MIMESIS

UNIVERSITY OF STRASBOURG

MASTER CSMI

Internship Report : Innovative non-conformal finite element methods for augmented surgery

Authors:
Frédérique Lecourtier

Supervisors:
Michel Duprez
Emmanuel Franck
Vanessa Lleras

Inria

Date: July 28, 2023

Contents

1 Introduction

This end of study internship is a 2nd year internship in the CSMI Master ("Calcul Scientifique et Mathématique de l'Information") of the University of Strasbourg. It is the continuation of a project done during the first semester of M2, the main objective of this project was the discovery of an innovative non-conformal finite element method for augmented surgery, the ϕ-FEM method. The purpose of this project was to have Cemosis and Mimesis collaborate through the use of Feel++ software (developed by Cemosis) in the framework of the ϕ-FEM method (one of the research topics of the Mimesis team).

1.1 Scientific Context

Finite element methods (FEM) are used to solve partial differential equations numerically. These can, for example, represent analytically the dynamic behavior of certain physical systems (mechanical, thermodynamic, acoustic, etc.). Among other things, it is a discrete algorithm for determining the approximate solution of a partial differential equation (PDE) on a compact domain with boundary conditions.
The standard FEM method, which requires precise meshing of the domain under consideration and, in particular, fitting with its boundary, has its limitations. In particular, in the medical field, meshing complex and evolving geometries such as organs (e.g. the liver) can be very costly. More specifically, in the application context of creating real-time digital twins of an organ, the standard FEM method would require complete remeshing of the organ each time it is deformed, which in practice is not workable.
This is why other methods, known as non-conformal finite element methods, have emerged in the last few years. These include CutFEM [2] or XFEM [8], based on the idea of introducing a fictitious domain larger than the domain under consideration. We're interested here in another non-conformal method, which we'll present in more detail later, called ϕ-FEM. We'll only use it in the context of Poisson problem solving, for Dirichlet boundary conditions [6]. But the method has been extended to Neumann conditions [5] and then to solve various mechanical problems, including linear elasticity and heat transfer problems, and even to solve the Stokes problem [4]

1.2 Presentation of Mimesis

Mimesis is a joint Inria ("Institut national de recherche en sciences et technologies du numérique") and CNRS ("Centre national de la recherche scientifique") Research Team. The Mimesis team was created in May 2021 with the Research Director, Stéphane Cotin, as leader.
The Mimesis research team is working on a set of scientific challenges in scientific computing, data assimilation, machine learning and control, with the goal of creating real-time digital twins of an organ. Their main application areas are surgical training and surgical guidance in complex procedures. Their main clinical objectives are liver surgery, lung surgery and neurostimulation.
A COMPLETER !

1.3 Objectives

Première étape : comprendre les FNO (phifem ok car projet)
L'objectif principal du stage a été de combiner méthodes d'éléments finis et Machine Learning afin de résoudre le problème de Poisson avec condition de Dirichlet. Plus précisément, on souhaite entraîner, avec des solutions ϕ-FEM, un Fourier Neural Operator (FNO) à nous prédire les solutions pour une famille de problèmes donnée (c'est-à-dire un "type" de terme source). Les prédictions de ce réseau de neurones seront ensuite réinjecter dans un solveur élément finis afin d'y appliquer une correction.
Pour être plus précis, on testera différentes méthodes de correction qui permettront de réutiliser la prédiction du réseau pour aider le solveur à s'approcher le plus précisément possible de la solution.
On a testé sur une sol analytique, parler de sol plus perturbation, résultat analytique (preuve) et numérique...
On espère alors améliorer considérablement la précision du modèle et possiblement les temps de calcul.
The ϕ-FEM method has an advantage that is very interesting in the context of organ geometries. Indeed, this type of geometry can deform in time and meshing a fictitious domain around this geometry avoids having to remesh the geometry in time. Thus only the levelset function will be modified and the mesh can be fixed. Moreover, a Cartesian mesh of the fictitious domain allows us to use the same type of neural network as those applied to images: this is the subject that will be approached during the internship.

1.4 Deliverables

AJOUTER : rapport antora (avec ci github) / github repo / doc sphinx / suivi hebdomadaire du travail ... ?

2 Finite Element Methods (FEMs)

In the following, we will consider the Poisson problem with Dirichlet condition (homogeneous or inhomogeneous):
Problem : Find $u : \Omega \to \mathbb{R}^d$ such that

$$\begin{cases} -\Delta u = f, & \text{in } \Omega, \\ u = g, & \text{on } \partial\Omega, \end{cases}$$

with Δ the Laplace operator and $\Omega \subset \mathbb{R}^d$ a lipschitzian bounded open set (and $\partial\Omega$ its boundary).
Associated physical model : Newtonian gravity, Electrostatics, Fluid dynamics...
AJOUTER REF !

2.1 Standard FEM

In this section, we will present the standard finite element method. We'll start by presenting some general notions of functional analysis, then explain the general principle of FEM. Then we'll give a few more details on the method and finish by describing the application to the Poisson problem (with Dirichlet condition).

2.1.1 Some notions of functional analysis.

AJOUTER : Déf Espace de Hilbert (+ ce qu'il faut pour def ça : Espace de Soboloev ? ...) + Déf L^2,

2.1.2 General principle of the method

Let's consider a domain Ω whose boundary is denoted $\partial\Omega$. We seek to determine a function u defined on Ω, solution of a partial differential equation (PDE) for given boundary conditions. The general approach of the finite element method is to write down the variational formulation of this PDE, thus giving us a problem of the following type:
Variational Problem :

$$\text{Find } u \in V \text{ such that } a(u,v) = l(v),\ \forall v \in V$$

where V is a Hilbert space, a is a bilinear form and l is a linear form.
To do this, we multiply the PDE by a test function $v \in V$, then integrate over $L^2(\Omega)$.
The idea of FEM is to use Galerkin's method. We then look for an approximate solution u_h in V_h, a finite-dimensional space dependent on a positive parameter h such that

$$V_h \subset V, \quad \dim V_h = N_h < \infty, \quad \forall h > 0$$

The variational problem can then be approached by :

Approach Problem :

$$\text{Find } u_h \in V_h \text{ such that } a(u_h, v_h) = l(v_h),\ \forall v_h \in V$$

As V_h is of finite dimension, we can consider a basis $(\varphi_1, \dots, \varphi_{N_h})$ of V_h and thus decompose u_h on this basis as :

$$u_h = \sum_{i=1}^{N_h} u_i \varphi_i \tag{1}$$

The approached problem is then rewritten as

$$\text{Find } u_1, \dots, u_{N_h} \text{ such that } \sum_{i=1}^{N_h} u_i a(\varphi_i, v_h) = l(v_h),\ \forall v_h \in V$$

and

$$\text{Find } u_1, \dots, u_{N_h} \text{ such that } \sum_{i=1}^{N_h} u_i a(\varphi_i, \varphi_j) = l(\varphi_j),\ \forall j \in \{1, \dots, N_h\}$$

Solving the PDE involves solving the following linear system:

$$AU = b$$

with

$$A = (a(\varphi_i, \varphi_j))_{1 \le i,j \le N_h}, \quad U = (u_i)_{1 \le i \le N_h} \quad \text{and} \quad b = (l(\varphi_j))_{1 \le j \le N_h}$$

2.1.3 Some details on FEM

After having seen the general principle of FEM, it remains to define the V_h spaces and the $\{\varphi_i\}$ basis functions.

Remark. *The choice of V_h space is fundamental to have an efficient method that gives a good approximation u_h of u. In particular, the choice of the $\{\varphi_i\}$ basis of V_h influences the structure of the A matrix in terms of its sparsity and its condition number.*

To do this, we'll need several notions, which will be detailed in the following sections. First, we'll need to generate a **mesh** of our Ω domain. This will enable us to solve the PDE discretely at selected points. This is where the notion of **finite Lagrange elements** comes in. The properties of these elements, particularly in terms of their **affine family of finite elements**, is a key point of the method, which will enable us to bring each element of the mesh back to a **reference element** by using a **geometric transformation**. To describe these steps, we'll need to know 2 basic concepts: the **unisolvance** principle and the definitions of the **polynomial spaces** used ($\mathbb{P}_k$ and $\mathbb{Q}_k$).

