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: August 17, 2023

Contents

1 Introduction

This end of study internship is a 2nd year internship in the CSMI Master ("Calcul Scientifique et Mathématiques 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). This project was followed by a 6-month internship whose main objective was to correct the output of a Fourier Neural Operator (FNO) by a solver using the ϕ-FEM method.

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 [4] or XFEM [14], 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 [8]. But the method has been extended to Neumann conditions [7] and then to solve various mechanical problems, including linear elasticity [5, Chapter 2] and heat transfer problems [5, Chapter 5].

1.2 Presentation of the team

Created in January 2021 within ICube laboratory at the University of Strasbourg, MLMS[1] ("Machine Learning, Modélisation et Simulation") team is interested in data, models and simulations for medical science and human motion. It brings together computer scientists, mathematicians, bio-mechanicians, and neuroscientists to develop functional, physical, and

[1] MLMS : https://mlms.icube.unistra.fr/en/index.php/Presentation

geometric models around a transverse axis "Assistance to medical interventions by computer". MLMS hosts the MIMESIS[2] project-team as a sub-team. The MIMESIS research team aims at creating real-time digital twins of an organ, with main application domains as surgical training and surgical guidance during complex interventions. In 2023, a new inria team NECTARINE will be created within MLMS, who will focus on scientific challenges related to neuro-stimulation in the clinical context.
MIMESIS, directed by Stéphane Cotin, is a joint Inria[3] ("Institut national de recherche en sciences et technologies du numérique") and CNRS[4] ("Centre national de la recherche scientifique") Research Team. 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.

1.3 Objectives

The main objective of the internship was to combine finite element methods and Machine Learning in order to solve the Poisson problem with Dirichlet condition. More precisely, we want to train a neural network called Fourier Neural Network (FNO) [10] to predict the solutions of a PDE for a given problem family (i.e. a "type" of source term). This neural network is trained with a data set consisting of the ϕ-FEM solutions of the problems considered. The predictions of this neural network will then be fed back into a finite element solver to apply a correction to improve the accuracy of the solution : this was the subject covered during the internship. The finite element methods considered will be presented in Section 2 and the FNO in Section 3.
It is important to note that 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 (especially FNO).
To be more precise, we will test different correction methods (presented in Section 4.2) on different problems (presented in Section 4.1) which will enable us to use the network prediction to help the solver get as close as possible to the solution. We will start by testing these different types of solver on an analytical solution (Section 4.3.1), then on a "manually perturbed" solution (Section 4.3.2) and finally on a ϕ-FEM solution (Section 4.3.3).
After testing the various types of correction on the previous test cases, we'll apply these same methods to the prediction of an FNO (Section 4.3.4). The main objective is to enable the combination of FNO and correction to be more accurate than the conventional ϕ-FEM solver. By first testing the different corrections on the previous test cases, we hope to get an idea of the order of errors to be expected. During the course of the internship, we realized that the

[2]MIMESIS : https://mimesis.inria.fr/
[3]Inria : https://www.inria.fr/fr
[4]CNRS : https://www.cnrs.fr/fr

results obtained on the FNO did not correspond to the expected analytical results. For this reason, other types of neural networks were considered, namely multi-perceptron networks (Section 4.3.5.1) and PINNs (Section 4.3.5.2), with the aim of checking whether the results obtained are related to the use of the FNO.

1.4 Deliverables

In the context of the internship, the following deliverables are provided:

- a weekly tracking report, written in French, was produced as the internship progressed, listing the objectives and results for each week.
- a github repository containing all the code allowing to reproduce the results presented in this report, as well as the documents written during the internship. The codes have been implemented in Python: for the finite element solvers, we'll be using the FEniCS library, and for the neural network implementation, we'll be using Tensorflow and Pytorch. ajouter les codes en ligne
- an online report generated with a tool called antora[5]. A continuous integration has been set up on github to execute a python code for each new push, enabling the latex file to be converted directly into this antora documentation.
- a code documentation has also been set up with sphinx[6].

[5]Antora : https://antora.org/
[6]Sphinx : https://www.sphinx-doc.org/en/master/

2 Finite Element Methods (FEMs)

In the following, we will consider the Poisson problem with Dirichlet condition (homogeneous or non-homogeneous):
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).

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). For more information, please refer to [15] and [13].

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 $(\varphi_1, \ldots, \varphi_{N_h})$ 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

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

and

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

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 $\Sigma = \{a_1, \dots, a_N\}$ 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 $\alpha_1, \dots, \alpha_N$, there exists a unique element p of P such that $p(a_i) = \alpha_i, i = 1, \dots, N$. This means that the function*

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

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 $p_1, \dots, p_N$ of elements of P such that $p_i(a_j) = \delta_{ij}$. Given real $\alpha_1, \dots, \alpha_N$, the function $p = \sum_{i=1}^{N} \alpha_i p_i$ then verifies $p(a_j) = \alpha_j, j = 1 \dots, N$.*

Remark. *We call local basis functions of element K the N functions $p_1, \dots, p_N$ 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 $\mathbb{R}$: $\mathbb{P}_k = \text{Vect}\{1, X, \dots, X^k\}$ 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

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

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 :

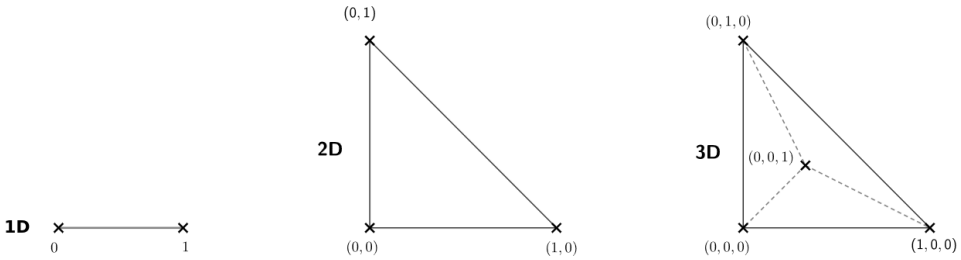

Figure 2.1: Example of reference Elements.

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 $\mathcal{T}_h = \{K_1, \ldots, K_{N_e}\}$ (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.

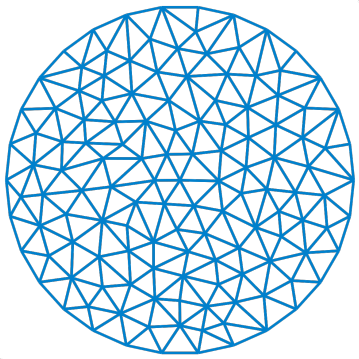

Figure 2.2: Example of a triangular mesh on a circles.

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 $K_1, \ldots, K_{N_e}$. Thus, for a cell $K \in \mathcal{T}_h$, we denote T_K the geometric transformation mapping $\hat{K}$ to K :

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

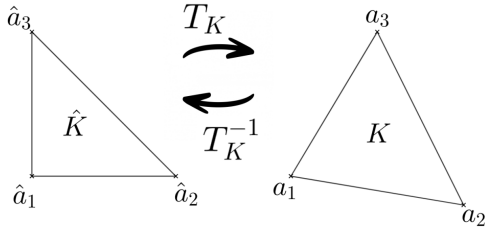

Figure 2.3: Geometric transformation applied to a triangle.

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

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

So for each $K \in \mathcal{T}_h$, we consider a tuple $\{a_{K,1}, \ldots, a_{K,n}\}$ (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

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

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 (Section 2.2.1. Next, we will give a general presentation of the method, starting with a description of the spaces required (Section 2.2.2.1), followed by a description of the ϕ-FEM method (Section 2.2.2.2). Finally, we will give some details on the ϕ-FEM method (section 2.2.3).

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 2.2.2.1.*

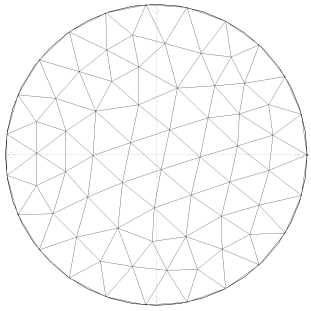

Figure 2.4: Standard FEM mesh example.

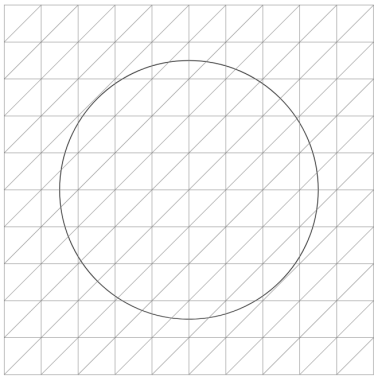

Figure 2.5: ϕ-FEM mesh example.

Remark. *For the purposes of this internship, the geometries considered are not organs (such as the liver), because these are complex geometries. We are 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. A paper in progress presents results with the combination of PhiFEM and an FNO on more complex geometries, notably ellipses.*

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 2.2.3.2. For more details on mesh assumptions, convergence results and finite element matrix condition number, please refer to [8]. ϕ-FEM schemes for the Poisson problem with Neumann or mixed (Dirichlet and Neumann) conditions are presented in [7, 5]. The ϕ-FEM scheme can also be found for other PDEs, including linear elasticity [5, Chapter 2], the heat equation [5, Chapter 5] and the Stokes problem [6].

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).

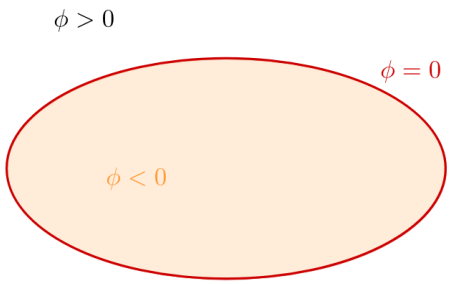

Figure 2.6: Definition of the level-set function.

