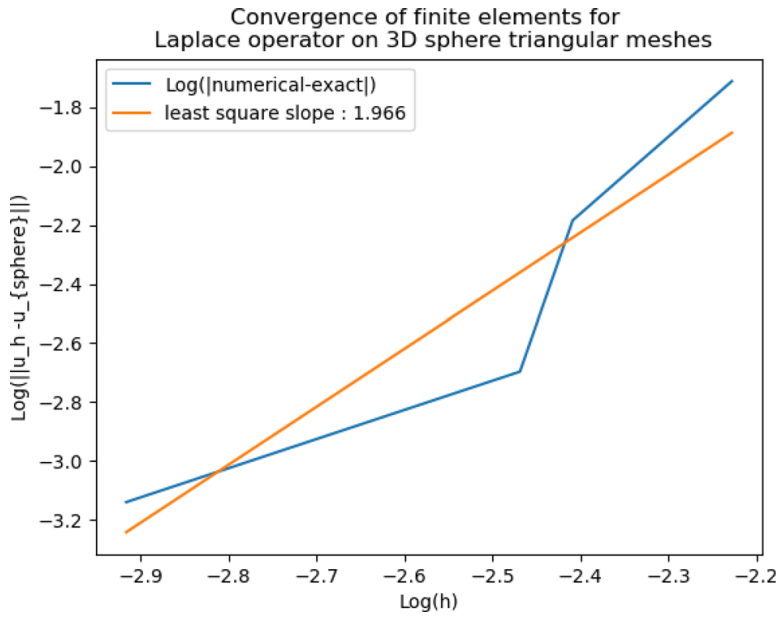


Fig. 4 Convergence of the finite element method on the sphere

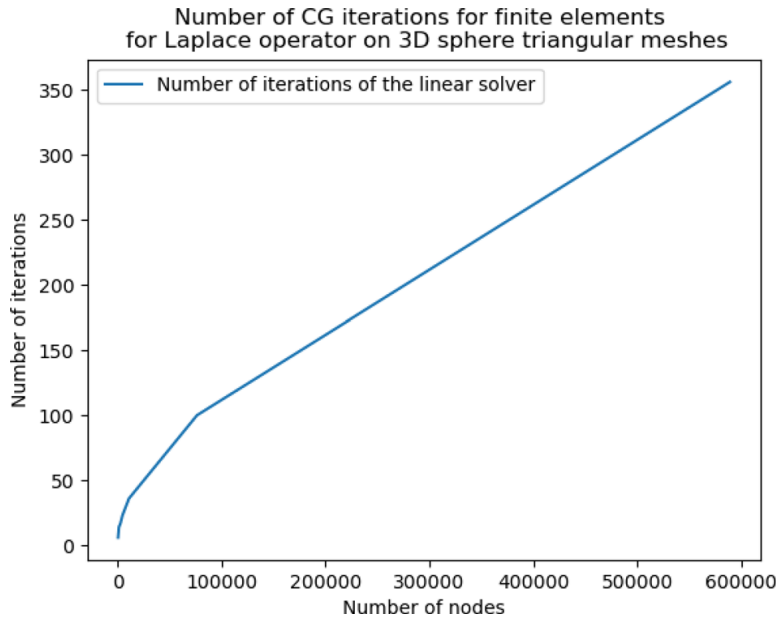


Fig. 5 Number of CG iterations for the finite element method on the sphere

the evolution of the number of CG iterations with the number of nodes. The number of iterations increases linearly with the number of nodes of the mesh.

CG Residual for the finite elements methods on the sphere

The picture 6 displays the evolution in Logarithmic scale of the residual $\epsilon_h = \|A_{\triangle\Gamma_h}X - b_h\|$ with the number of nodes in the mesh $\mathcal{T}_h$. This residual ϵ_h evolves from about 10^{-2} to about 10^{-5} as the number of nodes evolves from 288 to 600000. This is because we had to adapt the precision of the linear solver to the number of nodes. Indeed the matrix of the linear system $A_{\triangle\Gamma_h}X = b_h$ is singular and thus b_h should belong to the range of $A_{\triangle\Gamma_h}$ up to machine precision. The theoretical range of $A_{\triangle\Gamma_h}$ consists in vectors of zero mean, but at the computer level zero becomes

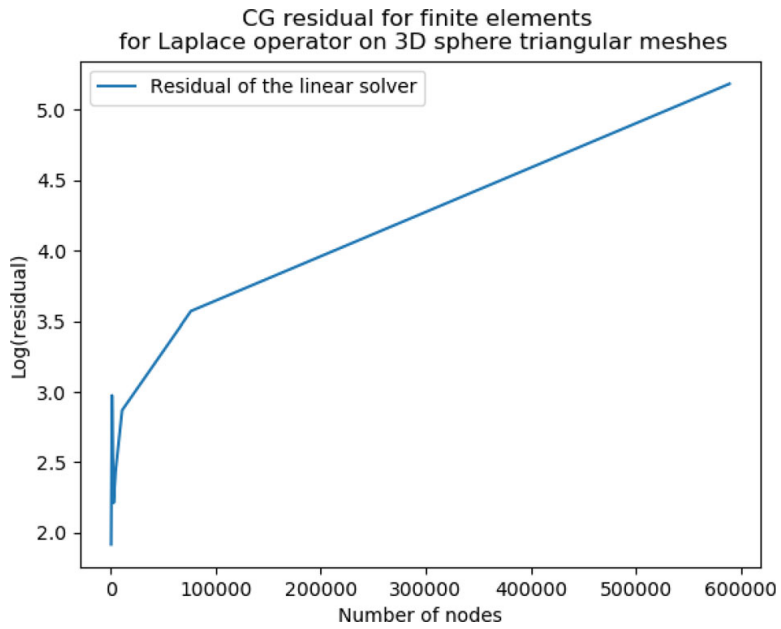


Fig. 6 CG residual for the finite element method on the sphere

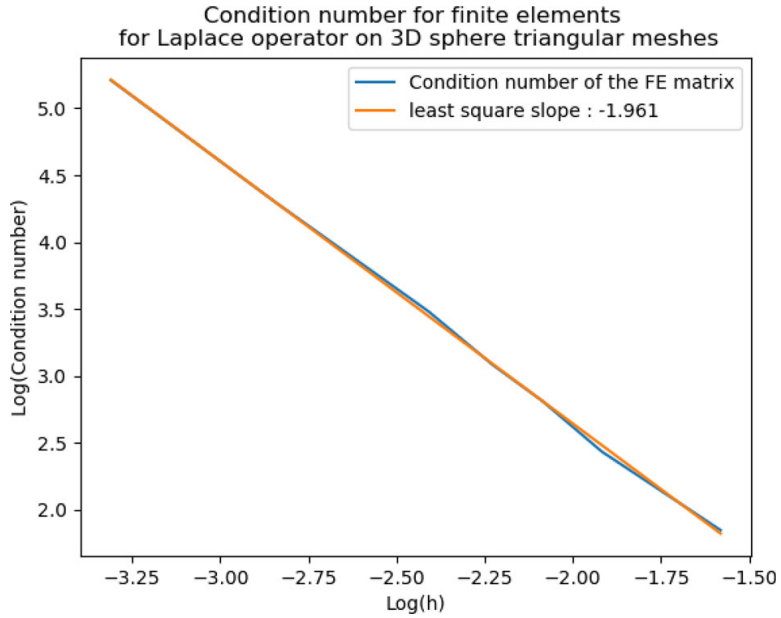


Fig. 7 Condition number for the finite element method on the sphere

machine precision and thus b_h should have mean lower than machine precision. If the machine precision is too small (say 10^{-10}) then the linear solver will fail when the integral of b_h is larger than 10^{-10}.

Condition number of the finite element matrix on the sphere

The picture 7 displays the evolution of the condition number with the cell minimal diameter h, in logarithmic scale. Using the SOLVERLAB python module, the rigidity matrices are filled with a spare matrix structured encapsulated from PETSc [3]. We extracted the diameter h on each mesh, then from the rigidity matrices we run the simulation to obtain the eigenvalues (Krylov-Schur method) and then the condition number (CondNumber = Rigidite.getConditionNumber(,)).

We observe first that the condition number increase as $h \to \infty$. Furthermore, the condition number grows as h to the power -1.962, which is very close to the theoretical value of 2 given in our main result in Theorem 8.

7 Conclusion and perspectives

We have seen that the properties of the finite element matrix on 3D surfaces are very similar to those in Euclidean spaces. The main difference when the surface is closed is that the matrix is not invertible, which makes the proofs more technical. However, adapting the technique used in the Euclidean context, we gave an upper bound for the condition number in $\mathcal{O}(h^{-2})$. Using a similar techniques, it is possible to derive a lower bound of the condition number following for instance [10] in the Euclidean case.

In Sect. 6, we have given some numerical results of the simulation of the Poisson problem on a sphere. The results showed that the discretisation converges with scheme order of approximately 1.966 and that the condition number increases nearly as $\mathcal{O}(h^{-2})$. zero). Theoretical estimation corroborated by numerical evaluations of condition number from the finite element matrix spectrum, showing that our upper bound is practical.

The numerical simulations performed in Sect. 6 showed the number of iterations of the linear solver increases linearly with the mesh size. This yields a sharp increase of the computational time as the number of nodes increases. Our simulation used the Incomplete LU factorisations as preconditioner but more advanced techniques such as multigrid perform better in the Euclidean context. Their adaptation to the curved would be based on a fine analysis of the finite element matrix taking into account its singularity. This work can therefore be seen as a first step in the design and analysis of advanced preconditioners for singular systems, particularly those arising from the discretisation of PDEs on closed surfaces.

Declarations

Some journals require declarations to be submitted in a standardised format. Please check the Instructions for Authors of the journal to which you are submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

Acknowledgements We would like to thank Reviewers for taking the necessary time and effort to review the manuscript. We sincerely appreciate all your valuable comments and suggestions, which helped us in improving the quality of the manuscript. We would like to take this opportunity to thank you for the effort and expertise that you contributed towards reviewing the article, without which it would be impossible to maintain the high standards of peer-reviewed journals

Availability of data and materials Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Code Availability Code is available in the corresponding author.

Declarations

Conflict of interest (check journal-specific guidelines for which heading to use) The authors declare no conflict of interest regarding the publication of this paper

Appendix A Coefficients of the finite element matrix

We compute the coefficients of the finite element matrix (22) on a closed piecewise triangular surface Γ_h. The mesh $\mathcal{T}_h$ of the domain Γ_h is composed of triangular elements $(\mathcal{T}_k)_{k\geq 1}$ having non zero area. The n vertices of Γ_h are denoted $\vec{x}_1,\ \vec{x}_2,\ \vec{x}_3,\ \ldots,\ \vec{x}_n$. To each vertex $\vec{x}_i$, we associate a nodal function, $\phi_i : \Gamma_h \to \mathbb{R}$ such that $\phi_i(x_j) = \delta_{ij}$.