2.1.3.1 Unisolvance

Definition 2.1. *Let $\Sigma = \{a_1, \ldots, a_N\}$ be a set of N distinct points of $\mathbb{R}^n$. Let P be a finite-dimensional vector space of $\mathbb{R}^n$ functions taking values in $\mathbb{R}$. We say that Σ is P-unisolvent if and only if for all real $\alpha_1, \ldots, \alpha_N$, there exists a unique element p of P such that $p(a_i) = \alpha_i, i = 1, \ldots, N$. This means that the function*

$$\begin{aligned} L : P &\to \mathbb{R}^N \\ p &\mapsto (p(a_1), \ldots, p(a_N)) \end{aligned}$$

is bijective.

Remark. *In practice, to show that Σ is P-unisolvent, we simply check that $\dim P = card\,(\Sigma)$ and then prove the injectivity or surjectivity of L. The injectivity of L is demonstrated by showing that the only function of P that annuls on all points of Σ is the null function. The surjectivity of L is shown by identifying a family $p_1, \ldots, p_N$ of elements of P such that $p_i(a_j) = \delta_{ij}$. Given real $\alpha_1, \ldots, \alpha_N$, the function $p = \sum_{i=1}^N \alpha_i p_i$ then verifies $p(a_j) = \alpha_j, j = 1 \ldots, N$.*

Remark. *We call local basis functions of element K the N functions $p_1, \ldots, p_N$ of P such that*

$$p_i(a_j) = \delta_{ij}, \quad 1 \le i, j \le N$$

2.1.3.2 Polynomial space

Let $\mathbb{P}_k$ be the vector space of polynomials of total degree less than or equal to k.

- In $\mathbb{R}$: $\mathbb{P}_k = \text{Vect}\{1, X, \ldots, X^k\}$ and $\dim \mathbb{P}_k = k+1$
- In $\mathbb{R}^2$: $\mathbb{P}_k = \text{Vect}\{X^i Y^j, 0 \le i+j \le k\}$ and $\dim \mathbb{P}_k = \frac{(k+1)(k+2)}{2}$
- In $\mathbb{R}^3$: $\mathbb{P}_k = \text{Vect}\{1, X^i Y^j Z^l, 0 \le i+j+l \le k\}$ and $\dim \mathbb{P}_k = \frac{(k+1)(k+2)(k+3)}{6}$

Let $\mathbb{Q}_k$ be the vector space of polynomials of degree less than or equal to k with respect to each variable.

- In $\mathbb{R}$: $\mathbb{Q}_k = \mathbb{P}_k$.
- In $\mathbb{R}^2$: $\mathbb{Q}_k = \text{Vect}\{X^i Y^j, 0 \le i, j \le k\}$ and $\dim \mathbb{Q}_k = (k+1)^2$
- In $\mathbb{R}^3$: $\mathbb{Q}_k = \text{Vect}\{1, X^i Y^j Z^l, 0 \le i, j, l \le k\}$ and $\dim \mathbb{Q}_k = (k+1)^3$

2.1.3.3 Finite Lagrange Element

The most classic and simplest type of finite element is the Lagrange finite element.

Definition 2.2 (Lagrange Finite Element)**.** *A finite Lagrange element is a triplet (K, Σ, P) such that*

- *K is a geometric element of $\mathbb{R}^n$ ($n = 1, 2$ or 3), compact, connected and of non-empty interior.*
- *$\Sigma = \{a_1, \ldots, a_N\}$ is a finite set of N distinct points of K.*
- *P is a finite-dimensional vector space of real functions defined on K and such that Σ is P-unisolvent (so $\dim P = N$).*

Example. *Let K be the segment $[a_1, a_2]$. Let's show that $\Sigma = \{a_1, a_2\}$ is P-unisolvent for $P = \mathbb{P}^1$.*
Since $\{1, x\}$ is a base of $\mathbb{P}^1$, we have $\dim P = card\, \Sigma = 2$.
Moreover, we can write $p_i = \alpha_i x + \beta_i, i = 1, 2$. Thus

$$\begin{cases} p_1(a_1) = 1 \\ p_1(a_2) = 0 \end{cases} \iff \begin{cases} \alpha_1 a_1 + \beta_1 = 1 \\ \alpha_1 a_2 + \beta_1 = 0 \end{cases} \iff \begin{cases} \alpha_1 = \dfrac{1}{a_1 - a_2} \\ \beta_1 = -\dfrac{a_2}{a_1 - a_2} \end{cases}$$

and

$$\begin{cases} p_2(a_1) = 0 \\ p_2(a_2) = 1 \end{cases} \iff \begin{cases} \alpha_2 a_1 + \beta_2 = 0 \\ \alpha_2 a_2 + \beta_2 = 1 \end{cases} \iff \begin{cases} \alpha_1 = \dfrac{1}{a_2 - a_1} \\ \beta_1 = -\dfrac{a_1}{a_2 - a_1} \end{cases}$$

Thus

$$p_1(x) = \frac{x - a_2}{a_1 - a_2} \quad \text{and} \quad p_2(x) = \frac{x - a_1}{a_2 - a_1}$$

We deduce the surjectivity of L and Σ is $\mathbb{P}^1$-unisolvent.
Thus (K, Σ, P) is a Lagrange Finite Element.

Definition 2.3. *Two finite elements $(\hat{K}, \hat{\Sigma}, \hat{P})$ and (K, Σ, P) are affine-equivalent if and only if there exists an inversible affine function F such that*

- *$K = F(\hat{K})$*
- *$a_i = F(\hat{a}_i), i = 1, \ldots, N$*
- *$P = \{\hat{p} \circ F^{-1}, \hat{p} \in \hat{P}\}$.*

*We then call an **affine family of finite elements** a family of finite elements, all affine-equivalent to the same element $(\hat{K}, \hat{\Sigma}, \hat{P})$, called the **reference element**.*

Remark. *Let $(\hat{K},\hat{\Sigma},\hat{P})$ and (K,Σ,P) be two affine-equivalent finite elements, via an F transformation. Let $\hat{p}_i$ be the local basis functions on $\hat{K}$. Then the local basis functions on K are $p_i = \hat{p}_i \circ F^{-1}$.*

Remark. *In practice, working with an affine family of finite elements means that all integral calculations can be reduced to calculations on the reference element.*
The reference elements in 1D, 2D triangular and 3D tetrahedral are :

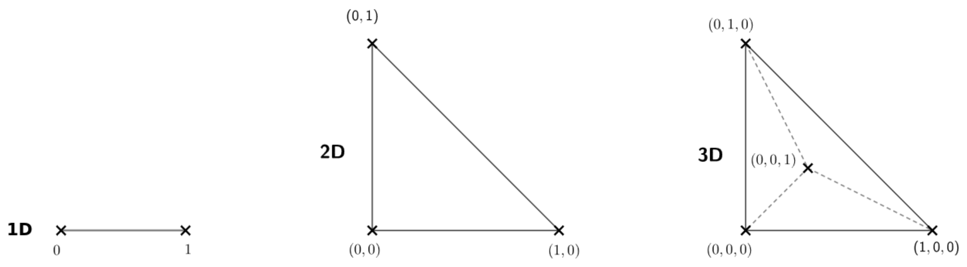


Figure 2.1: Example of reference Elements.

2.1.3.4 Mesh

In 1D, the construction of a mesh consists in creating a subdivision of the interval $[a,b]$. We can extend this definition in 2D and 3D by considering that a mesh is formed by a family of elements $\mathscr{T}_h = \{K_1,\dots,K_{N_e}\}$ (see Fig 2.2) where N_e is the number of elements.
In 2D, these elements can be triangles or rectangles. In 3D, they can be tetrahedrons, parallelepipeds or prisms.

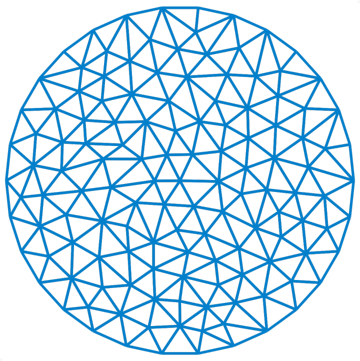


Figure 2.2: Example of a triangular mesh on a circles.