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

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

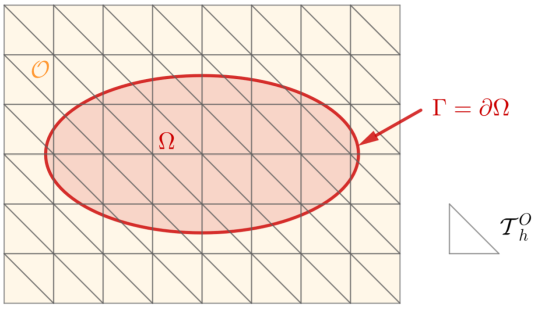

Figure 2.7: Fictitious domain.

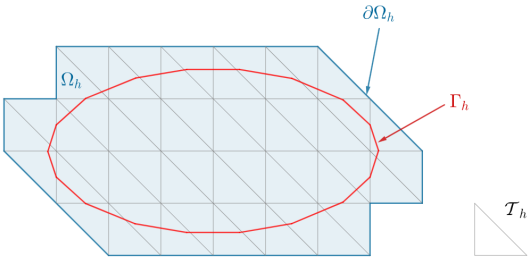

Figure 2.8: Domain considered.

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

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

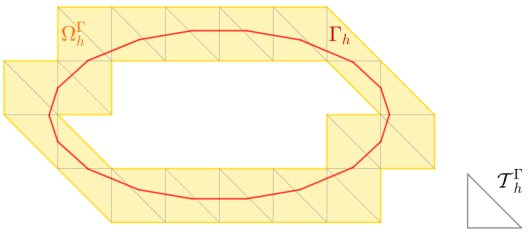

Figure 2.9: Boundary cells.

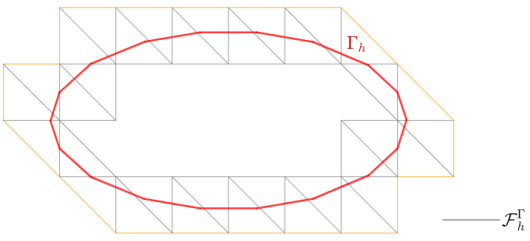

Figure 2.10: Boundary edges.

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 (since $\phi = 0$ 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 2.2.2.1, 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

In this section, we first give some information on stabilization terms (Section 2.2.3.1) and then present two methods for imposing non-homogeneous Dirichlet conditions, the direct method and the dual method (Section 2.2.3.2).

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 [3]. 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

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

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 [5] 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

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

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 [7].*

3 Fourier Neural Operator (FNO)

We will now introduce Fourier Neural Operators (FNO). For more information, please refer to the following article [10, 11, 9, 12].
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. The reference article [10] 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.

Remark. *The FNO used was implemented by Vincent Vigon*[7] *using Python's tensorflow library*[8]. *Furthermore, note that this report does not include a test of model parameter variation.*

3.1 Architecture of the FNO

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

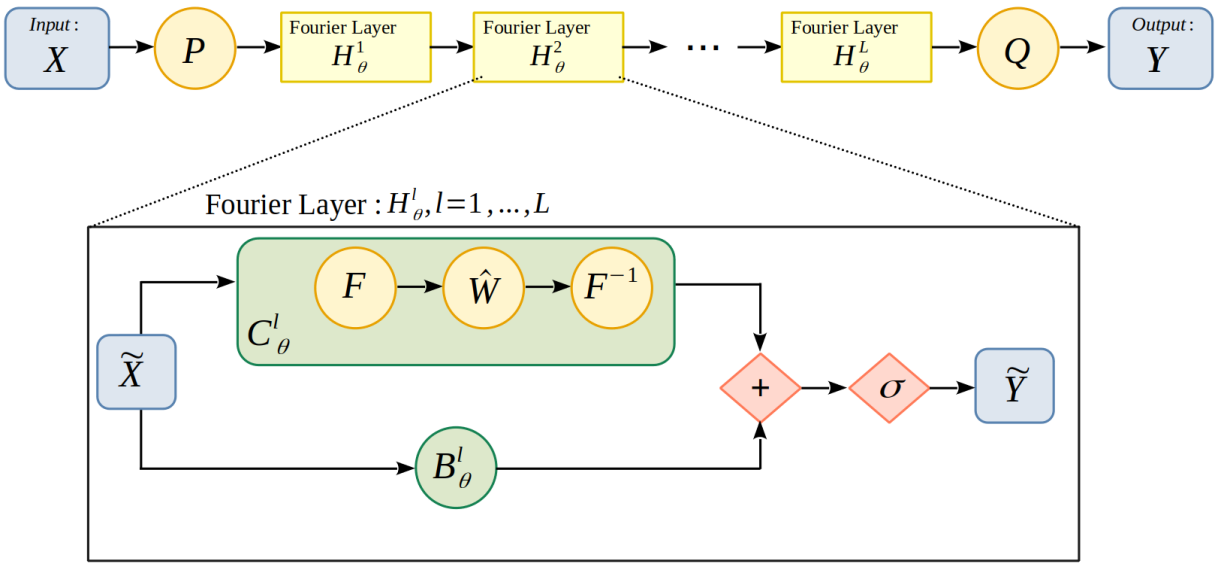

Figure 3.1: Architecture of the FNO.

The architecture of the FNO is as follows:

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

[7] Vincent Vigon: https://irma.math.unistra.fr/ vigon/
[8] Tensorflow: https://www.tensorflow.org/?hl=fr

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).

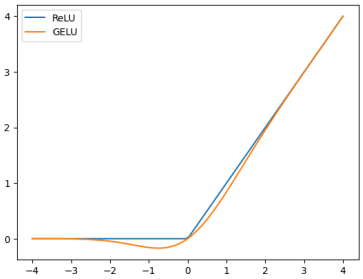

Figure 3.2: Activation functions used.

- $\mathscr{C}_\theta^l$ 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.

Remark. *These modulo operations are only natural for periodic images, which is not our case. The discontinuity that appears when we periodize the image causes oscillations on the edges of the filtered images. To limit this problem, we will apply a padding on the image which is the fact to extend the images by adding pixels all around, before performing the convolution. After the convolution, we restrict the image to partially erase the oscillations.*

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:

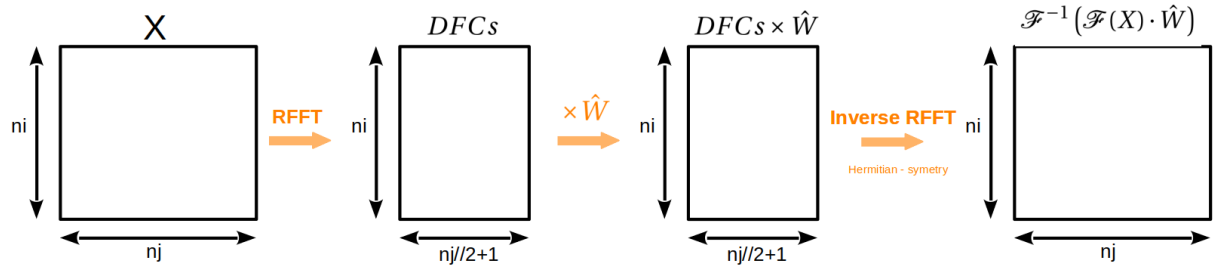

Figure 3.3: RFFT with Hermitian-symmetry scheme.

Remark. *In fact, we can check that $\mathscr{F}(X)$ satisfies Hermitian symmetry immediately.*

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 :

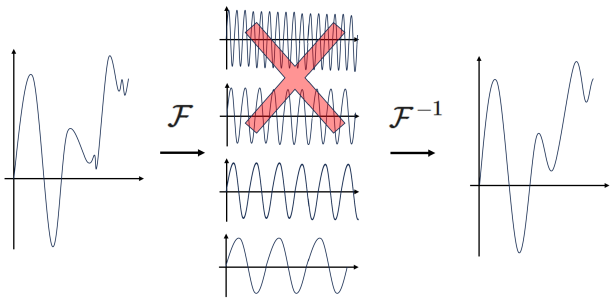

Figure 3.4: Low pass filter.

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

In our case, we want to use the FNO to predict the solutions of a PDE. As explained above, we'll train the FNO with a data set (sample of size n_{data}) generated by a ϕ-FEM solver. We will then inject the output of our FNO into a new solver that can correct the solution, i.e. improve its accuracy.

We are still trying to solve the Poisson problem with Dirichlet condition, defined by

$$\begin{cases} -\Delta(\phi w) = f, & \text{on } \Omega, \\ u = g, & \text{in } \Gamma, \end{cases}$$

with $u = \phi w$.
This problem can be approached in different ways. For example, we may want to consider a case where the level-set function ϕ changes, as in the case where we want to solve the problem of the geometry of an organ at different time steps. In this case, we'll need to generate a $\{\phi_i\}_{i=1,\dots,n_{data}}$ collection of level-sets sufficiently representative of the possible variations of the levelset. In a more simple case, if our geometry is defined by an ellipse in a precise domain, the $\{\phi_i\}_{i=1,\dots,n_{data}}$ family will group n_{data} ellipses whose parameters change, such as center or axes.
We may also want to solve the problem for a collection of source terms $\{f_i\}_{i=1,\dots,n_{data}}$. For example, this set could be a Gaussian set whose esperance and variance are varied. In the same idea, we might wish to vary the Dirchlet condition and thus create a collection $\{g_i\}_{i=1,\dots,n_{data}}$.