We observe that in $3D$, functions $\phi \in PL_0(\Gamma_h)$ take the following form on each triangle $\mathcal{T} \in \mathcal{T}_h$

$$\phi(\vec{x}) = \alpha^{\mathcal{T}} x + \beta^{\mathcal{T}} y + \gamma^{\mathcal{T}} z + \zeta^{\mathcal{T}}. \tag{A1}$$

The gradient of any function $\phi \in PL_0(\Gamma_h)$ is thus constant on each triangle $\mathcal{T} \in \mathcal{T}_h$ and its components can be deduced from (A1) and (9):

$$\forall \vec{x} \in \mathcal{T}, \quad \overrightarrow{\nabla}_{\Gamma_h}\phi(\vec{x}) = \left(\alpha^{\mathcal{T}}, \beta^{\mathcal{T}}, \gamma^{\mathcal{T}}\right). \tag{A2}$$

To determine the gradient of a function $\phi \in PL_0(\Gamma_h)$ on a triangle $\mathcal{T}$, we first remark that according to Lemma 1, we have:

$$n_x^{\mathcal{T}}\alpha^{\mathcal{T}} + n_y^{\mathcal{T}}\beta^{\mathcal{T}} + n_z^{\mathcal{T}}\gamma^{\mathcal{T}} = 0. \tag{A3}$$

Secondly, we take the values of ϕ at each node of $\mathcal{T}$. Denoting $s_1^{\mathcal{T}}$, $s_2^{\mathcal{T}}$ and $s_3^{\mathcal{T}}$ the three nodes of $\mathcal{T}$, we find that $\alpha^{\mathcal{T}}$, $\beta^{\mathcal{T}}$, $\gamma^{\mathcal{T}}$ and $\zeta^{\mathcal{T}}$ are solutions of the following system:

$$\begin{pmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & (\vec{x}_{s_1^{\mathcal{T}}})_y & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & (\vec{x}_{s_2^{\mathcal{T}}})_y & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & (\vec{x}_{s_3^{\mathcal{T}}})_y & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \end{pmatrix} \begin{pmatrix} \alpha^{\mathcal{T}} \\ \beta^{\mathcal{T}} \\ \gamma^{\mathcal{T}} \\ \zeta^{\mathcal{T}} \end{pmatrix} = \begin{pmatrix} \phi(\vec{x}_{s_1^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_2^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_3^{\mathcal{T}}}) \end{pmatrix}. \tag{A4}$$

From (A3) and (A4) we obtain the following system with four unknowns $\alpha^{\mathcal{T}}$, $\beta^{\mathcal{T}}$, $\gamma^{\mathcal{T}}$ and $\zeta^{\mathcal{T}}$

$$\begin{pmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & (\vec{x}_{s_1^{\mathcal{T}}})_y & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & (\vec{x}_{s_2^{\mathcal{T}}})_y & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & (\vec{x}_{s_3^{\mathcal{T}}})_y & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ n_x^k & n_y^k & n_z^k & 0 \end{pmatrix} \begin{pmatrix} \alpha^{\mathcal{T}} \\ \beta^{\mathcal{T}} \\ \gamma^{\mathcal{T}} \\ \zeta^{\mathcal{T}} \end{pmatrix} = \begin{pmatrix} \phi(\vec{x}_{s_1^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_2^{\mathcal{T}}}) \\ \phi(\vec{x}_{s_3^{\mathcal{T}}}) \\ 0 \end{pmatrix}. \tag{A5}$$

The determinant of this system is equal to $-2|\mathcal{T}|$, where $|\mathcal{T}|$ is the area of the triangle $\mathcal{T}$ which is assumed to be nonzero.

We use the Cramer formulae to express the solution of (A5):

$$\overrightarrow{\nabla}_{\Gamma_h}\phi(\vec{x})|_{\mathcal{T}} = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} \phi(\vec{x}_{s_1^{\mathcal{T}}}) & (\vec{x}_{s_1^{\mathcal{T}}})_y & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ \phi(\vec{x}_{s_2^{\mathcal{T}}}) & (\vec{x}_{s_2^{\mathcal{T}}})_y & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ \phi(\vec{x}_{s_3^{\mathcal{T}}}) & (\vec{x}_{s_3^{\mathcal{T}}})_y & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & \phi(\vec{x}_{s_1^{\mathcal{T}}}) & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & \phi(\vec{x}_{s_2^{\mathcal{T}}}) & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & \phi(\vec{x}_{s_3^{\mathcal{T}}}) & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & (\vec{x}_{s_1^{\mathcal{T}}})_y & \phi(\vec{x}_{s_1^{\mathcal{T}}}) & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & (\vec{x}_{s_2^{\mathcal{T}}})_y & \phi(\vec{x}_{s_2^{\mathcal{T}}}) & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & (\vec{x}_{s_3^{\mathcal{T}}})_y & \phi(\vec{x}_{s_3^{\mathcal{T}}}) & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}$$

We deduce the gradient of the nodal shape functions $\phi_{s_1^{\mathcal{T}}}$, $\phi_{s_2^{\mathcal{T}}}$ and $\phi_{s_3^{\mathcal{T}}}$:

$$\overrightarrow{\nabla}_{\Gamma_h}\phi_{s_1^{\mathcal{T}}}(\vec{x}) = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} 1 & (\vec{x}_{s_1^{\mathcal{T}}})_y & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ 0 & (\vec{x}_{s_2^{\mathcal{T}}})_y & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ 0 & (\vec{x}_{s_3^{\mathcal{T}}})_y & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & 1 & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & 0 & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & 0 & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & (\vec{x}_{s_1^{\mathcal{T}}})_y & 1 & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & (\vec{x}_{s_2^{\mathcal{T}}})_y & 0 & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & (\vec{x}_{s_3^{\mathcal{T}}})_y & 0 & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}$$

$$\overrightarrow{\nabla}_{\Gamma_h}\phi_{s_2^{\mathcal{T}}}(\vec{x}) = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} 0 & (\vec{x}_{s_1^{\mathcal{T}}})_y & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ 1 & (\vec{x}_{s_2^{\mathcal{T}}})_y & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ 0 & (\vec{x}_{s_3^{\mathcal{T}}})_y & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & 0 & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & 1 & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & 0 & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & (\vec{x}_{s_1^{\mathcal{T}}})_y & 0 & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & (\vec{x}_{s_2^{\mathcal{T}}})_y & 1 & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & (\vec{x}_{s_3^{\mathcal{T}}})_y & 0 & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}$$

$$\overrightarrow{\nabla}_{\Gamma_h}\phi_{s_3^{\mathcal{T}}}(\vec{x}) = \frac{1}{-2|\mathcal{T}|}\begin{pmatrix} \begin{vmatrix} 0 & (\vec{x}_{s_1^{\mathcal{T}}})_y & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ 0 & (\vec{x}_{s_2^{\mathcal{T}}})_y & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ 1 & (\vec{x}_{s_3^{\mathcal{T}}})_y & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ 0 & n_y^{\mathcal{T}} & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & 0 & (\vec{x}_{s_1^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & 0 & (\vec{x}_{s_2^{\mathcal{T}}})_z & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & 1 & (\vec{x}_{s_3^{\mathcal{T}}})_z & 1 \\ n_x^{\mathcal{T}} & 0 & n_z^{\mathcal{T}} & 0 \end{vmatrix} \\ \begin{vmatrix} (\vec{x}_{s_1^{\mathcal{T}}})_x & (\vec{x}_{s_1^{\mathcal{T}}})_y & 0 & 1 \\ (\vec{x}_{s_2^{\mathcal{T}}})_x & (\vec{x}_{s_2^{\mathcal{T}}})_y & 0 & 1 \\ (\vec{x}_{s_3^{\mathcal{T}}})_x & (\vec{x}_{s_3^{\mathcal{T}}})_y & 1 & 1 \\ n_x^{\mathcal{T}} & n_y^{\mathcal{T}} & 0 & 0 \end{vmatrix} \end{pmatrix}.$$

References

1. Allaire, G.: Numerical Analysis and Optimization - An Introduction to Mathematical Modeling and Numerical Simulation. Oxford University Press, Oxford (2007)
2. Aubin, T.: Some Nonlinear Problems in Riemannian Geometry. Springer, New York (1998)
3. Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W., Karpeyev, D., Kaushik, D., Knepley, M., May, D., Curfman McInnes, L., Mills, R., Munson, T., Rupp, K., Sanan, P., Smith, B., Zampini, S., Zhang, H.: PETSc/TAO Users Manual, Argonne National Laboratory, ANL-21/39 - Revision 3.17, (2021). https://petsc.org/release/docs/manual/manual.pdf
4. Bonito, A., Demlow, A., Nochetto, R. H.: Finite element methods for the Laplace-Beltrami operator,(2019). https://arxiv.org/pdf/1906.02786.pdf
5. Bonito, A., Pasciak, J.E.: Convergence analysis of variational and non-variational multi-grid algorithms for the Laplace-Beltrami operator. Math. Comp. **81**, 1263–1288 (2012). https://doi.org/10.1090/S0025-5718-2011-02551-2
6. Burman, E., Hansbo,P., Larson, M. G., Massing, A.: Cut Finite Element Methods for Partial Differential Equations on Embedded Manifolds of Arbitrary Codimensions. https://doi.org/10.48550/arXiv.1610.01660

7. Demlow, A.: Higher-order finite element methods and pointwise error estimates for elliptic problems on surfaces. SIAM J. Numer. Anal. **47**(2), 805–827 (2009). https://doi.org/10.1137/070708135
8. Dziuk, G.: Finite elements for the Beltrami operator on arbitrary surfaces. *in Partial differential equations and calculus of variations, S. Hildebrandt and R. Leis, eds., vol. 1357 of Lecture Notes in Mathematics, Springer*, pp. 142–155 (1988)
9. Dziuk, G., Elliott, C. M.: Finite element methods for surface PDEs, *Acta Numerica*, pp. 289–396 (2013)
10. Ern, A., Guermond, J-L.: Theory and practice of finite elements. Vol. **159**. Springer, New York (2013)
11. Ern, A., Guermond, J-L.: Evaluation for the condition number in linear system arising in finite element approximations, ESAIM: Mathematical Modelling and Numerical Analysis, Vol. **40**, issue 1 (2006)
12. Elliott, C.M., Stinner, B.: Computation of two-phase biomembranes with phase dependent material parameters using surface finite elements. Commun. Comput. Phys. **13**, 325–360 (2010)
13. Hebey, E.: Nonlinear analysis on manifolds ; Sobolev spaces and inequalities, *Courant Lecture Notes* Vol.**5** (2000)
14. Hildebrandt, K., Polthier, K.: On approximation of the Laplace-Beltrami operator and the Willmore energy of surfaces, *Eurographics Symposium on Geometry Processing***30** (2011)
15. Kornhuber, R., Yserentant, H.: Multigrid methods for discrete elliptic problems on triangular surfaces. Comput. Vis. Sci. **11**, 251–257 (2008). https://doi.org/10.1007/s00791-008-0102-4
16. Lafontaine, J.: An Introduction to Differentiable Manifolds. Springer, New York (2015)
17. Maes, J., Kunoth, A., Bultheel, A.: BPX-type preconditioners for second and fourth order elliptic problems on the sphere. SIAM J. Numer. Anal. **45**, 206–222 (2007). https://doi.org/10.1137/050647414
18. Olshanskii, M.A., Reusken, A.: A finite element method for surface PDEs: matrix properties. Numer. Math. **114**(3), 491–520 (2010)
19. Olshanskii, M.A., Reusken, A., Grande, J.: A finite element method for elliptic equations on surfaces. SIAM J. Numer. Anal. **47**, 3339–3358 (2009)
20. Ribes, A., Bruneton, A., Geay, A.: SALOME: an Open-Source simulation platform integrating ParaView (2017). https://doi.org/10.13140/RG.2.2.12107.08485
21. Ribes, A., Caremoli, C.: Salome platform component model for numerical simulation. Computer Software and Applications Conference, 2007. COMPSAC (2007). 31st Annual International. Vol. 2. IEEE(2007)
22. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing Company, Boston (1996)
23. PETSc Conjugate Gradient method, https://petsc.org/release/docs/manualpages/KSP/KSPCG.html
24. PETSc ILU preconditioner, https://petsc.org/release/docs/manualpages/PC/PCILU.html
25. PETSc linear solver, https://petsc.org/release/docs/manualpages/KSP/KSPSolve.html
26. PETSc matrix nullspace, https://petsc.org/release/docs/manualpages/Mat/MatSetNullSpace.html
27. http://www.salome-platform.org/
28. https://github.com/ndjinga/SOLVERLAB

Partial Differential Equations and Applications (2023) 4:44
https://doi.org/10.1007/s42985-023-00263-x

ORIGINAL PAPER

Well posedness for the Poisson problem on closed Lipschitz manifolds

Michaël Ndjinga[1] · Marcial Nguemfouo[2]

Received: 23 August 2022 / Accepted: 6 September 2023

Abstract
We study the weak formulation of the Poisson problem on closed Lipschitz manifolds. Lipschitz manifolds do not admit tangent spaces everywhere and the definition of the Laplace–Beltrami operator is more technical than on classical differentiable manifolds (see, e.g., Gesztesy in J Math Sci 172:279–346, 2011). They however arise naturally after the triangulation of a smooth surface for computer vision or simulation purposes. We derive Stokes' and Green's theorems as well as a Poincaré's inequality on Lipschitz manifolds. The existence and uniqueness of weak solutions of the Poisson problem are given in this new framework for both the continuous and discrete problems. As an example of application, numerical results are given for the Poisson problem on the boundary of the unit cube.