2.1.3.5 Geometric transformation

A mesh is generated by

- A reference element noted $\hat{K}$.

- A family of geometric transformations mapping $\hat{K}$ to the elements $K_1, \dots, K_{N_e}$. Thus, for a cell $K \in \mathcal{T}_h$, we denote T_K the geometric transformation mapping $\hat{K}$ to K :

$$T_K : \hat{K} \to K$$

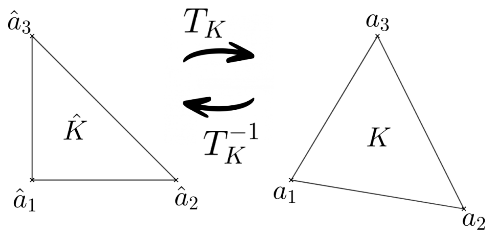


Figure 2.3: Geometric transformation applied to a triangle.

Let $(\hat{K}, \hat{\Sigma}, \hat{P})$ be the finite reference element with

- the nodes of the reference element $\hat{K}$: $\hat{\Sigma} = \{\hat{a}_1, \dots, \hat{a}_n\}$
- the local base functions of $\hat{K}$: $\{\hat{\psi}_1, \dots, \hat{\psi}_n\}$ (also called form functions)

So for each $K \in \mathcal{T}_h$, we consider a tuple $\{a_{K,1}, \dots, a_{K,n}\}$ (degrees of freedom) and the associated geometric transformation is defined by :

$$T_K : \hat{x} \mapsto \sum_{i=1}^{n} a_{K,i} \hat{\psi}_i(\hat{x})$$

In particular, we have

$$T_K(\hat{a}_i) = a_{K,i}, \quad i = 1, \dots, n$$

Remark. *In particular, if the form functions are affine, the geometric transformations will be too. This is an interesting property, as the gradient of these geometric transformations will be constant.*

Construction of the basis (φ_i) of V_h :
TO COMPLETE !

Remark. *In the following, we will assume that these transformations are C^1-diffeomorphisms (i.e. the transformation and its inverse are C^1 and bijective).*

2.1.4 Application to the Poisson problem

2 sous-sections ? -> Théorie -> Pratique
Ajouter formulation variationnel Poisson

Proposition 2.1 (Lax-Milgram)**.** *Let a be a continuous, coercive bilinear form on V and l a continuous, linear form on V. Then the variational problem has a unique solution $u \in V$.*
Moreover, if the bilinear form is symmetrical, u is a solution to the following minimization problem:

$$J(u) = \min_{v \in V} J(v), \quad J(v) = \frac{1}{2} a(v,v) - l(v)$$

It can then be shown that the Poisson problem with Dirichlet condition has a unique weak solution $u \in H_0^1(\Omega)$.
rajouter preuve
Rajouter : Calcul assemblage matrice dans le cas de ce problème ?
Rajouter : Convergence FEM standard !

2.2 ϕ-FEM

In this section, we will present the ϕ-FEM method. We will first present the context in which the method is used and its general principle, then go into a little more detail about the method in the case of the Poisson problem with Dirichlet condition. Finally, we will present the main numerical results of the reference article ?

2.2.1 Context and general principle of the method

The PhiFEM method is a new fictitious domain finite element method that does not require a mesh conforming to the real boundary. In the context of augmented surgery, this method presents a considerable advantage. During real-time simulation, the geometry (in our specific context, an organ such as the liver, for example) can deform over time. Methods such as standard FEM, which requires a mesh fitted to the boundary, necessitate a complete remeshing of the geometry at each time step (Figure 2.4). Unlike this type of method, ϕ-FEM requires only the generation of a single mesh : the mesh of a fictitious domain containing the entire geometry (Figure 2.5). As the boundary of the geometry is represented by a levelset function ϕ, only this function will change over time, which is a real time-saver.

Remark. *Note that changing the ϕ function creates new sets of cells, all of them described in the section ADD paragraph label antora.*

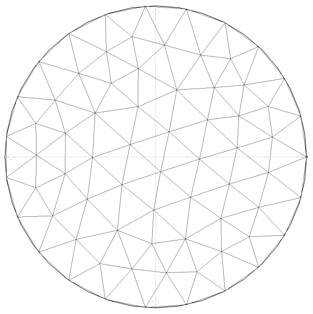


Figure 2.4: Standard FEM mesh example.

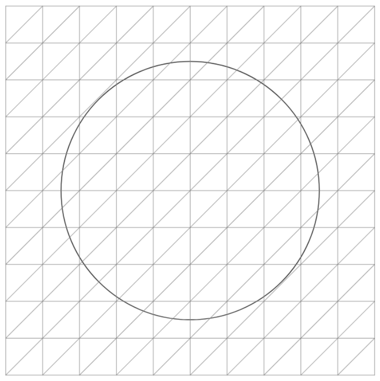


Figure 2.5: ϕ-FEM mesh example.

Remark. *For the purposes of this internship, the geometries considered are not organs (such as the liver), because these are complex geometries. We will be considering simpler geometries such as circles or squares. It is also important to note that the ϕ-FEM method has a considerable advantage: by constructing a fictitious mesh around the domain, we can generate a Cartesian mesh. This type of mesh can easily be represented by matrices, in the same way as images, hence the possibility of teaching these ϕ-FEM solutions to an FNO who generally works on images. Un article récent présente des résultats avec la combinaison de PhiFEM et d'un FNO sur des géométries d'ellipse + add ref si publié avant rendu.*

2.2.2 General presentation of the ϕ-FEM method

In this section, we consider the case of the Poisson problem with homogeneous Dirichlet condition ($g = 0$ on Γ). For the case of non-homogeneous Dirichlet conditions, we will give more details in Section 2.2.3. For more details on mesh assumptions, convergence results and finite element matrix condition number, please refer to [6]. Models ϕ-FEM for the Poisson problem with Neumann or mixed conditions (Dirichlet and Neumann) are presented in [5, 3]. For consideration of the elasticity problem, please refer to [3].

2.2.2.1 Description of spaces

As previously said, we will consider the Poisson-Dirichlet problem

$$\begin{cases} -\Delta u = f, & \text{in } \Omega, \\ u = g, & \text{on } \partial\Omega, \end{cases} \tag{2}$$

where the domain Ω and its boundary Γ are given by a level-set function ϕ such that

$$\Omega = \{\phi < 0\} \quad \text{and} \quad \Gamma = \{\phi = 0\}.$$

The level-set function ϕ is supposed to be known on $\mathbb{R}^d$, sufficiently smooth, and to behave near Γ as the signed distance to Γ (Figure 2.6).

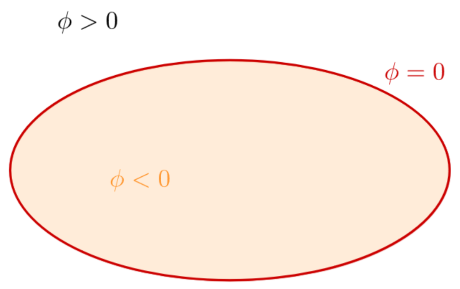


Figure 2.6: Definition of the level-set function.

Example. *If Ω is a circle of center A of coordinates (x_A, y_A) and radius r, the level-set function can be defined by*

$$\phi(x, y) = -r^2 + (x - x_A)^2 + (y - y_A)^2.$$

If Ω is an ellipse with center A of coordinates (x_A, y_A) and parameters (a, b), the level-set function can be defined by

$$\phi(x, y) = -1 + \frac{(x - x_A)^2}{a^2} + \frac{(y - y_A)^2}{b^2}.$$

We assume that Ω is inside a domain $\mathcal{O}$ and we introduce a simple quasi-uniform mesh $\mathcal{T}_h^{\mathcal{O}}$ on $\mathcal{O}$ (Figure 2.7).