Remark. *Note that we're working in a discrete way here, so for each i, the terms f_i, g_i and ϕ_i are in fact 2D matrices of size (ni, nj).*

Remark. *Note also that the FNO has less difficulty learning solutions that don't have a wide range of values. This is why the collection of $\{f_i\}$ that we'll be using in the following will in fact be the normalization of the previous collection :*

$$f_{i,norm} = \frac{f_i}{max_{i=1,\dots,n_{data}} ||f_i||_{L^2(\mathcal{O})}}.$$

Here are the steps that will be performed to train the FNO (Figure 3.5):

- We start by creating a dataset containing the level-set, source term and boundary condition collections, defined by

  $$X\_train = \{f_i, g_i, \phi_i\}_{i=1,\dots,n_{data}}.$$

  Remark. *Note that we can also consider the problem as homogeneous, in which case X_train will only be generated from f and ϕ. We may also wish to fix the geometry, in which case the training sample X_train will not contain the ϕ term.*

  We can then solve these n_{data} problems using the ϕ-FEM method, where the solution to each of them is defined by $u = \phi w$. We then define the training sample

  $$Y\_train = \{w_i\}_{i=1,\dots,n_{data}}$$

  where $u_i = \phi_i w_i$ is the solution associated to the i-th problem of the X_train sample, i.e. solution of

  $$\begin{cases} -\Delta(\phi_i w_i) = f_i, & \text{on } \Omega, \\ u_i = g_i, & \text{in } \Gamma. \end{cases}$$

Remark. *Note that in practice, we have enriched the data by increasing the number of channels in the X_train sample. In fact, for each problem i, we add to the sample the primitives of f_i according to x and y, as well as the second primitives according to xx and yy. The X_train sample is then of size (n_{data}, ni, nj, nk) with $nk = 7$ the number of channels if we consider the 3 collections. The Y_train sample is of size $(n_{data}, ni, nj, 1)$.*

- At this moment, we have the (X_train,Y_train) pair that will enable us to train our FNO. More precisely, we're looking to minimize a loss function on the model's θ parameters by using gradient descent. We'll choose the following loss function:

$$loss_\theta = loss_\theta^{(0)} + loss_\theta^{(1)} + loss_\theta^{(2)}$$

with

$$loss_\theta^{(0)} = \frac{1}{n_{data}} \sum_{i=1}^{n_{data}} mse(\phi_i w_i - \phi_i w_{\theta,i})$$

$$loss_\theta^{(1)} = \frac{1}{n_{data}} \sum_{i=1}^{n_{data}} mse(\nabla_x(\phi_i w_i) - \nabla_x(\phi_i w_{\theta,i})) + mse(\nabla_y(\phi_i w_i) - \nabla_y(\phi_i w_{\theta,i}))$$

$$loss_\theta^{(2)} = \frac{1}{n_{data}} \sum_{i=1}^{n_{data}} mse(\nabla_{xx}(\phi_i w_i) - \nabla_{xx}(\phi_i w_{\theta,i})) + mse(\nabla_{yy}(\phi_i w_i) - \nabla_{yy}(\phi_i w_{\theta,i}))$$

with w_i the ϕ-FEM solution associated to the i-th problem considered (i.e. the i-th Y_train sample data), $w_{\theta,i}$ the FNO prediction associated to the i-th problem, ϕ_i the i-th levelset and mse the "Mean Square Error" function defined by

$$mse(A) = \frac{1}{ni}\frac{1}{nj} \sum_{i=0}^{ni-1} \sum_{j=0}^{nj-1} A_{i,j}^2.$$

Remark. *Note that first and second derivatives according to x and y are calculated here by finite differences.*

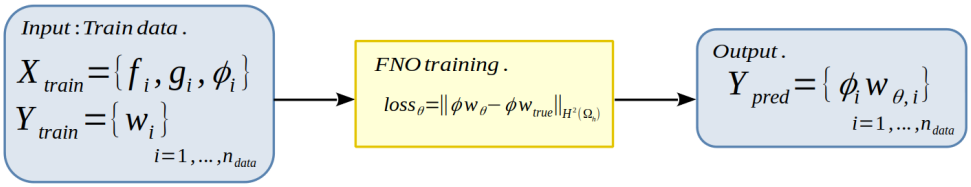

Figure 3.5: Representation of the FNO training.

We can now proceed to the correction stage (Figure 3.6), where we consider a type of correction defined in Section 4.2 (correction by adding, correction by multiplying or correction

by multiplying on an enhanced problem). We can then consider a new test sample X_test constructed in the same way as the training sample and defined by

$$X\_test = \{f_i^{test}, g_i^{test}, \phi_i^{test}\}_{i=1,\dots,n_{test}}.$$

We will then provide this sample as input to the FNO in order to obtain its prediction $w_{\theta,i}^{test}$ for each problem i of the test sample (where θ corresponds to the parameters learned during training). We then construct $u_{\theta_i}^{test} = \phi_i w_{\theta,i}^{test}$ which will be given as input to one of the correction solvers. We will then obtain what we call the corrected solution.

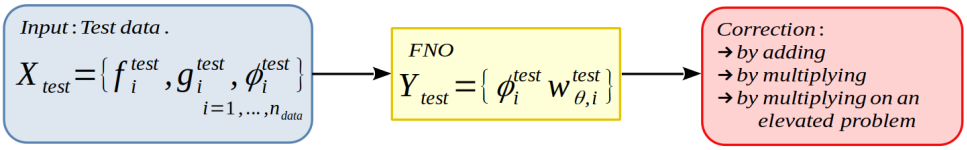

Figure 3.6: Representation of correction steps

4 Correction

Intro !

4.1 Presentation of different problems considered

A faire au fur et à mesure !
In this section, we present the different problems we'll be considering in Section 4.3. We will consider the geometry of a circle in Section 4.1.1 represented in Figure 4.1 as well as the geometry of a square in Section 4.1.2 represented in Figure 4.2. We'll also look at different solutions à compléter (sol analytique-exacte vs sol de référence calculés avec FEM sur-raffinée).

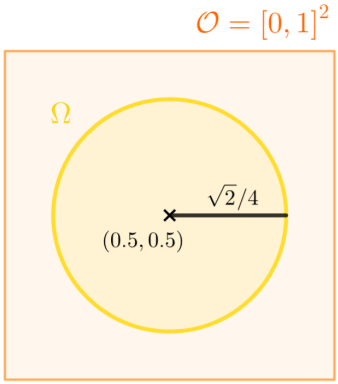

Figure 4.1: Representation of the first domain considered : the Circle.

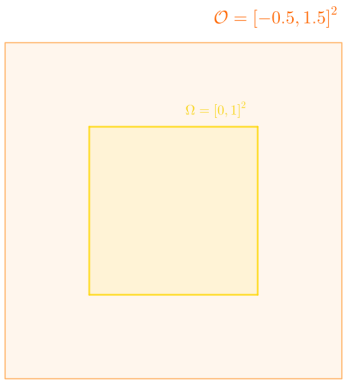

Figure 4.2: Representation of the second domain considered : the Square.

Remark. *Note that in the case where the problem is non-homogeneous, we can consider*

$$g(x,y) = u_{ex}(x,y) \times (1 + \phi(x,y))$$

with ϕ the levelset function, null by definition on Γ.

4.1.1 First domain : the Circle.

Here, we'll consider the Ω domain to be the circle of radius $\sqrt{2}/4$ and center $(0.5, 0.5)$.This domain is entirely included in the fictitious domain $O = [0,1]^2$ (Figure 4.1).
We will consider the levelset function ϕ defined by

$$\phi(x,y) = -1/8 + (x - 1/2)^2 + (y - 1/2)^2,$$

negative function inside the domain, null at the boundary and positive outside it
We then consider different problems à compléter.

4.1.1.1 First problem

Here, we are interested in the Poisson problem with an analytical solution

$$u_{ex}(x,y) = S \times \sin\left(8\pi f\left((x-0.5)^2 + (y-0.5)^2\right) + \varphi\right)$$

where $S \in [0,1]$ is the amplitude of the signal, $f \in \mathbb{N}$ can be associated with the "frequency" of the signal and $\varphi \in [0,1]$ the phase at the origin
Thus, the second associated member is defined by

$$\begin{aligned} f(x,y) = 256\pi^2 S f^2 \left((x-0.5)^2 + (y-0.5)^2\right) \sin\left[8\pi f\left((x-0.5)^2 + (y-0.5)^2\right) + \varphi\right] \\ - 32\pi S f \cos\left[8\pi f\left((x-0.5)^2 + (y-0.5)^2\right) + \varphi\right] \end{aligned}$$

Remark. *Note that for $\varphi = 0$, the Dirichlet conditions considered are then homogeneous on the circle (i.e. $g = 0$ on Γ).*

ajouter image solution et f
ajouter les autres cas tests : f gaussienne...

4.1.2 Second domain : the Square.

Here, we'll consider the Ω domain to be the unit square $\Omega = [0,1]^2$. This domain is entirely included in the fictitious domain $O = [-0.5, 1.5]^2$ (Figure 4.2).
In this case, we'll consider two levelset functions. The first function ϕ, defined by

$$\phi(x,y) = x(1-x)y(1-y),$$

will be used when solving the weak problem. However, it cannot be used to construct our cell and face sets, as it is only negative within the Ω domain (Figure 4.3).

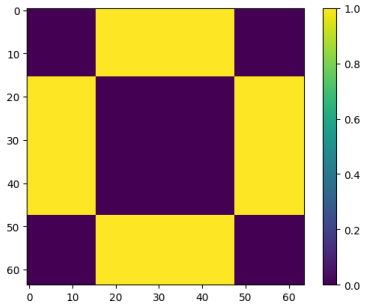

Figure 4.3: Representation of $\phi(x,y) < 0$.

To construct the sets of cells and faces needed to perform the ϕ-FEM method, we will consider a second function, denoted ϕ_C and defined by

$$\phi_C(x,y) = \max(|x-0.5|, |y-0.5|) - 0.5,$$

which is indeed negative inside the domain, zero at the boundary and positive outside, but which does not sufficiently satisfy the regularity conditions necessary for ϕ-FEM.
We the considered different problems à compléter.

4.1.2.1 First problem

Here, we are interested in the Poisson problem with as analytical solution

$$u_{ex}(x,y) = S \times \sin\left(2\pi f x + \varphi\right) \times \sin\left(2\pi f y + \varphi\right)$$

where $S \in [0,1]$ is the amplitude of the signal, $f \in \mathbb{N}$ can be associated with the "frequency" of the signal and $\varphi \in [0,1]$ the phase at the origin.
Thus, the associated second member is defined by

$$f(x,y) = 8\pi^2 S f^2 \sin\left(2\pi f x + \varphi\right) \sin\left(2\pi f y + \varphi\right)$$

Remark. *It should be noted that as for the circle for $\varphi = 0$, the Dirichlet conditions considered are then homogeneous on the square (i.e. $g = 0$ on Γ).*

4.2 Presentation of the different correction methods considered

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).