Keywords Lipschitz manifold · Laplace–Beltrami operator · Finite element method · Elliptic equation · Closed surfaces

Mathematics Subject Classification 58J05 · 35J05 · 65N30

1 Introduction

Traditionally in the context of the finite element simulation of PDEs on surfaces [see, e.g., 1–4], a continuous surface Γ is first triangulated into a piecewise linear surface Γ_h that is no longer differentiable but merely a Lipschitz manifold. The solution u of a linear PDE on Γ can then be approximated by the solution u_h of the same PDE on Γ_h. Using a variational

This article is part of the section "Computational Approaches" edited by Siddhartha Mishra.

Marcial Nguemfouo
marcial.nguemfouo@facsciences-uy1.cm

Michaël Ndjinga
michael.ndjinga@cea.fr

1 CEA-Saclay, Université Paris-Saclay, DES, ISAS, DM2S, STMF, 91191 Gif-sur-Yvette, France

2 Department of Mathematics, Faculty of Sciences, University of Yaounde 1, P.O. Box 812 Yaoundé, Cameroon

Published online: 21 September 2023

formulation, one then seek for $\tilde{u}_h$ the projection of u_h on the finite dimensional space generated by the nodal functions of Γ_h. The numerical resolution of the resulting linear system yields the nodal values of $\tilde{u}_h$.

In [1, 2], the well-posedness of the Poisson problem $-\triangle_\Gamma u = f$ on a smooth surface Γ immersed in $\mathbb{R}^d$ is studied, and a finite element approximation is proposed. In the context of immersed surfaces, [2] defines differential operators (tangential gradient and divergence) which are based on Fermi coordinates. Fermi coordinates [see Section 2.3 in 2] are global coordinates on a hypersurface $\Gamma \in \mathbb{R}^d$ that avoid the use of local charts. Unfortunately, the existence of Fermi coordinates requires Γ to be C^2, or more precisely to satisfy both an interior and an exterior sphere condition. The issue of polyhedral surfaces is raised in [1] (page 3), which quotes [5] for the definition of Sobolev spaces on $C^{0,1}$ manifolds. However, [5] deal with $C^{2,\kappa}$ manifolds (see Definition 2.10 and Section 4.2). The assumption of C^2 regularity made on the surface Γ in [1, 2] restricts the scope of surfaces Γ for which the methodology yields the well-posedness of the continuous problem. Furthermore, even if Γ is assumed C^2, after triangulation of Γ, the new surface Γ_h is piecewise linear, and no longer C^2. The study and characterisation of u_h the solution of the Poisson problem on Γ_h can not therefore be performed using the methodology in [1, 2] due to the lack of smoothness of Γ_h.

In order to perform calculus on more general surfaces that are not C^2, especially on triangulated surfaces, we can not rely on the approach of [1, 2] using Fermi coordinates on C^2 immersed surfaces. We chose instead to extend the definition of the classical differential operators to Lipschitz manifolds following [6] (Sect. 3), [7] (Appendix A) and [8].

Lipschitz manifolds are an intermediate structure between topological manifolds where the tangent space is defined nowhere and C^1 manifolds where the tangent space is defined everywhere. On Lipschitz manifolds the tangent space is defined almost everywhere, which is enough to define spaces of functions that are weakly differentiable. Gesztesy et al. [7] defines Lebesgue L^p spaces, as well as Sobolev $W^{1,p}$ and $W^{-1,p}$ spaces. The main technical difficulty is that Lipschitz manifolds are not differentiable everywhere. For that reason, they make an intensive use of differentiability almost everywhere through the Rademacher's theorem. We use this new setting for the definition and study of the Poisson problem on a Lipschitz manifold, and propose a finite element method for its numerical simulation.

The article is organised as follows. In Sect. 2, following [7], we recall the definition of Lipschitz manifolds, and the associated differential and exterior calculi. We go on introducing the Sobolev spaces as well as the Laplace–Beltrami operator on a Lipschitz manifold. We extend [7] by proving Stokes' 1 and Green's 2 theorems as well as a Poincaré's inequality on Lipschitz manifolds. In Sect. 3, in the context of closed Lipschitz manifold we study the Poisson problem and give a theorem of existence and uniqueness of solutions. This theorem relies on the weak formulation of the problem that is available thanks to the Green's theorem. Section 4 is devoted to the introduction of the linear finite element discretization and Sect. 5 is concerned with the numerical simulation of the Poisson problem on a Lipschitz manifold: the boundary of the unit cube.

2 Calculus on Lipschitz manifolds

In standard calculus, the gradient, divergence and Laplace operators are classically defined on the Euclidean space $\mathbb{R}^d$. They can however also be defined on a C^1 manifold $\mathcal{M}$ in an intrinsic way (no immersion into $\mathbb{R}^d$), using a Riemannian metric [see for example Section 1.2 in 9] or the Hodge operator [see Section 11.2 in 10].

Following [7], we recall the definition of Lipschitz manifolds in Sect. 2.1. Lipschitz differentiable forms are defined in Sect. 2.3. Sobolev spaces on Lipschitz manifolds are introduced in Sect. 2.4, and the Laplace–Beltrami operator in Sect. 2.5.

2.1 Lipschitz manifolds

We recall that a **topological manifold** $\mathcal{M}$ of dimension $d \in \mathbb{N}^*$ is a topological space such that for every $x \in \mathcal{M}$ there exists an open set $U \subset \mathcal{M}$ and a map $\phi : U \to \mathbb{R}^d$ such that $x \in U$ and ϕ is an homeomorphism onto its image $\phi(U)$. The couple (U, ϕ), where $U \in \mathcal{M}$ is an open set around x and $\phi : U \to \phi(U) \subset \mathbb{R}^d$ is an homeomorphism, is called a **local coordinate chart**. An atlas on $\mathcal{M}$ is a countable family $\{U_i, \phi_i\}_{i\in I}$ such that $\mathcal{M} = \cup_{i\in I} U_i$ and (U_i, ϕ_i) is a local coordinate chart for each $i \in I$.

We recall that a map $\phi : E \to F$ between two metric spaces E and F is a **bilipschitz map** if it is a Lipschitz homeomorphism onto its image $\phi(E)$, and its inverse is also Lipschitz.

Definition 1 (*Lipschitz manifold*) A **Lipschitz manifold** is a topological manifold with an atlas (called a **Lipschitz atlas**) $\{U_i, \phi_i\}_{i\in I}$ such that for any $i, j \in I$ the transition map $\phi_i \circ \phi_j^{-1}$ is bilipschitz from $\phi_j(U_i \cap U_j)$ to $\phi_i(U_i \cap U_j)$.

Since differentiability will occur only almost everywhere, functional analysis on Lipschitz manifolds requires the definition of negligible sets.

Definition 2 (*Negligible sets*) Let Γ be a Lipschitz manifold and $\{U_i, \phi_i\}_{i\in I}$ its Lipschitz atlas.
A set $S \subset \Gamma$ is negligible provided $\phi_i(U_i \cap S)$ has measure zero in $\mathbb{R}^d$.

A property that is true on all but a negligible set of points of Γ is said to hold almost everywhere.

Given a Lipschitz manifold Γ with a Lipschitz atlas $\{U_i, \phi_i\}_{i\in I}$, a point $x \in \Gamma$ is said to be a **singular point** if there exists $i, j \in I$ such that $x \in U_i \cap U_j$ and $\phi_i \circ \phi_j^{-1}$ is not differentiable at $\phi_j(x)$. A point x that is not singular is called a **regular point**.

A consequence of the Rademacher's theorem [see Corollary 11.7 in 11] is that **the set of singular points is negligible**. Hence for any Lipschitz manifold, one can properly define a tangent space only almost everywhere (see Definition 5 below). Note that if the Lipschitz manifold is embedded in $\mathbb{R}^d$, it admits almost everywhere a normal vector.

Unlike smooth manifolds where differentiability can take place everywhere and be of any order, differentiability on Lipschitz manifolds can happen only at regular points and is necessarily of order one.

Definition 3 (*Differentiable map*) Let Γ be a Lipschitz manifold. A map $f : \Gamma \to \mathbb{R}$ is said to be differentiable at $x \in \Gamma$ provided

- x is a regular point of Γ
- there exists a local chart (U, ϕ) such that $x \in U$ and $f \circ \phi^{-1} : \phi(U) \to \mathbb{R}$ is differentiable at $\phi(x)$

Definition 4 (*Measurable and Lipschitz maps*) Let Γ be a Lipschitz manifold, $m \in \mathbb{N}^*$. A map $f : \Gamma \to \mathbb{R}^m$ is measurable (resp. Lipschitz) if for any local coordinate chart (U, ϕ), the map $f \circ \phi^{-1} : \phi(U) \to \mathbb{R}^m$ is measurable (resp. Lipschitz).

Lipschitz maps are differentiable almost everywhere thanks to the Rademacher's theorem.

2.2 Tangent space

Let $x \in \Gamma$ be a regular point. A **path through** x is a continuous map $\gamma_x :]-\epsilon, \epsilon[\to \Gamma, \epsilon > 0$ with $\gamma_x(0) = x$ and such that there exists $i \in I$, $x \in U_i$ and $\phi_i \circ \gamma_x$ is differentiable at 0.
We define the following equivalence relation between paths through x: γ_{1x} and γ_{2x} are equivalent if they share the same derivative at x:

$$\gamma_{1x} \equiv_x \gamma_{2x} \Leftrightarrow (\phi_i \circ \gamma_{1x})'(0) = (\phi_i \circ \gamma_{2x})'(0). \tag{1}$$

Following [7] appendix A and [6] Section 3.1, we define the tangent space almost everywhere as the space of equivalence classes $\dot{\gamma_x}$ for the relation (1).