We introduce now an approximation $\phi_h \in V_{h,\mathcal{O}}^{(l)}$ of ϕ given by $\phi_h = I_{h,\mathcal{O}}^{(l)}(\phi)$ where $I_{h,\mathcal{O}}^{(l)}$ is the standard Lagrange interpolation operator on

$$V_{h,\mathcal{O}}^{(l)} = \left\{ v_h \in H^1(\mathcal{O}) : v_{h|_T} \in \mathbb{P}_l(T)\ \forall T \in \mathcal{T}_h^{\mathcal{O}} \right\}$$

and we denote by $\Gamma_h = \{\phi_h = 0\}$, the approximate boundary of Γ (Figure 2.8).
We will consider $\mathcal{T}_h$ a sub-mesh of $\mathcal{T}_h^{\mathcal{O}}$ obtained by removing the elements located entirely outside Ω (Figure 2.8). To be more specific, $\mathcal{T}_h$ is defined by

$$\mathcal{T}_h = \left\{ T \in \mathcal{T}_h^{\mathcal{O}} : T \cap \{\phi_h < 0\} \neq \emptyset \right\}.$$

We denote Ω_h the domain covered by the $\mathcal{T}_h$ mesh (Ω_h will be slightly larger than Ω) and $\partial\Omega_h$ its boundary (Figure 2.8). The domain Ω_h is defined by

$$\Omega_h = \left(\cup_{T \in \mathcal{T}_h} T \right)^O.$$

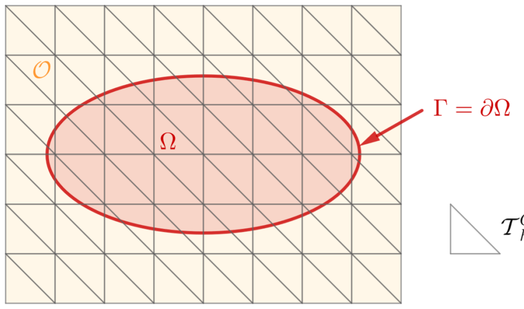


Figure 2.7: Fictitious domain.

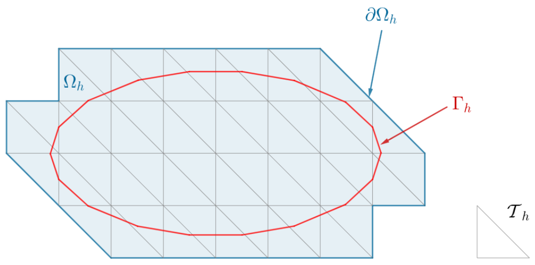


Figure 2.8: Domain considered.

Now, we can introduce $\mathcal{T}_h^\Gamma \subset \mathcal{T}_h$ (Figure 2.9) which contains the mesh elements cut by the approximate boundary $\Gamma_h = \{\phi_h = 0\}$, i.e.

$$\mathcal{T}_h^\Gamma = \{T \in \mathcal{T}_h : T \cap \Gamma_h \neq \emptyset\},$$

and $\mathcal{F}_h^\Gamma$ (Figure 2.10) which collects the interior facets of the mesh $\mathcal{T}_h$ either cut by Γ_h or belonging to a cut mesh element

$$\mathcal{F}_h^\Gamma = \{E \text{ (an internal facet of } \mathcal{T}_h) \text{ such that } \exists T \in \mathcal{T}_h : T \cap \Gamma_h \neq \emptyset \text{ and } E \in \partial T\}.$$

We denote by Ω_h^Γ the domain covered by the $\mathcal{T}_h^\Gamma$ mesh (Figure 2.9) and also defined by

$$\Omega_h^\Gamma = \left(\cup_{T \in \mathcal{T}_h^\Gamma} T \right)^O.$$

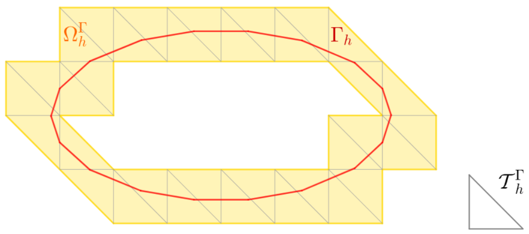


Figure 2.9: Boundary cells.

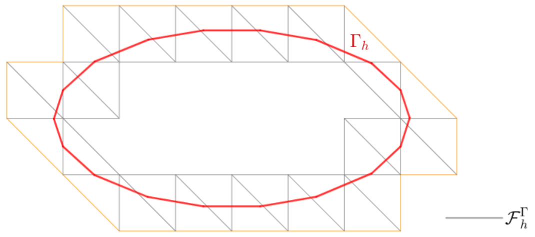


Figure 2.10: Boundary edges.

2.2.2.2 Description of the ϕ-FEM method

As with standard FEM, the general idea behind ϕ-FEM is to find a weak solution (i.e. a solution to the variational problem) to the considered problem (2). The main difference lies in the spaces considered. In fact, we are no longer looking to solve the problem on Ω (of boundary Γ) but on Ω_h (of boundary $\partial\Omega_h$). Since our boundary conditions are defined on Γ, we don't have a direct condition on the $\partial\Omega_h$ boundary, so we will have to add terms to the variational formulation of the problem, called stabilization terms.
Assuming that the right-hand side f is currently well-defined on Ω_h and that the solution u can be extended on Ω_h such that $-\Delta u = f$ on Ω_h, we can introduce a new unknown $w \in H^1(\Omega_h)$ such that $u = \phi w$ and the boundary condition on Γ is satisfied s(ince $\phi = 0$ on Γ). After an integration by parts, we have

$$\int_{\Omega_h} \nabla(\phi w)\cdot\nabla(\phi v) - \int_{\partial\Omega_h} \frac{\partial}{\partial n}(\phi w)\phi v = \int_{\Omega_h} f\phi v, \quad \forall v \in H^1(\Omega_h)$$

Remark. *Note that Ω_h is constructed using ϕ_h and therefore implicitly depends on ϕ.*

Given an approximation ϕ_h of ϕ on the mesh $\mathcal{T}_h$, as defined in Section ADD paragraph label antora, and a finite element space V_h on $\mathcal{T}_h$, we can then search for $w_h \in V_h$ such that

$$a_h(w_h, v_h) = l_h(v_h), \quad \forall v_h \in V_h.$$

We can consider the finite element space $V_h = V_h^{(k)}$ with

$$V_h^{(k)} = \left\{v_h \in H^1(\Omega_h) : v_{h|_T} \in \mathbb{P}_k(T)\ \forall T \in \mathcal{T}_h\right\}.$$

The bilinear form a_h and the linear form l_h are defined by

$$a_h(w,v) = \int_{\Omega_h} \nabla(\phi_h w)\cdot\nabla(\phi_h v) - \int_{\partial\Omega_h} \frac{\partial}{\partial n}(\phi_h w)\phi_h v + G_h(w,v)$$

and

$$l_h(v) = \int_{\Omega_h} f\phi_h v + G_h^{rhs}(v)$$

with

$$G_h(w,v) = \sigma h \sum_{E\in\mathcal{F}_h^\Gamma} \int_E \left[\frac{\partial}{\partial n}(\phi_h w)\right]\left[\frac{\partial}{\partial n}(\phi_h v)\right] + \sigma h^2 \sum_{T\in\mathcal{T}_h^\Gamma} \int_T \Delta(\phi_h w)\Delta(\phi_h v)$$

and

$$G_h^{rhs}(v) = -\sigma h^2 \sum_{T\in\mathcal{T}_h^\Gamma} \int_T f\Delta(\phi_h v).$$

with σ an independent parameter of h, which we'll call the stabilization parameter.

Remark. *Note that $[\cdot]$ is the jump on the interface E defined by*

$$\left[\frac{\partial}{\partial n}(\phi_h w)\right] = \nabla(\phi_h w)^+\cdot n - \nabla(\phi_h w)^-\cdot n$$

with n is the unit normal vector outside E.