Remark. *In what follows, we assume that $\tilde{\phi}$ already has the right conditions at the boundary, i.e. $\tilde{\phi} = g$ on Γ.*

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

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

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

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

with $\tilde{f} = f + \Delta\tilde{\phi}$.
Thus for the standard FEM method, the weak formulation will be given by

$$\int_\Omega \nabla \tilde{C} \cdot \nabla v = \int_\Omega \tilde{f} v$$

where the homogeneous Dirichlet conditions can be strongly imposed by classical methods (penalization, elimination...).
For the ϕ-FEM method, we look for C such that $\tilde{C} = \phi C$ and the weak formulation (associated with a homogeneous problem because $\tilde{C} = 0$ on Γ) is given by

$$\int_{\Omega_h} \nabla(\phi C) \cdot \nabla(\phi v) - \int_{\partial\Omega_h} \frac{\partial}{\partial n}(\phi C)\phi v + G_h(w, v) = \int_{\Omega_h} \tilde{f}\phi v + G_h^{rhs}(v)$$

with

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

and

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

Remark. *In practice, it may be useful to integrate by parts (IPP) the term containing $\Delta\tilde{\phi}$ (implicitly included in $\tilde{f}$). expliciter !*

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.*

So for the standard FEM method, the weak formulation will be given by

$$\int_\Omega \nabla(\tilde{\phi}C)\cdot\nabla(\tilde{\phi}v) = \int_\Omega f\tilde{\phi}v$$

where homogeneous or non-homogeneous Dirichlet conditions can be strongly imposed by classical methods (penalization, elimination...)
For the ϕ-FEM method, the weak formulation for the homogeneous problem is given by

$$\int_{\Omega_h} \nabla(\tilde{\phi}C)\cdot\nabla(\tilde{\phi}v) - \int_{\partial\Omega_h} \frac{\partial}{\partial n}(\tilde{\phi}C)\tilde{\phi}v + G_h(w,v) = \int_{\Omega_h} f\tilde{\phi}v + G_h^{rhs}(v)$$

with

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

and

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

In the non-homogeneous case, it is important to impose the boundary conditions either by the direct method or by the dual method presented in section add section. PAS SURE !

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

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

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

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

So for the standard FEM method, the weak formulation will be given by

$$\int_\Omega \nabla(\hat{\phi}C)\cdot\nabla(\hat{\phi}v) = \int_\Omega f\hat{\phi}v$$

where homogeneous or non-homogeneous Dirichlet conditions can be strongly imposed by classical methods (penalization, elimination...)
For the ϕ-FEM method, the weak formulation for the homogeneous problem is given by

$$\int_{\Omega_h} \nabla(\hat{\phi}C)\cdot\nabla(\hat{\phi}v) - \int_{\partial\Omega_h} \frac{\partial}{\partial n}(\hat{\phi}C)\hat{\phi}v + G_h(w,v) = \int_{\Omega_h} f\hat{\phi}v + G_h^{rhs}(v)$$

with

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

and

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

In the non-homogeneous case, it is important to impose the edge conditions either by the direct method or by the dual method presented in section add section. PAS SURE !

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 $\Longleftrightarrow$ 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 4.3.4 and then with a multi-perceptron network and PINNs in Section 4.3.5. The reasons for considering other neural networks will be explained in detail in these sections.

4.3.1 Correction on exact solution

First, we will look at the correction of an exact solution. In other words, we consider

$$\tilde{\phi} = u_{ex}$$

This first step enables us to check that, by supplying an already exact solution to the various correctors under consideration, we again obtain an exact output solution (to the nearest machine error).
In all the following cases, we'll consider $S = 0.5$, as well as $\varphi = 0$ in the case of the homogeneous problem and $\varphi = 1$ in the non-homogeneous case. We will vary f between 1 (low solution variability) and 4 (high solution variability) and we will choose to take 100 vertices in each direction *nb_vert*=100.

Remark. *We will consider here only the case of correction by addition (with and without IPP on $\Delta\tilde{\phi}$) as well as the case of correction by multiplication.*

- **Results on the Circle :**

  We consider here the Circle problem with the solution defined in Section 4.1.1.1.

  Homogeneous case :

  We consider here the homogeneous problem (i.e. with $\varphi = 0$) and seek to test the various correction methods with standard FEM and ϕ-FEM methods.

	FEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	2.10e-03	2.44e-10	1.29e-13	2.97e-13
f = 2	6.62e-03	1.53e-10	1.28e-13	2.80e-13
f = 3	1.41e-02	8.86e-11	1.27e-13	2.68e-13
f = 4	2.42e-02	9.52e-11	1.26e-13	2.61e-13

Figure 4.4: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Circle with standard FEM.

	PhiFEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	8.05e-04	8.12e-11	3.32e-13	4.29e-13
f = 2	6.31e-03	6.50e-11	9.60e-12	1.12e-11
f = 3	2.04e-02	1.53e-10	1.45e-10	7.04e-11
f = 4	4.57e-02	8.78e-10	8.23e-10	1.04e-09

Figure 4.5: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Circle with ϕ-FEM.

  Non-homogeneous case :

  We consider here the non-homogeneous problem (i.e. with $\varphi = 1$) and seek to test the various correction methods with standard FEM and ϕ-FEM methods.

	FEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	2.52e-03	2.48e-10	2.06e-14	2.10e-14
f = 2	1.05e-02	1.00e-10	1.66e-14	2.50e-14
f = 3	2.33e-02	7.67e-11	1.70e-14	2.60e-14
f = 4	4.07e-02	4.02e-11	1.56e-14	3.28e-14

Figure 4.6: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Circle with standard FEM.

	PhiFEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	9.09e-05	2.97e-10	4.47e-13	1.68e-10
f = 2	3.97e-04	1.17e-10	4.55e-12	5.05e-11
f = 3	9.26e-04	8.05e-11	4.44e-11	2.89e-10
f = 4	1.66e-03	7.61e-10	7.65e-10	1.52e-09

Figure 4.7: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Circle with ϕ-FEM.

- **Results on the Square :**

  We consider here the Square problem with the solution defined in Section 4.1.2.1.

  Homogeneous case :

  We consider here the homogeneous problem (i.e. with $\varphi = 0$) and seek to test the various correction methods with standard FEM and ϕ-FEM methods.

	FEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	2.61e-03	5.16e-11	1.28e-13	2.48e-13
f = 2	1.04e-02	1.43e-11	1.27e-13	2.49e-13
f = 3	2.34e-02	8.07e-12	1.26e-13	2.49e-13
f = 4	4.12e-02	1.28e-11	1.24e-13	2.49e-13

Figure 4.8: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Square with standard FEM.

	PhiFEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	1.84e-03	2.21e-11	2.21e-13	3.08e-13
f = 2	1.44e-02	8.80e-12	4.77e-12	4.80e-12
f = 3	3.69e-02	1.30e-10	1.29e-10	1.73e-10
f = 4	6.84e-02	1.27e-09	1.27e-09	2.02e-09

Figure 4.9: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Square with ϕ-FEM.

  Non-homogeneous case :

  We consider here the non-homogeneous problem (i.e. with $\varphi = 1$) and seek to test the various correction methods with standard FEM and ϕ-FEM methods.

	FEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	2.30e-03	4.74e-11	1.10e-13	1.72e-13
f = 2	9.53e-03	1.79e-11	1.15e-13	2.16e-13
f = 3	2.19e-02	1.19e-11	1.17e-13	2.29e-13
f = 4	3.88e-02	8.32e-12	1.18e-13	2.35e-13

Figure 4.10: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Square with standard FEM.

	PhiFEM	Corr_add	Corr_add_IPP	Corr_mult
f = 1	1.20e-04	1.42e-11	2.96e-13	2.02e-11
f = 2	5.74e-04	9.83e-12	4.45e-12	2.90e-11
f = 3	1.34e-03	1.16e-10	1.15e-10	1.78e-10
f = 4	2.39e-03	1.13e-09	1.14e-09	1.46e-09

Figure 4.11: Errors $||\cdot||_{L^2,rel}$ obtained with different methods on the Square with ϕ-FEM.

It would therefore seem that the various correction methods work in the different cases considered.

4.3.2 Correction on disturbed solution

Now, let's consider a deliberately disturbed solution. The purpose of this step is to check that the correction solvers also work with a solution that is very close to the real solution, but not exact. In this section, we will consider a manually disturbed solution, i.e. the exact solution to which we've added a small, analytically known perturbation.
As explained above, we begin by considering $\tilde{\phi}$ as a manually perturbed solution defined by

$$\tilde{\phi}(x,y) = u_{ex}(x,y) + \epsilon P(x,y)$$

where u_{ex} defines the exact solution to the problem, P the perturbation applied to it and ϵ is a real number that allows the amplitude of the perturbation to be easily increased or decreased.

Remark. *Notice that by taking $\epsilon = 0$, we return to the case of correction on an exact solution presented in Section 4.3.1. Recall the relative errors obtained by standard FEM and ϕ-FEM on the circle and on the square for frequencies $f \in \{1,2,3,4\}$ for the homogeneous (Figure 4.12) and non-homogeneous problem (Figure 4.13).*

		f = 1	f = 2	f = 3	f = 4
circle	fem	2.10e-03	6.62e-03	1.41e-02	2.42e-02
	phifem	8.05e-04	6.31e-03	2.04e-02	4.57e-02
square	fem	2.61e-03	1.04e-02	2.34e-02	4.12e-02
	phifem	1.84e-03	1.44e-02	3.69e-02	6.84e-02

Figure 4.12: Table summarizing the errors obtained by standard FEM and ϕ-FEM on the circle and the square (homogeneous case).

		f = 1	f = 2	f = 3	f = 4
circle	fem	2.52e-03	1.05e-02	2.33e-02	4.07e-02
	phifem	9.09e-05	3.97e-04	9.26e-04	1.66e-03
square	fem	2.30e-03	9.53e-03	2.19e-02	3.88e-02
	phifem	1.20e-04	5.74e-04	1.34e-03	2.39e-03

Figure 4.13: Table summarizing the errors obtained by standard FEM and ϕ-FEM on the circle and the square (non-homogeneous case).