Definition 5 (*Tangent space*)
Let Γ be a Lipschitz oriented manifold. The tangent space of Γ at $x \in \Gamma$ is the vector space

$$\begin{aligned} T_x\Gamma &= \{\dot{\gamma_x}, \gamma_x \text{ a path through } x\} \quad \text{if } x \text{ is a regular point} \\ T_x\Gamma &= \{0\} \qquad\qquad\qquad\qquad\quad \text{if } x \text{ is a singular point} \end{aligned} \tag{2}$$

If Γ is a Lipschitz oriented manifold of dimension $d \in \mathbb{N}^*$, its tangent space at every regular point is a vector space of dimension d. At every regular point $x \in \Gamma$ belonging to a local chart $(U, \phi = (\phi^1, \ldots, \phi^d))$, we can define a basis of the tangent space $T_x\Gamma$ by considering the d curves $\phi_x^j(t) = \phi_i^{-1}(t_1^x, \ldots, t_{j-1}^x, t_j^x + t, t_{j+1}^x, \ldots, t_d^x)$ where $(t_1^x, \ldots, t_{j-1}^x, t_j^x + t, t_{j+1}^x, \ldots, t_d^x) = \phi_i^{-1}(x)$. The curves $t \to \phi^j(t)$ form d independent classes for the relation (1). Hence $\dot{\phi_x^j}$ form a basis of $T_x\Gamma$.

Definition 6 (*Strong gradient map*) Let Γ be a Lipschitz manifold, $x \in \Gamma$ a regular point and $f : \Gamma \to \mathbb{R}$ be differentiable at x.
The linear tangent map of f at x denoted $Grad_x f$ is the map sending the equivalence class $\dot{\gamma}$ of the relation $\equiv_x$ to the equivalence class $\dot{f \circ \gamma}$ of the relation $\equiv_{f(x)}$:

$$Grad_x f : T_x(\Gamma) \to \mathbb{R} \tag{3}$$

$$\frac{\mathrm{d}}{\mathrm{d}\gamma_x} \to \frac{\mathrm{d}}{\mathrm{d}t}(f \circ \gamma)(t)|_{t=0}. \tag{4}$$

On a Lipschitz manifold of dimension d, if $x \in \Gamma$ is a regular point, $T_x\Gamma$ is a real vector space of dimension d and basis $\dot{\phi_x^1}, \ldots \dot{\phi_x^d}$. Following a classical duality argument, the strong gradient of f at x, which is a linear form, can therefore be represented by a d-dimensional real vector denoted $\overrightarrow{\nabla}_x f$ such that

$$Grad_x f\left(\sum_{i=1}^{d} a_i \dot{\phi_x^i}\right) = (a_1, \ldots, a_d) \cdot \overrightarrow{\nabla}_x f.$$

2.3 Measurable and Lipschitz differential forms

On any vector space E of dimension d, we introduce for any integer $l \in \{0, 1, \ldots, d\}$ the l-th exterior power $\Lambda^l E$ as the vector space of l-linear alternate forms on E.
Because of the lack of differentiability of Lipschitz manifolds, we cannot work with smooth differential forms as done classically. Instead, following [6] (Section 3.2), we introduce measurable and Lipschitz differential forms as follows.

Definition 7 (*Measurable and Lipschitz differential forms*)
Let $p \in [1, \infty]$, $l \in \{0, 1, \ldots, d\}$, Γ be a closed Lipschitz manifold of dimension d, and $\{U_i, \phi_i\}_{i \in I}$ its Lipschitz atlas.

A measurable (resp. Lipschitz) differential form ω of degree l is a mapping defined almost everywhere on Γ such that

- $\omega \in \Lambda^l T_x \Gamma$ for almost every $x \in \Gamma$
- for any local chart $(U, \phi = (\phi^1, \ldots, \phi^d))$ and any multi-index J of length l, there exists a measurable (resp. Lipschitz) function a_J such that almost everywhere

$$\omega = \sum_{|J|=l} a_J \mathrm{d}\phi^J. \tag{5}$$

Due to the lack of regularity, we cannot define the exterior derivative as an endomorphism on differential forms as done classically [see, e.g., Theorem 5.42 in 12]. Instead, the derivative of a Lipschitz differentiable form is a measurable differentiable form defined as follows.

Definition 8 (*Exterior derivative of a Lipschitz differentiable form*)
Let Γ be a Lipschitz manifold of dimension d, and $\{U_i, \phi_i\}_{i \in I}$ its Lipschitz atlas. Let $l \in \{0, 1, \ldots, d\}$, and ω a Lipschitz differential form of degree l on Γ. The exterior derivative of ω is the measurable differential form $\mathrm{d}\omega$ of degree $l + 1$ such that on any local chart $(U, \phi = (\phi^1, \ldots, \phi^d))$

$$\mathrm{d}\omega = \sum_{|J|=l} \sum_{j \in J} \frac{\partial a_J}{\partial \phi^j} \mathrm{d}\phi^j \wedge \mathrm{d}\phi^J, \tag{6}$$

where a_J are the Lipschitz coefficients of ω: $\omega = \sum_{|J|=l} a_J \mathrm{d}\phi^J$.

In Definition 8, the differentiability of a Lipschitz form follows from the Rademacher's theorem.

The orientability of a manifold is a necessary condition for the existence of a volume form [see, e.g., Theorem 6.5 in 12], which in turns yields a measure and integration theory on the manifold. The classical definition of orientability involves the positiveness everywhere of the Jacobian of the transition maps $\phi_i \circ \phi_j^{-1}$. However, in the case of Lipschitz manifolds, the transition maps are differentiable only almost everywhere.

Definition 9 (*Orientation*) A Lipschitz manifold Γ with Lipschitz atlas $\{U_i, \phi_i\}_{i \in I}$ is oriented provided the change of coordinates $\phi_i \circ \phi_j^{-1}$ has positive jacobian on $U_i \cap U_j$ almost everywhere.

An orientation is therefore given by an atlas such that the coordinate charts $(U_i, \phi_i = (\phi_{i1}, \ldots, \phi_{id}))$ define a positively oriented basis $\left(\frac{\mathrm{d}}{\mathrm{d}\phi_{i1}}, \ldots, \frac{\mathrm{d}}{\mathrm{d}\phi_{id}}\right)$ of the tangent space almost everywhere. On any oriented Lipschitz manifold Γ of dimension d, there exists a Lipschitz differentiable form $\mathrm{d}V_\Gamma(x) \in \Lambda^d T_x \Gamma$ such that $\mathrm{d}V_\Gamma(x) \neq 0$ for almost every $x \in \Gamma$. The proof is a direct adaptation of Theorem 6.5 in [12]. Such a form is called a **volume form** [see, e.g., Definition 6.3 in 12] and takes the form $\mathrm{d}V_\Gamma = f \mathrm{d}\phi^1 \wedge \cdots \wedge \mathrm{d}\phi^d$, where f is measurable.

Integrating volume forms requires the pullback operator. Following the definition 5.18 in [12] we define the pullback of a volume form by a Lipschitz map.

Definition 10 (*Pullback of a volume form*)
Let Γ be a Lipschitz manifold of dimension $d \in \mathbb{N}^*$, U an open subset of $\mathbb{R}^d$ and let $g : U \to \Gamma$ be a Lipschitz map.

The pullback of a measurable differentiable form ω on Γ by g is the measurable differentiable form $g^*\omega$ on $\mathbb{R}^d$ defined by

$$(g^*\omega)_x(v_1,\dots,v_q) = \omega_{g(x)}(\nabla_{g(x)}g\cdot v_1,\dots,\nabla_{g(x)}g\cdot v_q) \quad a.e.\ x\in U,\ \forall v_1,\dots,v_q\in\mathbb{R}^d. \tag{7}$$

Note that the pullback of a Lipschitz differential form by a Lipschitz map is merely a measurable differential form (not Lipschitz). For instance, the pullback of a volume form $\mathrm{d}V_\Gamma = f\mathrm{d}\phi^1\wedge\cdots\wedge\mathrm{d}\phi^d$ by a Lipschitz map g is the form $(g^*\omega)_x = f\circ g(x)\det(\nabla_{g(x)}g)\mathrm{d}x^1\wedge\cdots\wedge\mathrm{d}x^d$ [see, e.g., Proposition 6.12 in 12], which is not necessarily Lipschitz.

On $\mathbb{R}^d$, volume forms take the form $\mathrm{d}V_{\mathbb{R}^d}(x) = f(x)\mathrm{d}x^1\cdots\mathrm{d}x^d$ and their integral is simply $\int_{\mathbb{R}^d}\mathrm{d}V_{\mathbb{R}^d} = \int_{\mathbb{R}^d} f(x)\mathrm{d}x^1\dots\mathrm{d}x^d$. On a general Lipschitz manifold, the **integral of a volume form** [see, e.g., Definition 6.16 in 12], is defined by

$$\int_\Gamma f\mathrm{d}\lambda_\Gamma = \sum_{i\in I}\int_{\phi_i(U_i)}(\phi_i^{-1})^*(\theta_i f\mathrm{d}V_\Gamma), \tag{8}$$

where $(\theta_i)_{i\in I}$ is a Lipschitz partition of unity on Γ, subordinate to the cover $(U_i)_{i\in I}$: $supp(\theta_i)\subset U_i$, $0\le\theta_i\le 1$ and $\sum_{i\in I}\theta_i = 1$ [see Proposition 6.14 in 12].
The value of the integral (8) does not depend on the choice of coordinate charts nor the partition of unity [see the proof of Theorem 6.15 in 12].

We are now ready to state Stokes' theorem for Lipschitz forms, which is required for the integration by part (see the proof of Green's theorem 2).

Theorem 1 (Stokes' theorem) *Let Γ be a compact oriented Lipschitz manifold of dimension d. Let ω be a Lipschitz form of degree $d-1$ on Γ. Then*

$$\int_\Gamma \mathrm{d}\omega = 0. \tag{9}$$

Proof Since Γ is compact, there is a finite set I' and a finite number of charts $(U_i,\phi_i)_{i\in I'}$ that covers Γ: $\Gamma = \cup_{i\in I'}U_i$. Let $(\alpha_i)_{i\in I'}$ be a smooth partition of unity subordinate to that cover. The existence of $(\alpha_i)_{i\in I'}$ is given by Proposition 6.14 in [12]. Since $\omega = \sum_{i\in I'}\alpha_i\omega$, it is sufficient to prove the theorem for $\alpha_i\omega$ that is, for a Lipschitz differentiable form supported within an open subset U_i.