2.2.3 Some details on ϕ-FEM

2.2.3.1 Stabilization terms

As introduced previously, the stabilization terms are intended to reduce the errors created by the "fictitious" boundary, but they also have the effect of ensuring the correct condition number of the finite element matrix and permitting to restore the coercivity of the bilinear scheme.
The first term of $G_h(w,v)$ defined by

$$\sigma h \sum_{E\in\mathscr{F}_h^\Gamma} \int_E \left[\frac{\partial}{\partial n}(\phi_h w)\right]\left[\frac{\partial}{\partial n}(\phi_h v)\right]$$

is a first-order stabilization term. This stabilization term is based on [1]. It also ensures the continuity of the solution by penalizing gradient jumps.
By substracting $G_h^{rhs}(v)$ from the second term of $G_h(w,v)$, i.e.

$$\sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T \Delta(\phi_h w)\Delta(\phi_h v) + \sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T f\Delta(\phi_h v),$$

which can be rewritten as

$$\sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T \left(\Delta(\phi_h w)+f\right)\Delta(\phi_h v),$$

we recognize the strong formulation of the Poisson problem. This second-order stabilization term penalizes the scheme by requiring the solution to verify the strong form on Ω_h^Γ. In fact, this term cancels out if $\phi_h w$ is the exact solution of the Poisson problem under consideration.

2.2.3.2 Non-homogeneous case

In the case of a non-homogeneous Dirichlet condition, we want to impose $u=g$ on Γ. To do this, we will consider 2 approaches introduced in [3] and presented below:

- **Direct method :** In this method, we must suppose that g is currently given over the entire Ω_h and not just over Γ. We can then write the solution u as

  $$u=\phi w+g, \text{ on } \Omega_h.$$

  It can then be injected into the weak formulation of the homogeneous problem and we can then search for w_h on Ω_h such that

  $$\int_{\Omega_h}\nabla(\phi_h w)\nabla(\phi_h v) - \int_{\partial\Omega_h}\frac{\partial}{\partial n}(\phi_h w)\phi_h v + G_h(w,v) = \int_{\Omega_h} f\phi_h v \\ - \int_{\Omega_h}\nabla g\nabla(\phi_h v) + \int_{\partial\Omega_h}\frac{\partial g}{\partial n}\phi_h v + G_h^{rhs}(v),\ \forall v\in\Omega_h$$

with

$$G_h(w,v) = \sigma h \sum_{E\in\mathscr{F}_h^\Gamma} \int_E \left[\frac{\partial}{\partial n}(\phi_h w)\right]\left[\frac{\partial}{\partial n}(\phi_h v)\right] + \sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T \Delta(\phi_h w)\Delta(\phi_h v)$$

and

$$G_h^{rhs}(v) = -\sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T f\Delta(\phi_h v) - \sigma h \sum_{E\in\mathscr{F}_h^\Gamma} \int_E \left[\frac{\partial g}{\partial n}\right]\left[\frac{\partial}{\partial n}(\phi_h v)\right] - \sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T \Delta g \Delta(\phi_h v)$$

- **Dual method :** We now assume that g is defined on Ω_h^Γ and not on Ω_h. We then introduce a new unknown p on Ω_h^Γ in addition to the unknown u on Ω_h and so we aim to impose
$$u = \phi p + g, \text{ on } \Omega_h^\Gamma.$$
So we look for u on Ω_h and p on Ω_h^Γ such that

$$\int_{\Omega_h} \nabla u \nabla v - \int_{\partial\Omega_h} \frac{\partial u}{\partial n} v + \frac{\gamma}{h^2} \sum_{T\in\mathscr{T}_h^\Gamma} \int_T \left(u - \frac{1}{h}\phi p\right)\left(v - \frac{1}{h}\phi q\right) + G_h(u,v) = \int_{\Omega_h} f v$$
$$+ \frac{\gamma}{h^2} \sum_{T\in\mathscr{T}_h^\Gamma} \int_T g\left(v - \frac{1}{h}\phi q\right) + G_h^{rhs}(v), \ \forall v \text{ on } \Omega_h,\ q \text{ on } \Omega_h^\Gamma.$$

with γ an other positive stabilization parameter,

$$G_h(u,v) = \sigma h \sum_{E\in\mathscr{F}_h^\Gamma} \int_E \left[\frac{\partial u}{\partial n}\right]\left[\frac{\partial v}{\partial n}\right] + \sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T \Delta u \Delta v$$

and

$$G_h^{rhs}(v) = -\sigma h^2 \sum_{T\in\mathscr{T}_h^\Gamma} \int_T f \Delta v.$$

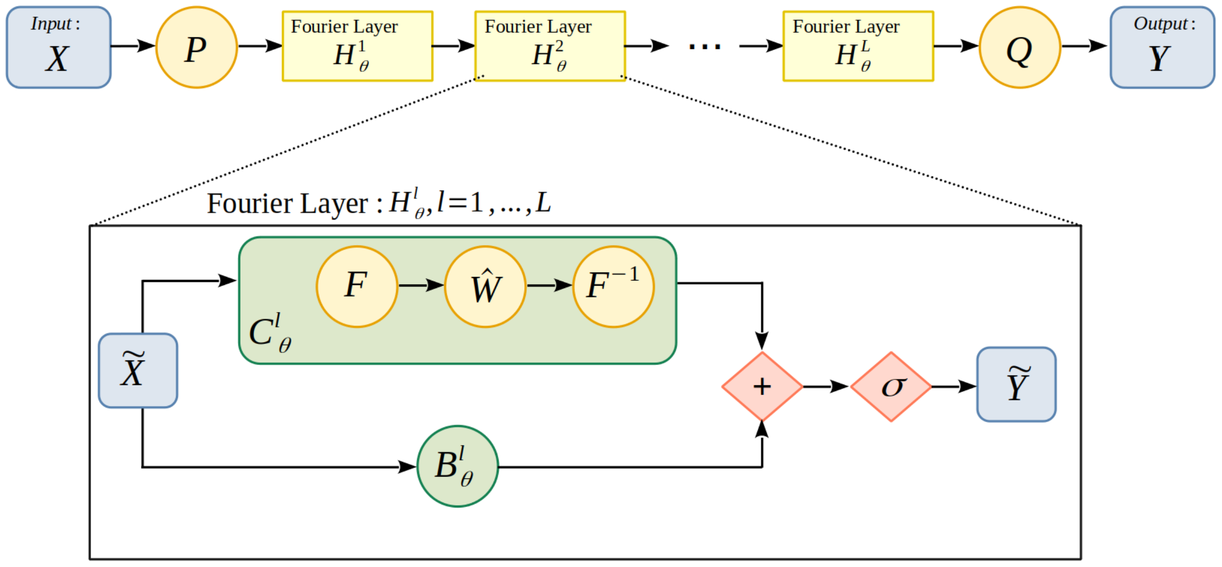
Remark. *The factors $\frac{1}{h}$ and $\frac{1}{h^2}$ control the condition number of the finite element matrix. For more details, please refer to the article [5].*

3 Fourier Neural Operator (FNO)

Est-ce que je rajouter réseau multi-perceptron et PINNs ? Si oui section "Neural Networks" puis sous-sections FNO,MultiPerceptron,PINNs
We will now introduce Fourier Neural Operators (FNO). For more information, please refer to the following article [7]. + add article Killian
In image treatment, we call image tensors of size $ni \times nj \times nk$, where $ni \times nj$ corresponds to the image resolution and nk corresponds to its number of channels. For example, an RGB (Red Green Blue) image has $nk = 3$ channels. We choose here to present the FNO as an operator acting on discrete images. Reference articles present it in its continuous aspect, which is an interesting point of view. Indeed, it is thanks to this property that it can be trained/evaluated with images of different resolutions.
The FNO methodology creates a relationship between two spaces from a finite collection of observed input-output pairs. Est-ce que je gardes cette phrase ?
A rajouter : implémentation effectuée/fournie par Vincent Vigon + on ne trouvera pas ici de test de variation des paramètres (modes, width...)

3.1 Architecture of the FNO

The following figure (Figure 3.1) describes the FNO architecture in detail:



Input : X
P
Fourier Layer H^1_θ
Fourier Layer H^2_θ
...
Fourier Layer H^L_θ
Q
Output : Y
Fourier Layer : $H^l_\theta, l=1,...,L$
$\widetilde{X}$
F
$\hat{W}$
F^{-1}
C^l_θ
B^l_θ
+
σ
$\widetilde{Y}$

Figure 3.1: Architecture of the FNO.