In our case, we will choose to consider P as being of the same form as our exact solution (defined with different parameters), but we could very well consider a completely different perturbation.

Remark. *Note that the shape of the perturbation has a huge influence on the accuracy of the solvers, and that the difficulty lies in the following cases where its expression is not explicitly known (as in the case of ϕ-FEM in Section 4.3.3 or FNO in Section 4.3.4).*

In the case of Circle geometry where we consider the problem 4.1.1.1, the perturbation will be defined by

$$P(x,y) = S_p \times \sin\left(8\pi f_p\left((x-0.5)^2 + (y-0.5)^2\right) + \varphi_p\right)$$

where $S_p \in [0,1]$ is the amplitude of the signal, $f_p \in \mathbb{N}$ can be associated with the "frequency" of the signal and $\varphi_p \in [0,1]$ the phase at the origin.
In the case of Square geometry where we consider the problem 4.1.2.1, the perturbation will be defined by

$$P(x,y) = S_p \times \sin\left(2\pi f_p x + \varphi_p\right) \times \sin\left(2\pi f_p y + \varphi_p\right)$$

where $S_p \in [0,1]$ is the amplitude of the signal, $f_p \in \mathbb{N}$ can be associated with the "frequency" of the signal and $\varphi_p \in [0,1]$ the phase at the origin.

Remark. *Note that for the boundary conditions of the solution to be satisfied, i.e. for* $\tilde{\phi} = u_{ex}$ *on* Γ*, it is essential that* $P = 0$ *on* Γ*. In the case of both circle and square, we will then take* $\varphi_p = 0$*.*

4.3.2.1 Results with differents ϵ

Results on the homogeneous case :
First, we consider the homogeneous case (i.e. with $\varphi = 0$). The aim is to test correction by addition (without IPP) and correction by multiplication by varying the amplitude of the perturbation (in other words, by varying ϵ). We'll consider the case of the circle and the square for standard FEM and ϕ-FEM methods, and we'll try to separate the cases according to the frequencies considered. In other words, for $f, f_p \in \{1,2,3,4\}$, we're interested in the following three cases. The first is the case where the solution frequency is greater than the perturbation frequency ($f > f_p$), i.e. a highly variable solution and a less variable perturbation. The second is where the solution and perturbation frequencies are equal ($f = f_p$), i.e. the solution and perturbation have the same variability. The last category covers cases where the perturbation is "nastier" than the solution, i.e. it has a higher frequency than the solution ($f < f_p$).

- **Results on the Circle :**

  We consider here the Circle problem with the solution defined in Section 4.1.1.1. Here, we consider correction by adding (without IPP) with standard FEM (Figure 4.14) and with ϕ-FEM (Figure 4.15).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	1.28e-11
	f = 3 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	8.33e-12
	f = 3 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	8.33e-12
	f = 4 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	1.08e-11
	f = 4 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	1.08e-11
	f = 4 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	1.08e-11
f = fp	f = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	4.16e-11
	f = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	1.28e-11
	f = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	8.33e-12
	f = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	1.08e-11
f < fp	f = 1 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	4.16e-11
	f = 1 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	4.16e-11
	f = 1 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	4.16e-11
	f = 2 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	1.28e-11
	f = 2 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	1.28e-11
	f = 3 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	8.33e-12

Figure 4.14: Correction by adding on the Circle with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	8.14e-04	8.14e-05	8.14e-06	8.14e-07	8.15e-08	6.50e-11
	f = 3 ; fp = 1	8.13e-04	8.13e-05	8.13e-06	8.13e-07	8.13e-08	1.53e-10
	f = 3 ; fp = 2	6.52e-03	6.52e-04	6.52e-05	6.52e-06	6.52e-07	1.53e-10
	f = 4 ; fp = 1	8.13e-04	8.13e-05	8.13e-06	8.13e-07	8.17e-08	8.78e-10
	f = 4 ; fp = 2	6.51e-03	6.51e-04	6.51e-05	6.51e-06	6.50e-07	8.78e-10
	f = 4 ; fp = 3	2.15e-02	2.15e-03	2.15e-04	2.15e-05	2.15e-06	8.78e-10
f = fp	f = 1	8.14e-04	8.14e-05	8.14e-06	8.15e-07	8.15e-08	8.12e-11
	f = 2	6.52e-03	6.52e-04	6.52e-05	6.52e-06	6.52e-07	6.50e-11
	f = 3	2.15e-02	2.15e-03	2.15e-04	2.15e-05	2.15e-06	1.53e-10
	f = 4	4.89e-02	4.89e-03	4.89e-04	4.89e-05	4.89e-06	8.78e-10
f < fp	f = 1 ; fp = 2	6.52e-03	6.52e-04	6.52e-05	6.52e-06	6.52e-07	8.12e-11
	f = 1 ; fp = 3	2.16e-02	2.16e-03	2.16e-04	2.16e-05	2.16e-06	8.12e-11
	f = 1 ; fp = 4	4.90e-02	4.90e-03	4.90e-04	4.90e-05	4.90e-06	8.12e-11
	f = 2 ; fp = 3	2.15e-02	2.15e-03	2.15e-04	2.15e-05	2.15e-06	6.50e-11
	f = 2 ; fp = 4	4.90e-02	4.90e-03	4.90e-04	4.90e-05	4.90e-06	6.50e-11
	f = 3 ; fp = 4	4.90e-02	4.90e-03	4.90e-04	4.90e-05	4.90e-06	1.53e-10

Figure 4.15: Correction by adding on the Circle with ϕ-FEM.

Then, we consider correction by multiplying with standard FEM (Figure 4.16) and with ϕ-FEM (Figure 4.17).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	6.45e-01	8.61e-02	8.68e-03	8.68e-04	8.68e-05	2.66e-14
	f = 3 ; fp = 1	7.87e-01	9.35e-02	9.39e-03	9.40e-04	9.40e-05	2.67e-14
	f = 3 ; fp = 2	5.74e-01	7.98e-02	8.04e-03	8.04e-04	8.04e-05	2.67e-14
	f = 4 ; fp = 1	8.48e-01	9.66e-02	9.67e-03	9.67e-04	9.67e-05	3.71e-14
	f = 4 ; fp = 2	6.61e-01	8.75e-02	8.82e-03	8.82e-04	8.82e-05	3.71e-14
	f = 4 ; fp = 3	5.60e-01	7.75e-02	7.81e-03	7.81e-04	7.81e-05	3.71e-14
f = fp	f = 1	2.78e-02	2.78e-03	2.78e-04	2.78e-05	2.78e-06	2.56e-14
	f = 2	3.72e-02	3.72e-03	3.72e-04	3.72e-05	3.72e-06	2.66e-14
	f = 3	4.66e-02	4.66e-03	4.66e-04	4.66e-05	4.66e-06	2.67e-14
	f = 4	5.42e-02	5.42e-03	5.42e-04	5.42e-05	5.42e-06	3.71e-14
f < fp	f = 1 ; fp = 2	6.37e-01	4.01e-03	3.98e-04	3.98e-05	3.98e-06	2.56e-14
	f = 1 ; fp = 3	7.82e-01	5.70e-03	5.53e-04	5.55e-05	5.55e-06	2.56e-14
	f = 1 ; fp = 4	8.44e-01	2.89e-02	7.42e-04	7.39e-05	7.39e-06	2.56e-14
	f = 2 ; fp = 3	5.68e-01	6.57e-02	6.77e-03	6.78e-04	6.78e-05	2.66e-14
	f = 2 ; fp = 4	6.54e-01	6.58e-03	6.38e-04	6.38e-05	6.38e-06	2.66e-14
	f = 3 ; fp = 4	5.56e-01	6.82e-02	6.94e-03	6.94e-04	6.94e-05	2.67e-14

Figure 4.16: Correction by multiplying on the Circle with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	9.73e-01	1.04e-01	1.03e-02	1.03e-03	1.03e-04	1.12e-11
	f = 3 ; fp = 1	1.02e+00	1.03e-01	1.02e-02	1.02e-03	1.02e-04	7.04e-11
	f = 3 ; fp = 2	7.61e-01	8.59e-02	8.60e-03	8.60e-04	8.60e-05	7.04e-11
	f = 4 ; fp = 1	9.96e-01	1.01e-01	1.01e-02	1.01e-03	1.01e-04	1.04e-09
	f = 4 ; fp = 2	8.58e-01	9.48e-02	9.45e-03	9.45e-04	9.45e-05	1.04e-09
	f = 4 ; fp = 3	6.96e-01	7.69e-02	7.72e-03	7.72e-04	7.72e-05	1.04e-09
f = fp	f = 1	4.29e-13	4.24e-13	4.26e-13	4.29e-13	4.27e-13	4.29e-13
	f = 2	1.12e-11	1.12e-11	1.12e-11	1.12e-11	1.12e-11	1.12e-11
	f = 3	7.04e-11	7.04e-11	7.04e-11	7.04e-11	7.04e-11	7.04e-11
	f = 4	1.04e-09	1.04e-09	1.04e-09	1.04e-09	1.04e-09	1.04e-09
f < fp	f = 1 ; fp = 2	9.74e-01	4.66e-04	7.55e-05	7.76e-06	7.78e-07	4.29e-13
	f = 1 ; fp = 3	1.02e+00	6.28e-03	3.56e-04	3.14e-05	3.10e-06	4.29e-13
	f = 1 ; fp = 4	9.98e-01	5.23e-02	5.75e-04	8.60e-05	8.88e-06	4.29e-13
	f = 2 ; fp = 3	7.61e-01	1.02e-01	1.04e-02	1.04e-03	1.04e-04	1.12e-11
	f = 2 ; fp = 4	8.59e-01	1.07e-02	9.31e-04	9.07e-05	9.04e-06	1.12e-11
	f = 3 ; fp = 4	6.96e-01	8.63e-02	8.72e-03	8.73e-04	8.73e-05	7.04e-11

Figure 4.17: Correction by multiplying on the Circle with ϕ-FEM.