We assume therefore in the following that $supp(\omega)\subset U_i$ where $(U_i,\phi_i = (\phi_i^1,\dots,\phi_i^d))$ is a local chart. Since ω is a Lipschitz differentiable form of degree $d-1$, for any $j\in\{1,2,\dots,d\}$, there exists a Lipschitz function a_j supported in U_i such that almost everywhere

$$\omega(x) = \sum_{j=1}^d a_j(x)\mathrm{d}\phi^1\wedge\mathrm{d}\phi^2\wedge\cdots\wedge\widehat{\mathrm{d}\phi^j}\wedge\cdots\wedge\mathrm{d}\phi^d,\quad x\in\Gamma. \tag{10}$$

Since ω is Lipschitz, it admits an exterior derivative which is

$$\mathrm{d}\omega(x) = \sum_{j=1}^d\frac{\partial a_j}{\partial\phi^j}(x)\,\mathrm{d}\phi^j\wedge\mathrm{d}\phi^1\wedge\mathrm{d}\phi^2\wedge\cdots\wedge\widehat{\mathrm{d}\phi^j}\wedge\cdots\wedge\mathrm{d}\phi^d \tag{11}$$

$$= \left(\sum_{j=1}^d(-1)^{j-1}\frac{\partial a_j}{\partial\phi^j}(x)\right)\mathrm{d}\phi^1\wedge\mathrm{d}\phi^2\wedge\cdots\wedge\mathrm{d}\phi^d, \tag{12}$$

From the definition of the tangent map (Definition 6) we have

$$\frac{\partial a_j}{\partial\phi^j}(\phi_i^{-1}(x)) = \frac{\partial a_j\circ\phi_i^{-1}}{\partial x^j}(x). \tag{13}$$

The pullback of the exterior derivative is

$$(\phi_i^{-1})^*(\mathrm{d}\omega) = \left(\sum_{j=1}^{d}(-1)^{j-1}\frac{\partial a_j}{\partial \phi^j}\right)(\phi_i^{-1}(x))\,(\phi_i^{-1})^*(\mathrm{d}\phi^1)\wedge(\phi_i^{-1})^*(\mathrm{d}\phi^2) \wedge\cdots\wedge(\phi_i^{-1})^*(\mathrm{d}\phi^d) \tag{14}$$

$$= \left(\sum_{j=1}^{d}(-1)^{j-1}\frac{\partial a_j}{\partial \phi^j}\right)(\phi_i^{-1}(x))\,\mathrm{d}x^1\wedge\mathrm{d}x^2\wedge\cdots\wedge\mathrm{d}x^d, \tag{15}$$

since $(\phi_i^{-1})^*(\mathrm{d}\phi_i^j) = \mathrm{d}\phi^j\circ\phi_i^{-1} = \mathrm{d}x^j$.
From the definition of form integrals (8):

$$\int_{U_i}\mathrm{d}\omega = \int_{\phi_i(U_i)}(\phi_i^{-1})^*(\mathrm{d}\omega) \tag{16}$$

$$= \int_{\phi_i(U_i)}\left(\sum_{j=1}^{d}(-1)^{j-1}\frac{\partial a_j}{\partial \phi^j}\right)(\phi_i^{-1}(x))\,\mathrm{d}x^1\wedge\mathrm{d}x^2\wedge\cdots\wedge\mathrm{d}x^d. \tag{17}$$

Using the tangent map property (13) we obtain

$$\int_{U_i}\mathrm{d}\omega = \int_{\phi_i(U_i)}\left(\sum_{j=1}^{d}(-1)^{j-1}\frac{\partial a_j\circ\phi_i^{-1}}{\partial x^j}\right)(x)\,\mathrm{d}x^1\wedge\mathrm{d}x^2\wedge\cdots\wedge\mathrm{d}x^d. \tag{18}$$

Since a_j is compactly supported within the open set U_i, a_j is zero in a neighborhood of ∂U_i and therefore on ∂U_i we have $a_j = 0$ and $\nabla a_j = 0$. Similarly on $\partial\phi_i(U_i)$, $a_j\circ\phi_i^{-1} = 0$ and $\nabla a_j\circ\phi_i^{-1} = 0$, where ∇a_j denotes the tangent map of a_j. ∇a_j goes from $T_x\Gamma$ to $\mathbb{R}$ and can therefore be identified with a vector denoted $\left(\frac{\partial a_j}{\partial\phi^1},\ldots,\frac{\partial a_j}{\partial\phi^d}\right)$.
We can therefore extend $a_j\circ\phi_i^{-1}$ to $\mathbb{R}^d$ as a Lipschitz function f_j with compact support:

$$f_j(x) = \begin{cases} a_j\circ\phi_i^{-1}(x) & \text{if } x\in U_i \\ 0 & \text{if } x\notin U_i \end{cases}. \tag{19}$$

Since a_j is Lipschitz and ϕ_i^{-1} is Lipschitz by definition of a Lipschitz chart, f_j is Lipschitz as composition of Lipschitz functions and then is differentiable by Rademacher's Theorem.

Now (18) becomes

$$\int_{U_i}\mathrm{d}\omega = \int_{\mathbb{R}^d}\left(\sum_{j=1}^{d}(-1)^{j-1}\frac{\partial f_j}{\partial x^j}\right)(x)\,\mathrm{d}x^1\wedge\mathrm{d}x^2\wedge\cdots\wedge\mathrm{d}x^d \tag{20}$$

$$= \sum_{j=1}^{d}(-1)^{j-1}\int_{\mathbb{R}^d}\frac{\partial f_j}{\partial x^j}(x)\,\mathrm{d}x^1\wedge\mathrm{d}x^2\wedge\cdots\wedge\mathrm{d}x^d \tag{21}$$

$$= \sum_{j=1}^{d}(-1)^{j-1}\int_{\mathbb{R}^{d-1}}\left(\int_{-\infty}^{+\infty}\frac{\partial f_j}{\partial x^j}(x)\,\mathrm{d}x^j\right)\mathrm{d}x^1\wedge\mathrm{d}x^2\wedge\cdots\wedge\widehat{\mathrm{d}x^j}\wedge\cdots\wedge\mathrm{d}x^d \tag{22}$$

where (22) is a consequence of Fubini's theorem.
The result follows from the fact that f_j has compact support. □

2.4 Sobolev spaces on Lipschitz manifolds

The Lebesgue spaces are defined by a pullback of the Euclidean Lebesgue space.

Definition 11 (*Lebesgue space* $L^p(\Gamma)$)
Let $p \in [1, \infty]$, Γ be a compact oriented Lipschitz manifold and $\{U_i, \phi_i\}_{i\in I}$ its Lipschitz atlas.
The space $L^p(\Gamma)$ is defined as the set of real valued functions f defined almost everywhere on Γ such that for any local chart $\{U_i, \phi_i\}$, $f \circ \phi_i^{-1} \in L^p(\phi_i(U_i))$.

The associated norm is

$$\|f\|_p = \left(\int_\Gamma |f|^p \, \mathrm{d}\lambda_\Gamma\right)^{\frac{1}{p}}. \tag{23}$$

The integral above is defined through a volume form (see formula 8).

The strong gradient $\overrightarrow{\nabla}_x \phi$ was defined in (6) for a differentiable function f at a regular point x. The usual definition of Sobolev Spaces [see 13] is based on smooth test functions. Since differentiability is limited to order one on Lipschitz manifolds we choose to replace smooth test functions by Lipschitz test functions. A function $f \in L^p(\Gamma)$ is said to be **weakly differentiable** if there exists $\vec{g} \in L^p(\Gamma)^d$ such that for any Lipschitz function $\phi : \Gamma \to \mathbb{R}$ we have

$$\int_\Gamma f \overrightarrow{\nabla}_x \phi \, \mathrm{d}\lambda_\Gamma = -\int_\Gamma \phi \vec{g} \, \mathrm{d}\lambda_\Gamma. \tag{24}$$

$\vec{g}$ is called the **weak gradient** of f and denoted $\overrightarrow{\nabla}_\Gamma f$.

Because Lipschitz manifolds admit only one order of differentiability, we can study only the space of order one weakly differentiable functions. The Sobolev space $W^{1,p}(\Gamma)$ is defined in a similar way to the Euclidean case.

Definition 12 (Sobolev space $W^{1,p}(\Gamma)$)
Let $p \in [1, \infty]$, Γ be a compact oriented Lipschitz manifold. The space $W^{1,p}(\Gamma)$ is defined as the set of weakly differentiable functions equipped with the norm

$$\|f\|^p_{W^{1,p}(\Gamma)} = \|f\|^p_p + \|\overrightarrow{\nabla}_\Gamma f\|^p_p. \tag{25}$$

We introduce the following classical notations in the Hilbertian case: $H^1(\Gamma) = W^{1,2}(\Gamma)$.

From the Definition 12 of Sobolev spaces, it is straightforward that

$$\overrightarrow{\nabla}_\Gamma : W^{1,p}(\Gamma) \to L^p(\Gamma)^d. \tag{26}$$

The definition of the weak divergence by a duality approach as well as the study of weak solution of the Poisson problem both require Sobolev spaces with negative index. Following [13] sections 3.7 to 3.13, we define $W^{-1,p}(\Gamma)$ using duality.

Definition 13 (*Dual Sobolev space* $W^{-1,p}(\Gamma)$)
Let $p \in [1, \infty[$, Γ be a compact oriented Lipschitz manifold. The space $W^{-1,p}(\Gamma)$ is defined as the dual of the space $W^{1,p}(\Gamma)$: $W^{-1,p}(\Gamma) = (W^{1,p'}(\Gamma))'$, where p' is the conjugate of p: $\frac{1}{p'} + \frac{1}{p} = 1$.

It is equipped with the norm

$$\|L\|_{W^{-1,p}(\Gamma)} = \sup_{f\in W^{1,p}(\Gamma), f\neq 0} \frac{|L(f)|}{\|f\|_{W^{1,p}(\Gamma)}}. \tag{27}$$

$W^{-1,p}(\Gamma)$ is the extension to $W^{1,p}(\Gamma)$ of distributions that act normally on infinitely smooth test functions [see Section 3.10 in 13].
The **weak divergence** is defined for $p \in]1, \infty]$ as the adjoint of the weak gradient

$$\nabla_\Gamma \cdot : L^p(\Gamma)^d \to W^{-1,p}(\Gamma). \tag{28}$$

Indeed, if $p' < \infty$ is the conjugate of p, $\overrightarrow{\nabla}_\Gamma : W^{1,p'}(\Gamma) \to L^{p'}(\Gamma)^d$ yields an adjoint operator $\nabla_\Gamma \cdot$ from $(L^{p'}(\Gamma)^d)'$ (which is isometrically equivalent to $L^p(\Gamma)^d$) into $(W^{1,p'}(\Gamma))' = W^{-1,p}(\Gamma)$.

The surface divergence of a general function $\vec{f} \in L^p(\Gamma)^d$ is therefore a linear operator acting on $W^{1,p}(\Gamma)$. However, for $\vec{f} \in W^{1,p}(\Gamma)^d$, $\nabla \cdot \vec{f}$ can be identified by duality with a function in $L^p(\Gamma)$.

The following Green's theorem connects the divergence and the gradient on Lipschitz manifolds. From a functional analytic point of view, it is an extension of the definition of weak differentiability (24) to test functions in $W^{1,p}(\Gamma)$.

Theorem 2 (Green's theorem) *Let Γ be a compact oriented Lipschitz manifold of dimension d, $u \in W^{1,p}(\Gamma)$ and $\vec{v} \in W^{1,p'}(\Gamma)^d$. Then*

$$\int_\Gamma \vec{v} \cdot \overrightarrow{\nabla}_\Gamma u \, \mathrm{d}\lambda_\Gamma = -\int_\Gamma u \nabla_\Gamma \cdot \vec{v} \, \mathrm{d}\lambda_\Gamma. \tag{29}$$

Proof Define the following Lipschitz differential forms of order $(d-1)$:

$$\omega_i = u v_i \, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \widehat{\mathrm{d}x^i} \wedge \cdots \wedge \mathrm{d}x^d, \quad i = 1, \ldots, d,$$

where $v_i, i = 1, \ldots, d$ are the components of v.
Since

$$\mathrm{d}\omega_i = \frac{\partial}{\partial x^i}(u v_i) \, \mathrm{d}x^1 \wedge \mathrm{d}x^2 \wedge \cdots \wedge \mathrm{d}x^d = \frac{\partial}{\partial x^i}(u v_i) \, \mathrm{d}\lambda_\Gamma,$$

we have

$$\nabla_\Gamma \cdot (u\vec{v}) \, \mathrm{d}\lambda_\Gamma = \sum_{i=1}^d \frac{\partial}{\partial x^i}(u v_i) \, \mathrm{d}\lambda_\Gamma = \sum_{i=1}^d \mathrm{d}\omega_i.$$

Given that in a similar way to the strong gradient, the weak gradient satisfies the product rule

$$\nabla_\Gamma \cdot (u\vec{v}) = u \nabla_\Gamma \cdot \vec{v} + \vec{v} \cdot \overrightarrow{\nabla}_\Gamma u,$$

the result follows from Stokes theorem 1:

$$\int_\Gamma \vec{v} \cdot \overrightarrow{\nabla}_\Gamma u \, \mathrm{d}\lambda_\Gamma + \int_\Gamma u \nabla_\Gamma \cdot \vec{v} \, \mathrm{d}\lambda_\Gamma = \int_\Gamma \nabla_\Gamma \cdot (u\vec{v}) \, \mathrm{d}\lambda_\Gamma = \sum_{i=1}^d \int_\Gamma \mathrm{d}\omega_i = 0.$$

□

A consequence of the Green's theorem 2 is that the operator $-\Delta_\Gamma$ is symmetric and positive. Green's theorem 2 will be used to perform the integration by part needed to obtain the weak formulation (38) of the Laplace–Beltrami operator (Theorem 3).