The architecture of the FNO is as follows:

$$G_\theta = Q \circ \mathcal{H}^L_\theta \circ \cdots \circ \mathcal{H}^1_\theta \circ P$$

We'll now describe the composition of the Figure 3.1 in a little more detail :

- We start with input X of shape (batch_size, height, width, nb_channels) with batch_size the number of images to be processed at the same time, height and width the dimensions of the images and nb_channels the number of channels. Simplify by (bs,ni,nj,nk).
- We perform a P transformation in order to move to a space with more channels. This step enables the network to build a sufficiently rich representation of the data. For example, a Dense layer (also known as fully-connected) can be used.
- We then apply L Fourier layers, noted $\mathscr{H}_\theta^l$, $l = 1, \dots, L$, whose specifications will be detailed in Section 3.2.
- We then return to the target dimension by performing a Q transformation. In our case, the number of output channels is 1.
- We then obtain the output of the Y model of shape (bs,ni,nj,1).

3.2 Fourier Layer structure

Each Fourier layer is divided into two sublayers:

$$\tilde{Y} = \mathscr{H}_\theta^l(\tilde{X}) = \sigma\left(\mathscr{C}_\theta^l(\tilde{X}) + \mathscr{B}_\theta^l(\tilde{X})\right)$$

where

- $\tilde{X}$ corresponds to the input of the current layer and $\tilde{Y}$ to the output.
- σ is an activation function. For $l = 1, \dots, L-1$, we'll take the activation function ReLU (Rectified Linear Unit) and for $l = L$ we'll take the activation function GELU (Gaussian Error Linear Units).

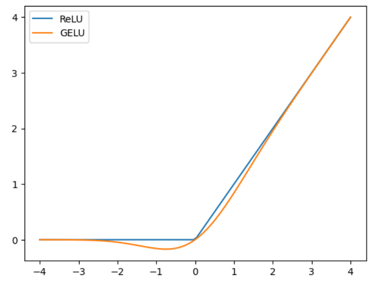


Figure 3.2: Activation functions used.

- $\mathscr{C}_\theta^l$ is a convolution layer where convolution is performed by FFT (Fast Fourier Transform). For more details, see Section 3.2.1.
- $\mathscr{B}_\theta^l$ is the "bias-layer". For more details, see Section 3.2.2.

3.2.1 Convolution sublayer

Each $\mathscr{C}_\theta^l$ convolution layer contains a trainable kernel $\hat{W}$ and performs the transformation

$$\mathscr{C}_\theta^l(X) = \mathscr{F}^{-1}(\mathscr{F}(X)\cdot\hat{W})$$

where $\mathscr{F}$ corresponds to the 2D Discrete Fourier Transform (DFT) on a $ni \times nj$ resolution grid and

$$(Y\cdot\hat{W})_{ijk} = \sum_{k'} Y_{ijk'}\hat{W}_{ijk'}$$

In other words, this transormation is applied channel by channel.

Remark. *An image is fundamentally a signal. Just as 1D signals show changes in amplitude (sound) over time, 2D signals show variations in intensity (light) over space. The Fourier transform allows us to move from the spatial or temporal domain into the frequency domain. In a sound signal (1D signal), low frequencies represent low-pitched sounds and high frequencies represent high-pitched sounds. In the case of an image (2D signal), low frequencies represent large homogeneous surfaces and blurred parts, while high frequencies represent contours, more generally abrupt changes in intensity and, finally, noise.*

The 2D DFT is defined by :

$$\mathscr{F}(X)_{ijk} = \frac{1}{ni}\frac{1}{nj}\sum_{i'=0}^{ni-1}\sum_{j'=0}^{nj-1} X_{i'j'k}e^{-2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)}$$

The inverse of the 2D DFT is defined by :

$$\mathscr{F}^{-1}(X)_{ijk} = \sum_{i'=0}^{ni-1}\sum_{j'=0}^{nj-1} X_{i'j'k}e^{2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)}$$

We can easily show that $\mathscr{F}$ is the reciprocal function of $\mathscr{F}^{-1}$. We have

$$\begin{aligned}
\mathscr{F}^{-1}(\mathscr{F}(X))_{ijk} &= \sum_{i'=0}^{ni-1}\sum_{j'=0}^{nj-1}\mathscr{F}(X)_{i'j'k}e^{2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)} \\
&= \frac{1}{ni}\frac{1}{nj}\sum_{i'j'}\sum_{i''j''} X_{i''j''k}e^{-2\sqrt{-1}\pi\left(\frac{i'i''}{ni}+\frac{j'j''}{nj}\right)}e^{2\sqrt{-1}\pi\left(\frac{ii'}{ni}+\frac{jj'}{nj}\right)} \\
&= \frac{1}{ni}\frac{1}{nj}\sum_{i''j''} X_{i''j''k}\sum_{i'j'} e^{2\sqrt{-1}\pi\frac{i'}{ni}(i-i'')}e^{2\sqrt{-1}\pi\frac{j'}{nj}(j-j'')}
\end{aligned}$$

Let

$$S = \sum_{i'j'} e^{2\sqrt{-1}\pi\frac{i'}{ni}(i-i'')}e^{2\sqrt{-1}\pi\frac{j'}{nj}(j-j'')}$$

Thus

- If $(i,j) = (i'', j'')$: $S = \sum_{i',j'} 1 = ni \times nj$
- If $(i,j) \neq (i'', j'')$:

$$\begin{aligned} S &= \sum_{i'} \left(e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')}\right)^{i'} \sum_{j'} \left(e^{\frac{2\sqrt{-1}\pi}{nj}(j-j'')}\right)^{j'} \\ &= \frac{1-\left(e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')}\right)^{ni}}{1-e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')}} \times \frac{1-\left(e^{\frac{2\sqrt{-1}\pi}{nj}(j-j'')}\right)^{nj}}{1-e^{\frac{2\sqrt{-1}\pi}{nj}(j-j'')}} \\ &= \frac{1-e^{2\sqrt{-1}\pi(i-i'')}}{1-e^{\frac{2\sqrt{-1}\pi}{ni}(i-i'')}} \times \frac{1-e^{2\sqrt{-1}\pi(j-j'')}}{1-e^{\frac{2\sqrt{-1}\pi}{ni}(j-j'')}} = 0 \end{aligned}$$

  as the sum of a geometric sequence.

We deduce that

$$\mathscr{F}^{-1}(\mathscr{F}(X))_{ijk} = \frac{1}{ni}\frac{1}{nj} \times ni \times nj \times X_{ijk} = X_{ijk}$$

And finally $\mathscr{F}$ is the reciprocal function of $\mathscr{F}^{-1}$.
For more details about the Convolution sublayer, see Section 3.3.

3.2.2 Bias subLayer

The bias layer is a 2D convolution with a kernel size of 1. This means that it only performs matrix multiplication on the channels, but pixel by pixel. In other words, it mixes channels via a kernel, but does not allow interaction between pixels.
Precisly,

$$\mathscr{B}^l_\theta(X)_{ijk} = \sum_{k'} X_{ijk} W_{k'k} + B_k$$

3.3 Some details on the convolution sublayer

In this section, we will specify some details for the convolution layer.

3.3.1 Border issues

Let $W = \mathscr{F}^{-1}(\hat{W})$, we have :

$$\mathscr{C}^l_\theta(\tilde{X}) = \mathscr{F}^{-1}\left(\mathscr{F}(X) \cdot \hat{W}\right) = \tilde{X} \star W$$

with

$$(\tilde{X} \star W)_{ij} = \sum_{i'j'} \tilde{X}_{i-i'[ni],j-j'[nj]} W_{i'j'}$$

In other words, multiplying in Fourier space is equivalent to performing a $\star$ circular convolution in real space. But these modulo operations are only natural for periodic images, which is not our case. A compléter (expliquer padding par ex)

3.3.2 FFT

To speed up computations, we will use the FFT (Fast Fourier Transform). The FFT is a fast algorithm to compute the DFT. It is recursive : The transformation of a signal of size N is make from the decomposition of two sub-signals of size $N/2$. The complexity of the FFT is $N\log(N)$ whereas the natural algorithm, which is a matrix multiplication, has a complexity of N^2.