- **Results on the Square :**

  We consider here the Square problem with the solution defined in Section 4.1.2.1. Here, we consider correction by adding (without IPP) with standard FEM (Figure 4.18) and with ϕ-FEM (Figure 4.19).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	1.28e-11
	f = 3 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	8.33e-12
	f = 3 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	8.33e-12
	f = 4 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	1.08e-11
	f = 4 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	1.08e-11
	f = 4 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	1.08e-11
f = fp	f = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	4.16e-11
	f = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	1.28e-11
	f = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	8.33e-12
	f = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	1.08e-11
f < fp	f = 1 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	4.16e-11
	f = 1 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	4.16e-11
	f = 1 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	4.16e-11
	f = 2 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	1.28e-11
	f = 2 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	1.28e-11
	f = 3 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	8.33e-12

Figure 4.18: Correction by adding on the Square with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	1.86e-03	1.86e-04	1.86e-05	1.86e-06	1.86e-07	1.33e-11
	f = 3 ; fp = 1	1.86e-03	1.86e-04	1.86e-05	1.86e-06	1.86e-07	1.24e-10
	f = 3 ; fp = 2	1.50e-02	1.50e-03	1.50e-04	1.50e-05	1.50e-06	1.24e-10
	f = 4 ; fp = 1	1.86e-03	1.86e-04	1.86e-05	1.86e-06	1.86e-07	1.27e-09
	f = 4 ; fp = 2	1.50e-02	1.50e-03	1.50e-04	1.50e-05	1.50e-06	1.27e-09
	f = 4 ; fp = 3	4.09e-02	4.09e-03	4.09e-04	4.09e-05	4.09e-06	1.27e-09
f = fp	f = 1	1.86e-03	1.86e-04	1.86e-05	1.86e-06	1.86e-07	5.43e-11
	f = 2	1.51e-02	1.51e-03	1.51e-04	1.51e-05	1.51e-06	1.33e-11
	f = 3	4.10e-02	4.10e-03	4.10e-04	4.10e-05	4.10e-06	1.24e-10
	f = 4	8.08e-02	8.08e-03	8.08e-04	8.08e-05	8.08e-06	1.27e-09
f < fp	f = 1 ; fp = 2	1.51e-02	1.51e-03	1.51e-04	1.51e-05	1.51e-06	5.43e-11
	f = 1 ; fp = 3	4.10e-02	4.10e-03	4.10e-04	4.10e-05	4.10e-06	5.43e-11
	f = 1 ; fp = 4	8.11e-02	8.11e-03	8.11e-04	8.11e-05	8.11e-06	5.43e-11
	f = 2 ; fp = 3	4.10e-02	4.10e-03	4.10e-04	4.10e-05	4.10e-06	1.33e-11
	f = 2 ; fp = 4	8.10e-02	8.10e-03	8.10e-04	8.10e-05	8.10e-06	1.33e-11
	f = 3 ; fp = 4	8.09e-02	8.09e-03	8.09e-04	8.09e-05	8.09e-06	1.24e-10

Figure 4.19: Correction by adding on the Square with ϕ-FEM.

Then, we consider correction by multiplying with standard FEM (Figure 4.20) and with ϕ-FEM (Figure 4.21).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	6.45e-01	8.61e-02	8.68e-03	8.68e-04	8.68e-05	2.66e-14
	f = 3 ; fp = 1	7.87e-01	9.35e-02	9.39e-03	9.40e-04	9.40e-05	2.67e-14
	f = 3 ; fp = 2	5.74e-01	7.98e-02	8.04e-03	8.04e-04	8.04e-05	2.67e-14
	f = 4 ; fp = 1	8.48e-01	9.66e-02	9.67e-03	9.67e-04	9.67e-05	3.71e-14
	f = 4 ; fp = 2	6.61e-01	8.75e-02	8.82e-03	8.82e-04	8.82e-05	3.71e-14
	f = 4 ; fp = 3	5.60e-01	7.75e-02	7.81e-03	7.81e-04	7.81e-05	3.71e-14
f = fp	f = 1	2.78e-02	2.78e-03	2.78e-04	2.78e-05	2.78e-06	2.56e-14
	f = 2	3.72e-02	3.72e-03	3.72e-04	3.72e-05	3.72e-06	2.66e-14
	f = 3	4.66e-02	4.66e-03	4.66e-04	4.66e-05	4.66e-06	2.67e-14
	f = 4	5.42e-02	5.42e-03	5.42e-04	5.42e-05	5.42e-06	3.71e-14
f < fp	f = 1 ; fp = 2	6.37e-01	4.01e-03	3.98e-04	3.98e-05	3.98e-06	2.56e-14
	f = 1 ; fp = 3	7.82e-01	5.70e-03	5.53e-04	5.55e-05	5.55e-06	2.56e-14
	f = 1 ; fp = 4	8.44e-01	2.89e-02	7.42e-04	7.39e-05	7.39e-06	2.56e-14
	f = 2 ; fp = 3	5.68e-01	6.57e-02	6.77e-03	6.78e-04	6.78e-05	2.66e-14
	f = 2 ; fp = 4	6.54e-01	6.58e-03	6.38e-04	6.38e-05	6.38e-06	2.66e-14
	f = 3 ; fp = 4	5.56e-01	6.82e-02	6.94e-03	6.94e-04	6.94e-05	2.67e-14

Figure 4.20: Correction by multiplying on the Square with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	6.45e-01	8.60e-02	8.71e-03	8.72e-04	8.72e-05	4.38e-12
	f = 3 ; fp = 1	7.89e-01	9.34e-02	9.41e-03	9.41e-04	9.41e-05	1.72e-10
	f = 3 ; fp = 2	5.79e-01	8.01e-02	8.11e-03	8.11e-04	8.11e-05	1.72e-10
	f = 4 ; fp = 1	8.49e-01	9.64e-02	9.67e-03	9.67e-04	9.67e-05	2.02e-09
	f = 4 ; fp = 2	6.68e-01	8.78e-02	8.86e-03	8.86e-04	8.86e-05	2.02e-09
	f = 4 ; fp = 3	5.70e-01	7.83e-02	7.94e-03	7.94e-04	7.94e-05	2.02e-09
f = fp	f = 1	3.37e-13	3.38e-13	3.39e-13	3.39e-13	3.38e-13	3.37e-13
	f = 2	4.38e-12	4.38e-12	4.38e-12	4.38e-12	4.39e-12	4.38e-12
	f = 3	1.72e-10	1.72e-10	1.72e-10	1.72e-10	1.72e-10	1.72e-10
	f = 4	2.02e-09	2.02e-09	2.02e-09	2.02e-09	2.02e-09	2.02e-09
f < fp	f = 1 ; fp = 2	6.46e-01	9.19e-04	7.68e-05	7.67e-06	7.67e-07	3.37e-13
	f = 1 ; fp = 3	7.90e-01	4.83e-03	2.93e-04	2.93e-05	2.93e-06	3.37e-13
	f = 1 ; fp = 4	8.51e-01	3.99e-02	6.66e-04	6.38e-05	6.38e-06	3.37e-13
	f = 2 ; fp = 3	5.80e-01	6.72e-02	7.07e-03	7.08e-04	7.08e-05	4.38e-12
	f = 2 ; fp = 4	6.70e-01	4.10e-03	3.40e-04	3.39e-05	3.39e-06	4.38e-12
	f = 3 ; fp = 4	5.70e-01	7.12e-02	7.30e-03	7.30e-04	7.30e-05	1.72e-10

Figure 4.21: Correction by multiplying on the Square with ϕ-FEM.

It would therefore seem that, overall, the smaller the perturbation applied (i.e. the smaller the ϵ), the more efficient the addition and multiplication correction solvers are in terms of accuracy. However, we would like to make a few comments on the results obtained:

- First of all, it would appear that, as with the standard FEM and ϕ-FEM solvers without correction, the more the solution varies (i.e. the larger f), the greater the error. This is a fairly intuitive result, since the more the solution varies, the more points are needed to approximate it.

- It would also seem that for $\epsilon = 1$ (i.e. a large perturbation), this parameter has a greater impact on the multiplicative corrector than on the additive corrector. We explained earlier the benefits of elevating the problem, which could be beneficial here. Results on elevation will be presented in the Section 4.3.2.2.

- In view of the results obtained here, it would also appear that, overall, correction by addition is more effective than correction by multiplication. Moreover, correction by addition has more advantages than correction by multiplication. In particular, if the solution cancels out on the domain, correction by multiplication will require elevating the problem sufficiently so that it no longer cancels out, unlike correction by addition.