2.5 The Laplace–Beltrami operator on Lipschitz manifolds

The Laplace–Beltrami operator is classically defined as a second order differential operator on C^2 manifolds [see for instance 10]. The lack of smoothness on a Lipschitz manifold prevents us from making sense of the Laplace–Beltrami operator as a differential operator of order two.
One way of defining the Laplace–Beltrami operator on a Lipschitz manifold Γ is

$$\triangle_\Gamma := \nabla_\Gamma \cdot \overrightarrow{\nabla}_\Gamma : W^{1,p}(\Gamma) \to W^{-1,p}(\Gamma). \tag{30}$$

The following theorem, taken from [7] (Theorem 1.3) gives some important properties of the Laplace–Beltrami operator on Lipschitz manifolds on the Sobolev space $H^1(\Gamma) = W^{1,2}(\Gamma)$.

Theorem 3 (Laplace–Beltrami operator on Lipschitz manifolds) *Let Γ be a compact, connected, oriented Lipschitz manifold. The operator $\triangle_\Gamma := \nabla_\Gamma \cdot \overrightarrow{\nabla}_\Gamma$ is well defined, self adjoint and bounded from $W^{1,p}(\Gamma)$ to $W^{-1,p}(\Gamma)$.*
Moreover the operator $-\triangle_\Gamma$ has a purely discrete spectrum

$$0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_j \leq \ldots \tag{31}$$

with $\lambda_j \to \infty$ as $j \to \infty$.
Furthermore, there is an Hilbertian basis $(\psi_j)_{j\geq 0}$ of $L^2(\Gamma)$ composed of eigenvectors $\psi_j \in H^1(\Gamma)$ of $-\triangle_\Gamma$.

A corollary of Theorem 3 is the existence of the Poincaré constant on $H^1(\Gamma)$ which is the first non zero eigenvalue λ_1 of $-\triangle_\Gamma$.

Corollary 1 (Poincaré's inequality) *Let Γ be a compact, connected, oriented Lipschitz manifold. There exists a constant $C > 0$ such that*

$$\forall u \in H^1(\Gamma), \quad \|u - |\Gamma|^{-1} \int_\Gamma u\|_{L^2(\Gamma)} \leq C \|\nabla_\Gamma u\|_{L^2(\Gamma)}. \tag{32}$$

A classical consequence of Poincaré's inequality is that provided $\int_\Gamma u = 0$, the L^2 norm of $\nabla_\Gamma u$ is equivalent to the H^1 norm of u:

$$\int_\Gamma u = 0 \Rightarrow \|\nabla_\Gamma u\|_{L^2(\Gamma)} \leq \|u\|_{H^1(\Gamma)} \leq (1 + C)\|\nabla_\Gamma u\|_{L^2(\Gamma)}. \tag{33}$$

3 The Poisson problem on a closed Lipschitz surface

Now that we have defined the Laplace–Beltrami operator in Sect. 2.5, we can study the Poisson problem on closed Lipschitz manifolds. We start by setting the problem and the relevant functional spaces in Sect. 3.1. We then give the weak formulation of the problem in Sect. 3.2. We end up by giving an existence result in Sect. 3.3.

3.1 Definition and functional spaces

Let Γ be a closed Lipschitz manifold. Since Γ is closed, it has no boundary ($\partial\Gamma = \emptyset$), and all the constant functions u are in the kernel of Δ_Γ. Δ_Γ is therefore not invertible on the space of functions $u \in H^1(\Gamma)$.

We choose to impose the global condition $\int_\Gamma u = 0$ to guarantee the uniqueness of solution. Hence we define

$$L_0^2(\Gamma) = \left\{ f \in L^2(\Gamma), \int_\Gamma f = 0 \right\}, \qquad H_0^1(\Gamma) = \left\{ u \in H^1(\Gamma), \int_\Gamma u = 0 \right\} \tag{34}$$

Consider the following Poisson problem for a given $f \in L_0^2(\Gamma)$:

$$\text{Find } u \in H_0^1(\Gamma) \text{ such that } \quad -\Delta_\Gamma u = f. \tag{35}$$

In the next section we give the weak form of the Poisson problem (35).

3.2 Weak form of the Poisson problem

The classical Laplace–Beltrami operator acts on C^2 functions. A classical solution of (35) is therefore a function $u \in C^2(\Gamma)$. Unfortunately such a strong solution doesn't always exists even if f is assumed continuous (see Section 3.1.2 in [14] in the Euclidean case).

We now define a weak form of the Laplace Beltrami operator which acts on $H_0^1(\Gamma)$ instead of $C^2(\Gamma)$. The **weak Laplace–Beltrami operator on** Γ: $H_0^1(\Gamma) \to (H_0^1(\Gamma))'$ is the operator sending $u \in H_0^1(\Gamma)$ to the linear functional

$$H_0^1(\Gamma) \to \mathbb{R} \tag{36}$$

$$v \to \int_\Gamma \overrightarrow{\nabla}_\Gamma u \cdot \overrightarrow{\nabla}_\Gamma v \, \mathrm{d}\lambda_\Gamma, \tag{37}$$

where $\overrightarrow{\nabla}_\Gamma u$ is the weak gradient of u on Γ given by (24).

The variational formulation for (35) is the following.

$$\text{Find } u \in H_0^1(\Gamma) \text{ such that } \quad \forall v \in H_0^1(\Gamma), \int_\Gamma \overrightarrow{\nabla}_\Gamma u \cdot \overrightarrow{\nabla}_\Gamma v \, \mathrm{d}\lambda_\Gamma = \int_\Gamma f v \, \mathrm{d}\lambda_\Gamma. \tag{38}$$

We obtain (38) from (35) using the Green's theorem 2.

A solution $u \in H_0^1(\Gamma)$ of (38) is called a **weak solution** of the Poisson problem (35). Indeed, it is only one time weakly differentiable whereas a strong (classical) solution is twice strongly differentiable.

In the next section we are going to prove following [2], that the Poisson problem with a zero mean right hand side admits, a unique solution with zero mean on a closed Lipschitz manifold.

3.3 Existence result

The existence and uniqueness of weak solutions for the variational formulation (38) of problem (35) on non smooth manifolds Γ is to our knowledge open. Classical existence results require the smoothness of Γ [see for instance Chapter 4, Section 1.2 in 10]. The following statement is taken from [1] Theorem 1 b).

Theorem 4 (Existence and uniqueness of weak solutions on a C^3 hypersurface [1])
Let Γ be a closed embedded C^3 hypersurface in $\mathbb{R}^3$. For every $f \in L_0^2(\Gamma)$, there exists a unique weak solution $u \in H_0^1(\Gamma)$ of $-\Delta u = f$ on Γ.

Proof : see Theorem 1 b) in [1]. □

Unfortunately on a Lipschitz manifold, similar existence results can not be found in the literature. Besides, as mentioned in [1] page 3, it is important to check that the space $H^1(\Gamma_h)$ is well defined, where Γ_h results from a triangulation of Γ.

The following statement is taken from [1] Theorem 2 b).

Theorem 5 (Existence and uniqueness of weak solutions on a triangulated hypersurface [1]) *Let Γ_h be a closed embedded $C^{0,1}$ hypersurface in $\mathbb{R}^3$. For every $f_h \in L^2(\Gamma_h)$ with $\int_{\Gamma_h} f_h = 0$, there exists a weak solution $u_h \in H^1(\Gamma_h)$ of $-\triangle u_h = f_h$ on Γ_h. Furthermore u_h is unique up to a constant.*

Dziuk [1] claims that the proof is a "simple application of usual Hilbert space methods". However these Hilbert space methods require a geometrical and functional setting for calculus on Lipschitz manifolds that is not easily found in the literature. Moreover Stokes' theorem 1 and Poincaré's inequality (Corollary 1) on Lipschitz manifolds are never stated and do not seem trivial to us.

Our existence theorem on Lipschitz manifolds relies on the Stokes' theorem 1 and on the Lax-Milgram's theorem.

Theorem 6 (Existence theorem on Lipschitz manifolds)
Let Γ be a closed Lipschitz manifold. Let $f \in L^2_0(\Gamma)$. There exists a unique weak solution $u \in H^1_0(\Gamma)$ of $-\Delta_\Gamma u = f$ on Γ. Furthermore, u depends continuously on the right hand side:

$$\exists C' > 0,\ \|u\|_{H^1_0(\Gamma)} \leq C' \|f\|_{L^2_0(\Gamma)}. \tag{39}$$

Proof In order to use the Lax–Milgram's theorem, using the bilinear form

$$\begin{array}{ccc} a(\cdot,\cdot) : H^1_0(\Gamma) \times H^1_0(\Gamma) & \to & \mathbb{R} \\ (u, v) & \to & \int_\Gamma \overrightarrow{\nabla}_\Gamma u \cdot \overrightarrow{\nabla}_\Gamma v \, \mathrm{d}\lambda_\Gamma \end{array}$$

and the linear form

$$\begin{array}{ccc} b(\cdot) : H^1_0(\Gamma) & \to & \mathbb{R} \\ v & \to & \int_\Gamma f v \, \mathrm{d}\lambda_\Gamma \end{array}$$

we rewrite the weak formulation (38) of (35) as

$$a(u, v) = b(v),\ \forall v \in H^1_0(\Gamma)$$

- The continuity of the bilinear form $a(\cdot,\cdot)$ is a consequence of the Cauchy-Schwarz's inequality:

$$\begin{aligned} |a(u, v)| &= \left| \int_\Gamma \overrightarrow{\nabla}_\Gamma u \cdot \overrightarrow{\nabla}_\Gamma v \, \mathrm{d}\lambda_\Gamma \right| \\ &\leq \|\overrightarrow{\nabla}_\Gamma u\|_{L^2_0(\Gamma)} . \|\overrightarrow{\nabla}_\Gamma v\|_{L^2_0(\Gamma)} \\ &\leq \|u\|_{H^1_0(\Gamma)} \|v\|_{H^1_0(\Gamma)} \end{aligned}$$

- We obtain the coercivity of $a(\cdot,\cdot)$ thanks to the Poincaré's inequality (Corollary 1) or more precisely inequality (33):

$$\begin{aligned} a(u, u) &= \left(\overrightarrow{\nabla}_\Gamma u, \overrightarrow{\nabla}_\Gamma u \right)_{L^2_0(\Gamma)} \\ &= \|\overrightarrow{\nabla}_\Gamma u\|^2_{L^2_0(\Gamma)} \\ &\geq \frac{1}{(1 + C)^2} \|u\|^2_{H^1_0(\Gamma)} \end{aligned}$$

- The continuity of $b(\cdot)$ is the consequence of the Cauchy-Schwarz's inequality and the Poincaré's inequality (Corollary 1)

$$\begin{aligned} |b(v)| = |(f, v)_{L_0^2(\Gamma)}| &\leq \|f\|_{L_0^2(\Gamma)} \|v\|_{L_0^2(\Gamma)} \\ &\leq \|f\|_{L_0^2(\Gamma)} \|v\|_{H_0^1(\Gamma)} \end{aligned}$$

Since the requirement of the Lax-Milgram's theorem are satisfied, the existence of a unique solution to the weak formulation (38) satisfying the stability estimate (39) holds with $C' = \frac{1}{(1+C)^2}$. □

In the next section we introduce the linear finite element method.