3.3.3 Real DFT

In reality, we'll be using a specific implementation of FFT, called RFFT (Real Fast Fourier Transorm).In fact, for $\mathscr{F}^{-1}(A)$ to be real if A is a complex-valued matrix, it is necessary that A respects the Hermitian symmetry:

$$A_{i,nj-(j+1)} = \bar{A}_{i,j}$$

In our case, we want $\mathscr{C}_\theta^l(X)$ to be a real image, so $\mathscr{F}(X)\cdot\hat{W}$ must verify Hermitian-symmetry. To do this, we only need to collect half of the Discrete Fourier Coefficients (DFC) and the other half will be deduced by Hermitian symmetry. More precisely, using the specific RFFT implementation, the DFCs are stored in a matrix of size $(ni, nj//2+1)$. Multiplication can then be performed by the $\hat{W}$ kernel, and when the inverse RFFT is performed, the DFCs will be automatically symmetrized. So the Hermitian symmetry of $\mathscr{F}(X)\cdot\hat{W}$ is verified and $\mathscr{C}_\theta^l(X)$ is indeed a real image.
To simplify, let's assume nk=1. Here is a diagram describing this idea:

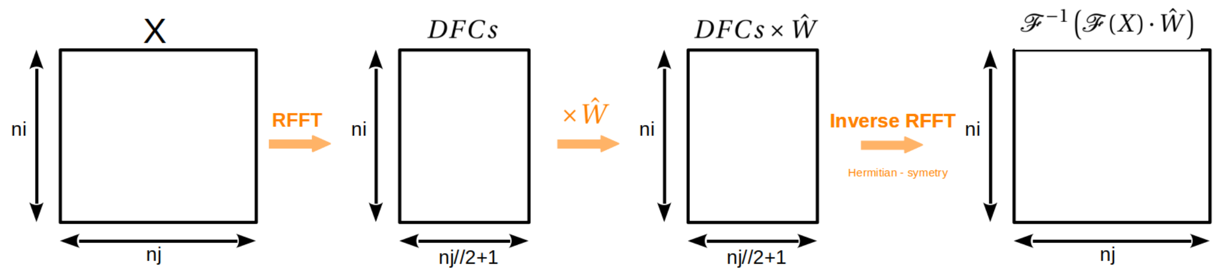


Figure 3.3: RFFT with Hermitian-symmetry scheme.

3.3.4 Low pass filter

When we perform a DFT on an image, the DFCs related to high frequencies are in practice very low. This is why we can easily filter an image by ignoring these high frequencies, i.e. by truncating the high Fourier modes. In fact, eliminating the higher Fourier modes enables a kind of regularization that helps the generalization. So, in practice, it's sufficient to keep only the DFCs corresponding to low frequencies. Typically, for images of resolution 32×32 to 128×128, we can keep only the 20×20 DFCs associated to low frequencies.
Here is a representation of this idea in 1D :

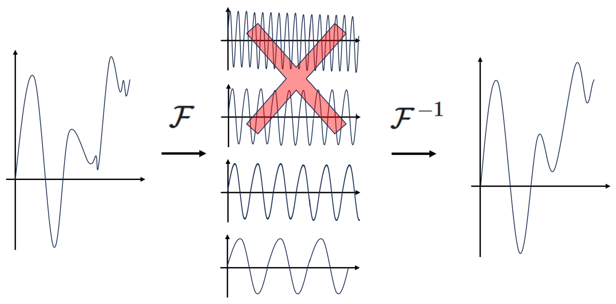


Figure 3.4: Low pass filter.

3.3.5 Global aspect of the FNO

Classical Convolutional Neural Networks (CNN) use very small kernels (typically 3×3). This operation only has a local effect, and it's the sequence of many convolutions that produces more global effects.
In addition, CNNs often use max or mean-pooling layers, which process the image on several scales. Max-pooling (respectively mean-pooling) consists in slicing the image into small pieces of size $n \times n$, then choosing the pixel with the highest value (respectively the average of the pixels) in each of the small pieces. In most cases, $n = 2$ is used, which divides the number of pixels by 4.
The FNO, on the other hand, uses a $\hat{W}$ frequency kernel and $W = \mathscr{F}^{-1}(\hat{W})$ has full support. For this reason, the effect is immediately non-local. As a result, we can use less layers and we don't need to use a pooling layer.

3.4 Application

Dans notre cas, on souhaite apprendre au FNO à prédire des solutions d'EDP. Plus précisément, on souhaite que le réseau soit capable de prédire la solution à partir d'un terme source f. enrichissement des données, grilles régulières
rajouter schéma

4 Correction

PLAN Correction : expliquer le contexte général (on souhaite corriger la sortie d'un FNO et faire référence à 3.4 Application où il faut mettre un schéma) puis expliquer en intro le cheminement (résultats sr sol analytique -> sol ana + pert -> FNO -> pb FNO -> test Legendre (car P10) -> rajouter trop d'erreurs donc Multiperceptron -> pb dérivées donc PINNS (on pourrait inclure les dérivées dans la loss)) + expliquer que le but est de considérer PhiFEM mais que certains résultats théoriques et pratiques sont avec FEM standard rajouter si c'est fait résultat correction avec $\tilde{\phi}$ sol phifem dans intro.
rajouter rq quelque part avec librairies utilisées : FEniCS, Tensorflow, Pytorch !
Pourquoi entrainement maillage fin puis correction maillage grossier ?

1. **Présentation des différentes méthodes de Correction considérées :** présentation des différentes méthodes de correction (+ résultats analytiques en annexe ? comme "papier" rehaussement etc) + ajouter schéma ?

   rq "précise" sur en quoi ça aide le solveur de lui fournir une solution proche de la solution exacte (fcts de base ?)

2. **Présentation des problèmes considérés :** Comme dit précédemment, on ne considérera ici que le pb de Poisson... + expliquer domaines (cercle et carré) + expliquer sol considérées (u trigo, f gaussienne => sol sur-raffinée, u polynomial pour PINNs) A faire en dernier en ne mettant que les trucs qu'on a considéré ensuite !!

3. **Différents résultats de correction**

   - **Correction sur solution exacte :** Résultat sur sol exacte avec FEM et PhiFEM
   - **Correction sur solution perturbée :**
     - solution perturbée manuellement
     - correction sur PhiFEM : Pas encore tester mais intéressant à tester !!!
   - **Correction avec FNO :** loss sur w (ou phi w)... + résultats Phifem P2 puis Legendre P10 (ajoute trop d'erreurs ?)
   - **Correction avec d'autres réseaux :** expliquer pourquoi on fait ça (comprendre si la correction fonctionne sur des réseaux plus simple et donc ça pourrait confirmer qu'il y a bien un problème au niveau de la perturbation créé par le FNO) + rajouter première implémentation tensorflow puis utilisation d'un travail en cours (avec pytorch)
     - **Multiperceptron :** présentation rapide avec schéma (+références) + résultats obtenus + expliquer qu'on doit rajouter l'apprentissage des dérivées car elles sont mauvaises et utilisées dans le solveur (dans la formulation variationnelle).

- **PINNs :** présentation rapide avec schéma ? + résultats obtenus pour $f = 1$ (+rajouter rq que par manque de temps, on n'a pas trouvé la bonne architecture du PINNs pour apprendre la solution trigo et donc on n'a considéré pour l'instant que la correction pour $f = 1$)

+ rajouter rq ou dans conclusion pour expliquer qu'on pourrait également tester de faire varier le type de solveur matricielle dans les solveurs pour la correction !

4.1 Presentation of the different problem considered

A faire à la fin !

4.2 Presentation of the different correction methods considered

ajouter formulation variationnelle : FEM + PhiFEM ?
Here we are given $\tilde{\phi}$ an "initial" solution to the problem under consideration, i.e. a solution that has not yet been corrected. This may be a perturbed analytic solution, a ϕ-FEM solution, or a solution predicted by a neural network (such as an FNO, a Multi-perceptron network or PINNs, for example). The aim is to reinject this solution into a new problem in order to improve the accuracy of the solution. To achieve this, we consider 3 types of correction: correction by addition (Section 4.2.1), correction by multiplication (Section 4.2.2) and correction by multiplication on an elevated problem (Section 4.2.3).