- An interesting result can also be observed. Indeed, it seems that in the case where $f = f_p$, the multiplication correction with ϕ-FEM seems to approach the solution almost perfectly for all ϵ considered. In fact, in the homogeneous case, for $f = f_p$ the perturbation is identical to the solution (i.e. $P = u_{ex}$) and so the solution injected into the correction solvers is of the form

$$\tilde{\phi} = u_{ex} + \epsilon P = (1+\epsilon)\, u_{ex}$$

In the case of correction by multiplication, we have $\tilde{u} = \tilde{\phi}C$. So for $\tilde{u} = u_{ex}$, we must have

$$\tilde{\phi}C = u_{ex} \quad \Longleftrightarrow \quad (1+\epsilon)u_{ex}C = u_{ex}$$

So if the solution does not cancel out on Ω, we must have

$$C = \frac{1}{1+\epsilon} \quad \text{on } \Omega$$

By imposing $C = \frac{1}{1+\epsilon}$ on Γ for FEM instead of $C = 1$, we should get closer to the ϕ-FEM results obtained. We can see in Figure 4.22 and Figure 4.23 that we obtain the expected results for FEM by changing the boundary condition $C = 1$ to $C = \frac{1}{1+\epsilon}$.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
C=1	f = 1	2.78e-02	2.78e-03	2.78e-04	2.78e-05	2.78e-06	2.56e-14
	f = 2	3.72e-02	3.72e-03	3.72e-04	3.72e-05	3.72e-06	2.66e-14
	f = 3	4.66e-02	4.66e-03	4.66e-04	4.66e-05	4.66e-06	2.67e-14
	f = 4	5.42e-02	5.42e-03	5.42e-04	5.42e-05	5.42e-06	3.71e-14
C=1/(1+eps)	f = 1	2.97e-13	2.99e-13	2.98e-13	2.99e-13	2.98e-13	2.97e-13
	f = 2	2.80e-13	2.79e-13	2.79e-13	2.80e-13	2.80e-13	2.80e-13
	f = 3	2.68e-13	2.68e-13	2.68e-13	2.68e-13	2.68e-13	2.68e-13
	f = 4	2.61e-13	2.61e-13	2.61e-13	2.61e-13	2.61e-13	2.61e-13

Figure 4.22: Results by changing FEM boundary conditions on the circle.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
C=1	f = 1	2.78e-02	2.78e-03	2.78e-04	2.78e-05	2.78e-06	2.56e-14
	f = 2	3.72e-02	3.72e-03	3.72e-04	3.72e-05	3.72e-06	2.66e-14
	f = 3	4.66e-02	4.66e-03	4.66e-04	4.66e-05	4.66e-06	2.67e-14
	f = 4	5.42e-02	5.42e-03	5.42e-04	5.42e-05	5.42e-06	3.71e-14
C=1/(1+eps)	f = 1	2.48e-13	2.49e-13	2.48e-13	2.49e-13	2.47e-13	2.48e-13
	f = 2	2.49e-13	2.49e-13	2.49e-13	2.49e-13	2.49e-13	2.49e-13
	f = 3	2.49e-13	2.49e-13	2.49e-13	2.49e-13	2.49e-13	2.49e-13
	f = 4	2.49e-13	2.49e-13	2.49e-13	2.49e-13	2.49e-13	2.49e-13

Figure 4.23: Results by changing FEM boundary conditions on the square.

Remark. *It should be noted, however, that in practice, for example in the case where $\tilde{\phi}$ is a ϕ-FEM solution or an FNO output, this case is not very realistic. There's no reason to expect the form of the perturbation created by the ϕ-FEM solver or by the FNO to be exactly identical to the solution under consideration.*

Results on the non-homogeneous case :
Then, we consider the non-homogeneous case (i.e. with $\varphi = 1$). The aim here is the same as in the homogeneous case, test correction by addition (without IPP) and correction by multiplication by varying the amplitude of the perturbation (in other words, by varying ϵ). We'll consider the case of the circle and the square for standard FEM and ϕ-FEM methods, and we'll try to separate the cases according to the frequencies considered. In other words, for $f, f_p \in \{1,2,3,4\}$, we're interested in the following three cases. The first is the case where the solution frequency is greater than the perturbation frequency ($f > f_p$), i.e. a highly variable solution and a less variable perturbation. The second is where the solution and perturbation frequencies are equal ($f = f_p$), i.e. the solution and perturbation have the same variability. The last category covers cases where the perturbation is "nastier" than the solution, i.e. it has a higher frequency than the solution ($f < f_p$).

- **Results on the Circle :**

  We consider here the Circle problem with the solution defined in Section 4.1.1.1. Here, we consider correction by adding (without IPP) with standard FEM (Figure 4.24) and with ϕ-FEM (Figure 4.25).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	2.10e-03	2.10e-04	2.10e-05	2.10e-06	2.10e-07	9.95e-11
	f = 3 ; fp = 1	2.10e-03	2.10e-04	2.10e-05	2.10e-06	2.10e-07	7.08e-11
	f = 3 ; fp = 2	6.62e-03	6.62e-04	6.62e-05	6.62e-06	6.62e-07	7.08e-11
	f = 4 ; fp = 1	2.10e-03	2.10e-04	2.10e-05	2.10e-06	2.10e-07	6.02e-11
	f = 4 ; fp = 2	6.62e-03	6.62e-04	6.62e-05	6.62e-06	6.62e-07	6.02e-11
	f = 4 ; fp = 3	1.41e-02	1.41e-03	1.41e-04	1.41e-05	1.41e-06	6.02e-11
f = fp	f = 1	2.10e-03	2.10e-04	2.10e-05	2.10e-06	2.10e-07	2.90e-10
	f = 2	6.62e-03	6.62e-04	6.62e-05	6.62e-06	6.62e-07	9.95e-11
	f = 3	1.41e-02	1.41e-03	1.41e-04	1.41e-05	1.41e-06	7.08e-11
	f = 4	2.42e-02	2.42e-03	2.42e-04	2.42e-05	2.42e-06	6.02e-11
f < fp	f = 1 ; fp = 2	6.62e-03	6.62e-04	6.62e-05	6.62e-06	6.62e-07	2.90e-10
	f = 1 ; fp = 3	1.41e-02	1.41e-03	1.41e-04	1.41e-05	1.41e-06	2.90e-10
	f = 1 ; fp = 4	2.42e-02	2.42e-03	2.42e-04	2.42e-05	2.42e-06	2.90e-10
	f = 2 ; fp = 3	1.41e-02	1.41e-03	1.41e-04	1.41e-05	1.41e-06	9.95e-11
	f = 2 ; fp = 4	2.42e-02	2.42e-03	2.42e-04	2.42e-05	2.42e-06	9.95e-11
	f = 3 ; fp = 4	2.42e-02	2.42e-03	2.42e-04	2.42e-05	2.42e-06	7.08e-11

Figure 4.24: Correction by adding on the Circle with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	7.94e-04	7.94e-05	7.94e-06	7.94e-07	7.95e-08	4.13e-11
	f = 3 ; fp = 1	7.93e-04	7.93e-05	7.93e-06	7.94e-07	7.94e-08	6.79e-11
	f = 3 ; fp = 2	6.36e-03	6.36e-04	6.36e-05	6.36e-06	6.36e-07	6.79e-11
	f = 4 ; fp = 1	7.93e-04	7.93e-05	7.93e-06	7.92e-07	7.90e-08	7.32e-10
	f = 4 ; fp = 2	6.35e-03	6.35e-04	6.35e-05	6.35e-06	6.35e-07	7.32e-10
	f = 4 ; fp = 3	2.10e-02	2.10e-03	2.10e-04	2.10e-05	2.10e-06	7.32e-10
f = fp	f = 1	7.95e-04	7.95e-05	7.95e-06	7.96e-07	7.96e-08	1.19e-10
	f = 2	6.36e-03	6.36e-04	6.36e-05	6.36e-06	6.36e-07	4.13e-11
	f = 3	2.10e-02	2.10e-03	2.10e-04	2.10e-05	2.10e-06	6.79e-11
	f = 4	4.77e-02	4.77e-03	4.77e-04	4.77e-05	4.77e-06	7.32e-10
f < fp	f = 1 ; fp = 2	6.37e-03	6.37e-04	6.37e-05	6.37e-06	6.37e-07	1.19e-10
	f = 1 ; fp = 3	2.10e-02	2.10e-03	2.10e-04	2.10e-05	2.10e-06	1.19e-10
	f = 1 ; fp = 4	4.79e-02	4.79e-03	4.79e-04	4.79e-05	4.79e-06	1.19e-10
	f = 2 ; fp = 3	2.10e-02	2.10e-03	2.10e-04	2.10e-05	2.10e-06	4.13e-11
	f = 2 ; fp = 4	4.78e-02	4.78e-03	4.78e-04	4.78e-05	4.78e-06	4.13e-11
	f = 3 ; fp = 4	4.78e-02	4.78e-03	4.78e-04	4.78e-05	4.78e-06	6.79e-11

Figure 4.25: Correction by adding on the Circle with ϕ-FEM.

Then, we consider correction by multiplying with standard FEM (Figure 4.26) and with ϕ-FEM (Figure 4.27).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	1.20e+00	1.03e-01	1.00e-02	1.00e-03	1.00e-04	2.42e-13
	f = 3 ; fp = 1	1.02e+00	9.96e-02	9.91e-03	9.91e-04	9.91e-05	2.47e-13
	f = 3 ; fp = 2	9.91e-01	9.26e-02	9.03e-03	9.00e-04	9.00e-05	2.47e-13
	f = 4 ; fp = 1	9.70e-01	9.92e-02	9.92e-03	9.92e-04	9.92e-05	2.46e-13
	f = 4 ; fp = 2	9.67e-01	9.37e-02	9.30e-03	9.29e-04	9.29e-05	2.46e-13
	f = 4 ; fp = 3	8.97e-01	8.55e-02	8.35e-03	8.33e-04	8.33e-05	2.46e-13
f = fp	f = 1	6.61e-01	1.02e-01	1.06e-02	1.07e-03	1.07e-04	2.17e-13
	f = 2	5.49e-01	8.74e-02	9.13e-03	9.17e-04	9.17e-05	2.42e-13
	f = 3	5.15e-01	8.22e-02	8.60e-03	8.64e-04	8.64e-05	2.47e-13
	f = 4	4.95e-01	7.96e-02	8.34e-03	8.37e-04	8.38e-05	2.46e-13
f < fp	f = 1 ; fp = 2	4.42e-01	1.04e-01	1.14e-02	1.15e-03	1.15e-04	2.17e-13
	f = 1 ; fp = 3	8.20e-01	2.53e-02	1.84e-03	1.79e-04	1.79e-05	2.17e-13
	f = 1 ; fp = 4	8.56e-01	7.39e-02	9.34e-03	9.56e-04	9.58e-05	2.17e-13
	f = 2 ; fp = 3	4.31e-01	7.23e-02	7.27e-03	7.27e-04	7.27e-05	2.42e-13
	f = 2 ; fp = 4	7.27e-01	1.21e-01	1.23e-02	1.23e-03	1.23e-04	2.42e-13
	f = 3 ; fp = 4	4.40e-01	6.91e-02	7.13e-03	7.15e-04	7.16e-05	2.47e-13