4 The finite element method

The finite element method (FEM) is a numerical method for solving problems of engineering and mathematical physics that are formulated by PDEs [see, e.g., 3, 14]).

When the domain is a hypersurface Γ, we assume as done in [1], the existence of a polyhedric surface Γ_h that approximates Γ and is made of triangles with nodes on Γ. We can approximate functions $u \in H_0^1(\Gamma)$ by functions $u_h \in H_0^1(\Gamma_h)$. We can then approximate functions $u_h \in H_0^1(\Gamma_h)$ by their projection on $PL_0(\Gamma_h)$ the space of continuous piecewise affine functions with zero mean.

Considering the weak Laplace–Beltrami operator on the piecewise linear manifold Γ_h, we seek for $\tilde{u}_h \in PL_0(\Gamma_h)$ the projection on $PL_0(\Gamma_h)$ of the solution $u_h \in H_0^1(\Gamma_h)$ of the Poisson problem $-\triangle_{\Gamma_h} u_h = f_h$. The finite element approximation therefore projects the original problem set on $H_0^1(\Gamma)$ onto the finite dimensional space $PL_0(\Gamma_h)$ using a discrete variational formulation.

The **discrete variational formulation** of the Poisson equation (35) is the following.

$$\text{Find } \tilde{u}_h \in PL_0(\Gamma_h) \text{ such that } \forall \tilde{v}_h \in PL_0(\Gamma_h), \quad \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h} \tilde{u}_h \cdot \overrightarrow{\nabla}_{\Gamma_h} \tilde{v}_h = \int_{\Gamma_h} f_h \tilde{v}_h. \tag{40}$$

In a similar fashion as in Eqs. (36, 37), the **discrete Laplace–Beltrami operator** $-\tilde{\triangle}_h : PL_0(\Gamma_h) \to PL_0(\Gamma_h)'$ is the operator sending $\tilde{u}_h \in PL_0(\Gamma_h)$ to the linear functional

$$PL_0(\Gamma_h) \to \mathbb{R} \tag{41}$$

$$\tilde{v}_h \to \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h} \tilde{u}_h \cdot \overrightarrow{\nabla}_{\Gamma_h} \tilde{v}_h, \tag{42}$$

where $\overrightarrow{\nabla}_{\Gamma_h} \tilde{u}_h$ is the weak gradient of $\tilde{u}_h$ on Γ_h given by (24).

$-\tilde{\triangle}_h$ acts between finite dimensional spaces and approximates the continuous weak Laplace–Beltrami operator (36, 37). It can be represented by a **stiffness matrix** $A_{\triangle_{\Gamma_h}}$ with size n the number of nodes of Γ_h.

The unknown function $\tilde{u}_h$ belongs to the finite dimensional space $PL_0(\Gamma_h)$. Define the **unknown vector** $(u_1, ..., u_n)$ as the vector of components of $\tilde{u}_h$ on the nodal basis:

$$\tilde{u}_h = \sum_{i=1}^{n} u_i \phi_i. \tag{43}$$

Since $PL_0(\Gamma_h)$ is generated by the nodal functions $\phi_i : \Gamma_h \to \mathbb{R}, \quad i = 1, ..., n$ such that $\phi_i(x_j) = \delta_{ij}$, the solution $\tilde{u}_h$ of the discrete Poisson problem (40) must therefore satisfy the following system of equations

$$\forall i \in \{1, ..., n\}, \qquad \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h} \tilde{u}_h \cdot \overrightarrow{\nabla}_{\Gamma_h} \phi_i = \int_{\Gamma_h} f_h \phi_i, \tag{44}$$

which takes the algebraic form

$$A_{\triangle_{\Gamma_h}} X = b_h, \quad \text{where } X = {}^t(u_1, ..., u_n). \tag{45}$$

From (44) the coefficients of the stiffness matrix $A_{\triangle_{\Gamma_h}} = (a_{ij})_{i,j=1,...,n}$, and of the right hand side vector $b_h = {}^t(b_1, ..., b_n)$ are therefore given by

$$a_{ij} = \int_{\Gamma_h} \overrightarrow{\nabla}_{\Gamma_h} \phi_i \cdot \overrightarrow{\nabla}_{\Gamma_h} \phi_j, \tag{46}$$

$$b_j = \int_{\Gamma_h} f \phi_j. \tag{47}$$

The stiffness matrix $A_{\triangle_{\Gamma_h}}$ is symmetric positive but not invertible [see Theorem 4.1 in 4]). However the finite element linear system admits a unique solution provided the right hand side has zero mean [see Theorem 4.2 in 4], hence the existence of the discrete solution $\tilde{u}_h$ as stated in the following theorem.

Theorem 7 (Existence of the discrete solution) *Let $f_h \in PL_0(\Gamma_h)$, there exists a unique discrete solution $\tilde{u}_h \in PL_0(\Gamma_h)$ to the discrete Poisson problem (40).*

The computation of the coefficients a_{ij} is usually performed by expressing the integral in (46) as a sum over the triangles T_k composing Γ_h:

$$a_{ij} = \sum_{T_k} \int_{T_k} \overrightarrow{\nabla}_{\Gamma_h} \phi_i \cdot \overrightarrow{\nabla}_{\Gamma_h} \phi_j. \tag{48}$$

The gradients $\overrightarrow{\nabla}_{\Gamma_h} \phi_i$ in (48) are thus computed in each separate triangle T_k where their value is well defined and constant. There is no need to compute the gradient at the edge between two triangles T_k, where it is not properly defined. Therefore the lack of smoothness of Γ_h does not yield any singularity in the computation of the stiffness matrix coefficients.

The coefficients a_{ij} of $A_{\triangle_{\Gamma_h}}$ can be computed explicitly as a function of the mesh nodes coordinates and are given in [4].

5 Numerical experiments on the unit cube boundary

In order to illustrate the applicability of our theoretical study we present in this section the numerical simulation of a Poisson problem on the boundary of the unit cube.

5.1 Setting of the problem

We consider the unit cube $\Omega_{cube} = [0, 1] \times [0, 1] \times [0, 1] \subset \mathbb{R}^3$ and its boundary

$$\Gamma = \partial \Omega_{cube}.$$

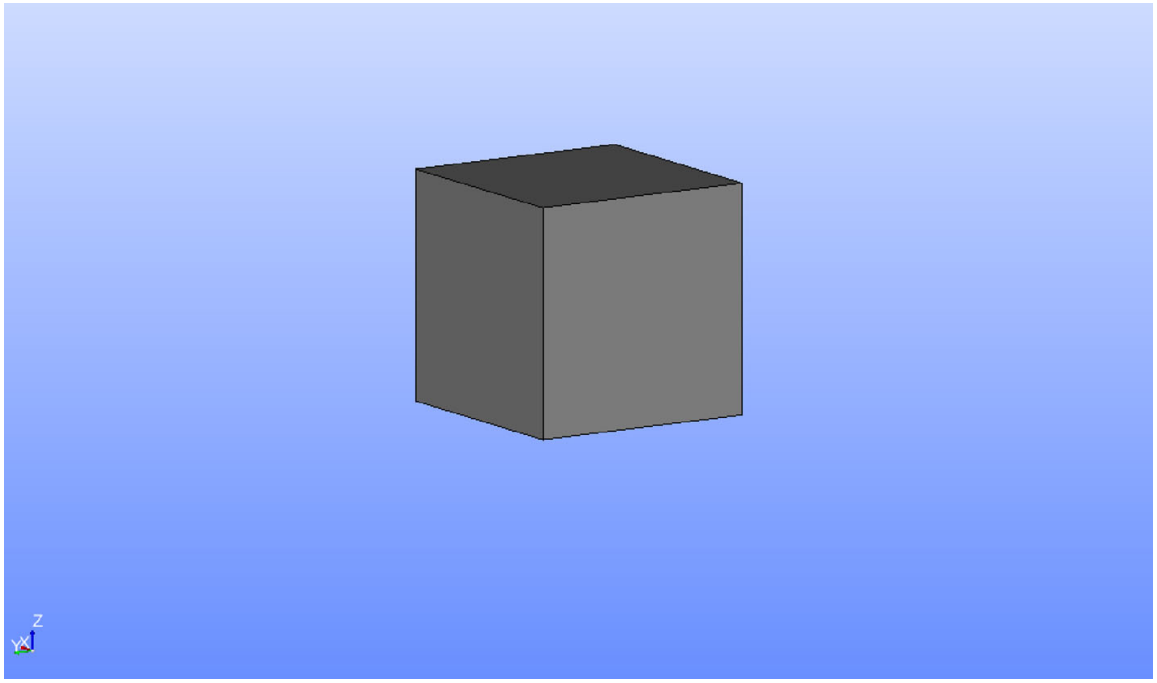


Fig. 1 The unit cube in SALOME CAO module

Γ is a topological manifold but not a differential manifold because of the presence of sharp edges where Γ admits no tangent space. Γ is however a Lipschitz manifold,which admits tangent spaces except at the edges. As a Lipschitz manifold, Γ can be endowed with a Laplace–Beltrami operator $\triangle_\Gamma = \nabla \cdot \overrightarrow{\nabla}$, defined as the combination of a surface divergence $\nabla_\Gamma \cdot$, and of a surface gradient $\overrightarrow{\nabla}_\Gamma$ [see 7] (Fig. 1).

We consider f the restriction to Γ of the smooth function $\cos(2\pi x)\cos(2\pi y)\cos(2\pi z)$. The explicit expressions of f on each face of Γ are

$$f(x, y, z) = \begin{cases} \cos(2\pi x)\cos(2\pi y) & \text{if } z = 0 \text{ or } z = 1 \\ \cos(2\pi x)\cos(2\pi z) & \text{if } y = 0 \text{ or } y = 1 \\ \cos(2\pi y)\cos(2\pi z) & \text{if } x = 0 \text{ or } x = 1 \end{cases}. \tag{49}$$

f is an eigenfunction of the Laplace–beltrami operator on Γ since

$$\begin{aligned} \triangle_\Gamma f(x, y, z) &= \begin{cases} \partial_{xx}\cos(2\pi x)\cos(2\pi y) + \partial_{yy}\cos(2\pi x)\cos(2\pi y) & \text{if } z = 0 \text{ or } z = 1 \\ \partial_{xx}\cos(2\pi x)\cos(2\pi z) + \partial_{zz}\cos(2\pi x)\cos(2\pi z) & \text{if } y = 0 \text{ or } y = 1 \\ \partial_{yy}\cos(2\pi y)\cos(2\pi z) + \partial_{zz}\cos(2\pi y)\cos(2\pi z) & \text{if } x = 0 \text{ or } x = 1 \end{cases} \\ &= \begin{cases} -2(2\pi)^2\cos(2\pi x)\cos(2\pi y) & \text{if } z = 0 \text{ or } z = 1 \\ -2(2\pi)^2\cos(2\pi x)\cos(2\pi z) & \text{if } y = 0 \text{ or } y = 1 \\ -2(2\pi)^2\cos(2\pi y)\cos(2\pi z) & \text{if } x = 0 \text{ or } x = 1 \end{cases} \\ &= -8\pi^2 f. \end{aligned}$$

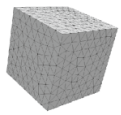
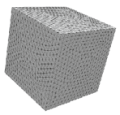
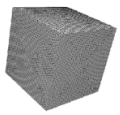
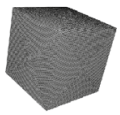
meshCubeSkin 1	meshCubeSkin 2	meshCubeSkin 3	meshCubeSkin 4
			
412 cells	1923 cells	7042 cells	13225 cells

Fig. 2 Meshes of the unit cube boundary

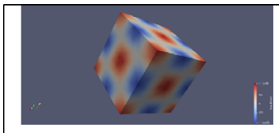
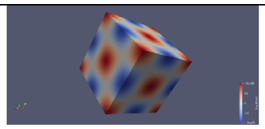
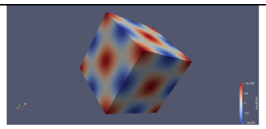
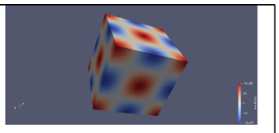
			
412 cells	1923 cells	7042 cells	13225 cells

Fig. 3 Numerical results of the finite elements on the unit cube boundary

We are going to solve the following Poisson problem on Γ:

$$
\begin{aligned}
-\triangle_{\Gamma} u &= f, \\
\int_{\Gamma} u &= 0,
\end{aligned}
\tag{50}
$$

where the right hand side f defined in (49) and the unknown $u \in H^1(\Gamma)$ are **zero mean functions**.