4.2.1 Correction by adding

In this first method, we will try to approximate the solution obtained $\tilde{\phi}$ to the exact solution by completing the difference between the two, which is what we will call correction by adding. To do this, we will consider

$$\tilde{u} = \tilde{\phi} + C$$

and we want to find $C : \Omega \to \mathbb{R}^d$ solution to the problem

$$\begin{cases} -\Delta \tilde{u} = f, & \text{on } \Omega, \\ \tilde{u} = g, & \text{in } \Gamma. \end{cases}$$

Remark. *Note that this problem is in fact equivalent to the initial add ref problem. We only hope that the approximate solution* $tilde u$ *obtained is more accurate than the approximate solution u obtained by solving the initial problem.*

Rewriting the problem, we seek to find $C : \Omega \to \mathbb{R}^d$ solution to the problem

$$\begin{cases} -\Delta C = \tilde{f}, & \text{on } \Omega, \\ C = 0, & \text{in } \Gamma. \end{cases} \qquad (\mathscr{C}_+)$$

with $\tilde{f} = f + \Delta\tilde{\phi}$.

4.2.2 Correction by multiplying

In this second method, we try to approach the exact solution in a different way. In fact, we want to bring the factor between the $\tilde{\phi}$ solution and the solution of the corrected problem closer to 1. In other words, by considering

$$\tilde{u} = \tilde{\phi}C,$$

we try to bring $C = \frac{\tilde{u}}{\tilde{\phi}}$ closer to 1 (for $\tilde{\phi} \neq 0$). This type of correction is called correction by multiplying.
So we're looking for $C : \Omega \to \mathbb{R}^d$ solution to the problem

$$\begin{cases} -\Delta(\tilde{\phi}C) = f & \Omega \\ C = 1 & \Gamma \end{cases} \tag{$\mathscr{C}_\times$}$$

Remark. *In the same way as for correction by adding, we note that this problem is equivalent to the initial add ref problem.*

4.2.3 Correction by multiplying on an elevated problem

We now introduce a third correction method, which we'll call multiplication correction on an elevated problem. This method is in fact very similar to the previous one (correction by multiplication), except that we are no longer trying to correct the same problem.
The initial modified problem, which we now consider, consists in finding $u : \Omega \to \mathbb{R}^d$ such that

$$\begin{cases} -\Delta\hat{u} = f, & \text{in } \Omega, \\ \hat{u} = g + m, & \text{on } \Gamma, \end{cases} \tag{$\mathscr{P}^{\mathscr{M}}$}$$

with $\hat{u} = u + m$ and m a constant.
ajouter schéma (à gauche une solution initiale simple et à droite la solution rehaussée.)
We then apply the same multiplication correction method, but this time on the modified problem, which has been elevated by a constant m. We then consider

$$\tilde{u} = \hat{\phi}C$$

avec

$$\hat{\phi} = \tilde{\phi} + m$$

and so we look for $C : \Omega \to \mathbb{R}^d$ solution to the problem

$$\begin{cases} -\Delta(\hat{\phi}C) = f & \Omega \\ C = 1 & \Gamma \end{cases} \tag{$\mathscr{C}_\times^{\mathscr{M}}$}$$

Remark. *Note that if the problem is sufficiently elevated for its solution to be strictly positive, the operation of bringing $C = \frac{\tilde{u}}{\hat{\phi}}$ closer to 1 doesn't pose a problem (since in this case $\hat{\phi} \neq 0$). Moreover, we can easily return to the original problem by subtracting m from $\tilde{u}$. In this way, by* *correcting the elevated problem by multiplication, we can easily return to correcting the initial problem by multiplication. En annexe, on a un document expliquant l'intérêt de rehausser le problème +add ref. rajouter également document qui explique qu'au final on s'est rendu compte que correction par multiplication sur problème rehaussé $\iff$ correction par addition + ajouter document*

4.3 Different correction results

As explained above, we wish to combine ϕ-FEM and FNO in order to predict the solution of the Poisson problem as accurately as possible. In this section, we present various results obtained using the 3 correction methods presented in the previous section (Section 4.2). It is important to note that, for practical purposes, almost all the following results obtained with ϕ-FEM will be compared with those obtained with the standard FEM method.
We'll start by presenting the results obtained on an analytical solution (Section 4.3.1). We'll consider here the "initial" solution $\tilde{\phi}$, which we'll inject into the correction problems, as the analytical solution of the problem. This first step simply enables us to check that, by supplying the exact solution directly to the correction solvers, they are indeed reduced to machine errors.
Next, in order to verify that correction solvers can improve accuracy when providing a solution close to the exact solution, we will consider the case of so-called "disturbed" solutions (Section 4.3.2). This step will also provide us with a basis for further work, giving us an idea of what we can expect in terms of neural network output correction.
Finally, we'll consider the case of neural networks with an FNO in Section 4.3.3 and then with a multi-perceptron network and PINNs in Section 4.3.4. The reasons for considering other neural networks will be explained in detail in these sections.

4.3.1 Correction on exact solution

4.3.2 Correction on disturbed solution

4.3.2.1 Manually perturbed solution

4.3.2.2 correction on PhiFEM

4.3.3 Correction with FNO

4.3.4 Correction with other networks

4.3.4.1 Multiperceptron

4.3.4.2 PINNs

5 Conclusion

Penser à parler de la thèse : sujet etc...
TO COMPLETE !

Bibliography

References

[1] Erik Burman. "Ghost penalty". In: *Comptes Rendus. Mathématique* 348.21 (2010), pp. 1217–1220. ISSN: 1778-3569. DOI: 10.1016/j.crma.2010.10.006. URL: http://www.numdam.org/articles/10.1016/j.crma.2010.10.006/ (visited on 07/26/2023).

[2] Erik Burman et al. "CutFEM: Discretizing geometry and partial differential equations". In: *International Journal for Numerical Methods in Engineering* 104.7 (2015). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.4823, pp. 472–501. ISSN: 1097-0207. DOI: 10.1002/nme.4823. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.4823 (visited on 07/26/2023).

[3] Stéphane Cotin et al. "ϕ-FEM: an efficient simulation tool using simple meshes for problems in structure mechanics and heat transfer". In: ().

[4] Michel Duprez, Vanessa Lleras, and Alexei Lozinski. "ϕ-FEM: an optimally convergent and easily implementable immersed boundary method for particulate flows and Stokes equations". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 57.3 (May 2023), pp. 1111–1142. ISSN: 2822-7840, 2804-7214. DOI: 10.1051/m2an/2023010. URL: https://www.esaim-m2an.org/10.1051/m2an/2023010 (visited on 07/26/2023).

[5] Michel Duprez, Vanessa Lleras, and Alexei Lozinski. "A new ϕ-FEM approach for problems with natural boundary conditions". In: *Numerical Methods for Partial Differential Equations* 39.1 (2023). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/num.22878, pp. 281–303. ISSN: 1098-2426. DOI: 10.1002/num.22878. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/num.22878 (visited on 07/26/2023).

[6] Michel Duprez and Alexei Lozinski. "ϕ-FEM: A Finite Element Method on Domains Defined by Level-Sets". In: *SIAM Journal on Numerical Analysis* 58.2 (Jan. 2020). Publisher: Society for Industrial and Applied Mathematics, pp. 1008–1028. ISSN: 0036-1429. DOI: 10.1137/19M1248947. URL: https://epubs.siam.org/doi/10.1137/19M1248947 (visited on 07/26/2023).

[7] Zongyi Li et al. *Fourier Neural Operator for Parametric Partial Differential Equations*. May 16, 2021. DOI: 10.48550/arXiv.2010.08895. arXiv: 2010.08895[cs,math]. URL: http://arxiv.org/abs/2010.08895 (visited on 07/26/2023).

[8] Nicolas Moës and Ted Belytschko. "X-FEM, de nouvelles frontières pour les éléments finis". In: *Revue Européenne des Éléments Finis* 11.2 (Jan. 2002), pp. 305–318. ISSN: 1250-6559. DOI: 10.3166/reef.11.305-318. URL: https://www.tandfonline.com/doi/full/10.3166/reef.11.305-318 (visited on 07/26/2023).