Our objective is to solve numerically the Poissson problem (35) using the finite element method described in [1].

- For the numerical resolution of our discrete problem, we use an iterative solver (Conjugate Gradient) because the stiffness matrix $A_{\triangle_{\Gamma_h}}$ is large and sparse [see 15].
- For the design and meshing of the domain we use the GEOMETRY and MESH modules of the software SALOME 9.5 [see 16, 17].
- For the visualisation of the result, we use the PARAVIS module included in SALOME [see 17].
- For the coding of the script, we use Python language and the open-source Linux based library [18] which is very practical for the manipulation of large matrices, vectors, meshes and fields. It (SOLVERLAB) can handle finite element and finite volume discretizations, read general $1D$, $2D$ and $3D$ geometries and meshes generated by SALOME.

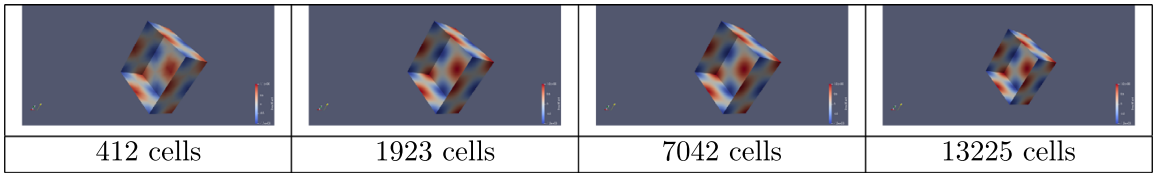
5.2 Meshing of the domain

Below are the meshes used in our convergence analysis (Fig. 2).

5.3 Visualisation of the results

Below are visualisations of the numerical results obtained on the different meshes

Below are clipings of the previous numerical results (Figs. 3, 4).



412 cells	1923 cells	7042 cells	13225 cells

Fig. 4 Clipings of the numerical results on the unit cube boundary

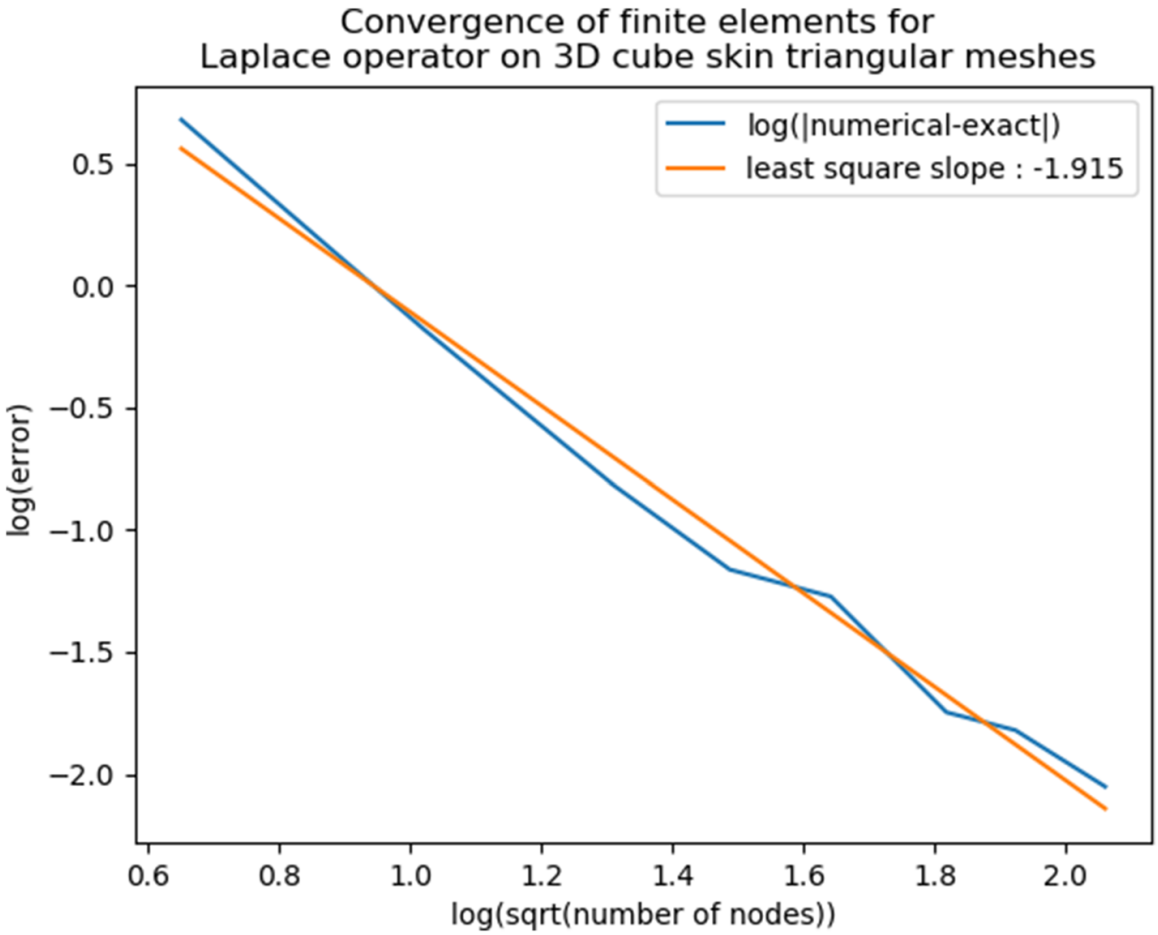


Fig. 5 Convergence of the finite element method on the cube boundary

5.4 Numerical convergence

When refining the mesh, we observe the convergence of the method with a numerical order of 1.91 (Fig. 5).

6 Conclusion and perspectives

The notion of Lipschitz manifold has been studied by some authors [6, 7] and is usefull in laying the foundation of the finite element method on general compact manifolds. We showed how it allows the proper definition of tangent spaces, gradient and divergence operators almost everywhere. Differentiability almost everywhere thanks to Rademacher's theorem is the key ingredient of the calculus on Lipschtz manifolds.

We have proposed proofs of the Stokes' (1) and Green's (2) theorems as well as a Poincaré's inequality. This enables the weak formulation of the Poisson problem. The well-posedness of

the weak formulation of the Poisson problem is then obtained thanks to the Lax-Milgram's theorem.

We gave an example of numerical simulation on a Lipschitz (non smooth) manifold to illustrate the fact that the lack of smoothness does not yield any numerical singularity. The numerical results of the Poisson problem on the unit cube boundary showed that the finite element approximation converges towards the exact solution.

This study has set the theoretical ground for further numerical analysis of more complex PDEs on Lipschitz manifolds. Note however that on Lipschitz manifolds the maximum order of differentiability is one which is enough for the classical weak formulation of second order elliptic PDEs. This is however a technical difficulty for general high order PDEs on Lipschitz manifolds. One workaround for PDEs of order three and more could however be to reset the PDE into a system of low order PDEs.

Yet, with this new approach we were not able to extend the regularity and convergence theorems of [1, 2] because the lack of smoothness of the Lipschitz manifold Γ prohibits the use of the lift operator. Further research should therefore be devoted to the theoretical analysis of the convergence of both the approximate manifold Γ_h and solution u_h towards the exact manifold Γ and solution u.

Author Contributions Not applicable.

Funding Not applicable.

Availability of data and materials The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no conflict of interest regarding the publication of this paper.

References

1. Dziuk, G.: Finite elements for the Beltrami operator on arbitrary surfaces in partial differential equations and calculus of variations. In: Keyes, D.E., Xu, J. (eds.) Lecture Notes Mathematics, vol. 1357, pp. 142–155. Springer, Berlin (1988)
2. Dziuk, G., Elliott, C.M.: Finite element methods for surface PDEs. Acta Numer. **22**, 289–396 (2013)
3. Elliott, C.M., Stinner, B.: Computation of two-phase biomembranes with phase dependent material parameters using surface finite elements. Commun. Comput. Phys. **13**, 325–360 (2010)
4. Nguemfouo, M., Ndjinga, M.: On the condition number of the finite element method for Laplace–Beltrami operator. (Revised in J. Elliptic Parabol. Equ) (2022)
5. Wloka, J.: Partial Differential Equations. Cambridge University Press, Cambridge (1987)
6. Goldshtein, V., Mitrea, I., Mitrea, M.: Hodge decompositions with mixed boundary conditions and application to partial differential equations on Lipschitz manifolds. J. Math. Sci. **172**, 347–400 (2011)
7. Gesztesy, F., Mitrea, I., Mitrea, D., Mitrea, M.: On the nature of the Laplace–Beltrami operator on Lipschitz manifolds. J. Math. Sci. **172**, 279–346 (2011)
8. Luukkainen, J., Väisälä, J.: Elements of Lipschitz topology. Ann. Acad. Sci. Fennicre Ser. A. I. Math. **3**, 85–122 (1977)
9. Hebey, E.: Nonlinear Analysis on Manifolds: Sobolev Spaces and Inequalities. Courant Institute of Mathematical Scienes, New York University (2000)
10. Aubin, T.: Some Nonlinear Problems in Riemannian Geometry. Springer, Berlin (1998)
11. Taylor, M.: Measure Theory and Integration. American Mathematical Society, Providence (2006)
12. Lafontaine, J.: An Introduction to Differentiable Manifolds. Springer, Berlin (2015)
13. Adams, R.A., Fournier, J.J.: Sobolev Spaces. In: Pure Applied Mathematics, 2nd edn. Elsevier, Academic Press, New York (2003)

14. Allaire, G.: Numerical Analysis and Optimization: an Introduction to Mathematical Modeling and Numerical Simulation. Oxford University Press, Oxford (2007)
15. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing Company, Boston (1996)
16. Ribes A., Caremoli C.: Salome platform component model for numerical simulation. COMPSAC. 31st Annual international computer software and applications conference, vol. 2. IEEE (2007)
17. SALOME: http://www.salome-platform.org (2001)
18. SOLVERLAB: https://github.com/ndjinga/SOLVERLAB (2021)