Figure 4.26: Correction by multiplying on the Circle with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	1.16e+00	1.01e-01	9.79e-03	9.76e-04	9.76e-05	2.92e-11
	f = 3 ; fp = 1	9.95e-01	9.71e-02	9.67e-03	9.67e-04	9.67e-05	2.91e-10
	f = 3 ; fp = 2	9.66e-01	9.07e-02	8.85e-03	8.83e-04	8.83e-05	2.91e-10
	f = 4 ; fp = 1	9.45e-01	9.67e-02	9.68e-03	9.68e-04	9.68e-05	1.52e-09
	f = 4 ; fp = 2	9.56e-01	9.25e-02	9.16e-03	9.15e-04	9.15e-05	1.52e-09
	f = 4 ; fp = 3	8.78e-01	8.50e-02	8.33e-03	8.31e-04	8.31e-05	1.52e-09
f = fp	f = 1	6.65e-01	1.01e-01	1.04e-02	1.05e-03	1.05e-04	9.46e-11
	f = 2	5.74e-01	8.67e-02	9.02e-03	9.05e-04	9.05e-05	2.92e-11
	f = 3	5.28e-01	8.22e-02	8.55e-03	8.58e-04	8.59e-05	2.91e-10
	f = 4	5.08e-01	8.05e-02	8.39e-03	8.42e-04	8.42e-05	1.52e-09
f < fp	f = 1 ; fp = 2	5.65e-01	1.96e-01	1.42e-02	1.45e-03	1.46e-04	9.46e-11
	f = 1 ; fp = 3	8.61e-01	2.55e-01	2.45e-02	2.30e-03	2.28e-04	9.46e-11
	f = 1 ; fp = 4	9.44e-01	2.72e+00	3.13e-02	3.46e-03	3.49e-04	9.46e-11
	f = 2 ; fp = 3	4.61e-01	8.23e-02	8.10e-03	8.13e-04	8.14e-05	2.92e-11
	f = 2 ; fp = 4	7.56e-01	1.52e-01	1.49e-02	1.49e-03	1.49e-04	2.92e-11
	f = 3 ; fp = 4	4.54e-01	7.27e-02	7.60e-03	7.63e-04	7.64e-05	2.91e-10

Figure 4.27: Correction by multiplying on the Circle with ϕ-FEM.

- **Results on the Square :**

  We consider here the Square problem with the solution defined in Section 4.1.2.1. Here, we consider correction by adding (without IPP) with standard FEM (Figure 4.28) and with ϕ-FEM (Figure 4.29).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	1.79e-11
	f = 3 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	1.19e-11
	f = 3 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	1.19e-11
	f = 4 ; fp = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	8.32e-12
	f = 4 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	8.32e-12
	f = 4 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	8.32e-12
f = fp	f = 1	2.61e-03	2.61e-04	2.61e-05	2.61e-06	2.61e-07	4.74e-11
	f = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	1.79e-11
	f = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	1.19e-11
	f = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	8.32e-12
f < fp	f = 1 ; fp = 2	1.04e-02	1.04e-03	1.04e-04	1.04e-05	1.04e-06	4.74e-11
	f = 1 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	4.74e-11
	f = 1 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	4.74e-11
	f = 2 ; fp = 3	2.34e-02	2.34e-03	2.34e-04	2.34e-05	2.34e-06	1.79e-11
	f = 2 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	1.79e-11
	f = 3 ; fp = 4	4.12e-02	4.12e-03	4.12e-04	4.12e-05	4.12e-06	1.19e-11

Figure 4.28: Correction by adding on the Square with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	1.79e-03	1.79e-04	1.79e-05	1.79e-06	1.79e-07	9.83e-12
	f = 3 ; fp = 1	1.79e-03	1.79e-04	1.79e-05	1.79e-06	1.79e-07	1.16e-10
	f = 3 ; fp = 2	1.45e-02	1.45e-03	1.45e-04	1.45e-05	1.45e-06	1.16e-10
	f = 4 ; fp = 1	1.79e-03	1.79e-04	1.79e-05	1.79e-06	1.79e-07	1.13e-09
	f = 4 ; fp = 2	1.45e-02	1.45e-03	1.45e-04	1.45e-05	1.45e-06	1.13e-09
	f = 4 ; fp = 3	3.94e-02	3.94e-03	3.94e-04	3.94e-05	3.94e-06	1.13e-09
f = fp	f = 1	1.79e-03	1.79e-04	1.79e-05	1.79e-06	1.79e-07	1.42e-11
	f = 2	1.45e-02	1.45e-03	1.45e-04	1.45e-05	1.45e-06	9.83e-12
	f = 3	3.94e-02	3.94e-03	3.94e-04	3.94e-05	3.94e-06	1.16e-10
	f = 4	7.79e-02	7.79e-03	7.79e-04	7.79e-05	7.79e-06	1.13e-09
f < fp	f = 1 ; fp = 2	1.45e-02	1.45e-03	1.45e-04	1.45e-05	1.45e-06	1.42e-11
	f = 1 ; fp = 3	3.94e-02	3.94e-03	3.94e-04	3.94e-05	3.94e-06	1.42e-11
	f = 1 ; fp = 4	7.78e-02	7.78e-03	7.78e-04	7.78e-05	7.78e-06	1.42e-11
	f = 2 ; fp = 3	3.94e-02	3.94e-03	3.94e-04	3.94e-05	3.94e-06	9.83e-12
	f = 2 ; fp = 4	7.78e-02	7.78e-03	7.78e-04	7.78e-05	7.78e-06	9.83e-12
	f = 3 ; fp = 4	7.78e-02	7.78e-03	7.78e-04	7.78e-05	7.78e-06	1.16e-10

Figure 4.29: Correction by adding on the Square with ϕ-FEM.

Then, we consider correction by multiplying with standard FEM (Figure 4.30) and with ϕ-FEM (Figure 4.31).

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	6.42e-01	8.64e-02	8.68e-03	8.67e-04	8.67e-05	2.16e-13
	f = 3 ; fp = 1	7.86e-01	9.36e-02	9.39e-03	9.40e-04	9.40e-05	2.29e-13
	f = 3 ; fp = 2	5.67e-01	7.97e-02	8.02e-03	8.02e-04	8.02e-05	2.29e-13
	f = 4 ; fp = 1	8.44e-01	9.66e-02	9.67e-03	9.67e-04	9.67e-05	2.35e-13
	f = 4 ; fp = 2	6.56e-01	8.78e-02	8.81e-03	8.81e-04	8.81e-05	2.35e-13
	f = 4 ; fp = 3	5.60e-01	7.74e-02	7.79e-03	7.79e-04	7.79e-05	2.35e-13
f = fp	f = 1	3.76e-01	7.12e-02	7.66e-03	7.70e-04	7.70e-05	1.72e-13
	f = 2	3.95e-01	7.31e-02	7.81e-03	7.85e-04	7.86e-05	2.16e-13
	f = 3	4.09e-01	7.47e-02	7.98e-03	8.02e-04	8.02e-05	2.29e-13
	f = 4	4.22e-01	7.64e-02	8.13e-03	8.17e-04	8.18e-05	2.35e-13
f < fp	f = 1 ; fp = 2	5.23e-01	7.12e-02	7.39e-03	7.39e-04	7.39e-05	1.72e-13
	f = 1 ; fp = 3	6.62e-01	9.68e-03	9.73e-04	9.96e-05	9.99e-06	1.72e-13
	f = 1 ; fp = 4	7.18e-01	5.27e-02	4.69e-03	4.69e-04	4.69e-05	1.72e-13
	f = 2 ; fp = 3	5.35e-01	6.31e-02	6.51e-03	6.52e-04	6.52e-05	2.16e-13
	f = 2 ; fp = 4	5.88e-01	7.69e-02	7.98e-03	7.99e-04	7.99e-05	2.16e-13
	f = 3 ; fp = 4	5.42e-01	6.72e-02	6.81e-03	6.82e-04	6.82e-05	2.29e-13

Figure 4.30: Correction by multiplying on the Square with standard FEM.

		eps = 1	eps = 0.1	eps = 0.01	eps = 0.001	eps = 0.0001	eps = 0.0
f > fp	f = 2 ; fp = 1	6.18e-01	8.29e-02	8.35e-03	8.35e-04	8.35e-05	2.90e-11
	f = 3 ; fp = 1	7.58e-01	9.00e-02	9.01e-03	9.01e-04	9.01e-05	1.78e-10
	f = 3 ; fp = 2	5.60e-01	7.69e-02	7.74e-03	7.74e-04	7.74e-05	1.78e-10
	f = 4 ; fp = 1	8.15e-01	9.28e-02	9.31e-03	9.31e-04	9.31e-05	1.46e-09
	f = 4 ; fp = 2	6.44e-01	8.47e-02	8.51e-03	8.51e-04	8.51e-05	1.46e-09
	f = 4 ; fp = 3	5.61e-01	7.51e-02	7.59e-03	7.59e-04	7.59e-05	1.46e-09
f = fp	f = 1	3.85e-01	6.89e-02	7.51e-03	7.56e-04	7.57e-05	2.02e-11
	f = 2	3.97e-01	7.08e-02	7.58e-03	7.63e-04	7.64e-05	2.90e-11
	f = 3	4.17e-01	7.30e-02	7.77e-03	7.81e-04	7.81e-05	1.78e-10
	f = 4	4.38e-01	7.46e-02	7.98e-03	8.03e-04	8.03e-05	1.46e-09
f < fp	f = 1 ; fp = 2	6.35e-01	1.44e-01	9.08e-03	8.95e-04	8.94e-05	2.02e-11
	f = 1 ; fp = 3	7.85e-01	3.12e-01	1.33e-02	1.56e-03	1.59e-04	2.02e-11
	f = 1 ; fp = 4	8.59e-01	6.36e-01	1.22e-02	1.09e-03	1.09e-04	2.02e-11
	f = 2 ; fp = 3	5.65e-01	6.67e-02	6.65e-03	6.64e-04	6.63e-05	2.90e-11
	f = 2 ; fp = 4	6.56e-01	8.07e-02	8.15e-03	8.13e-04	8.13e-05	2.90e-11
	f = 3 ; fp = 4	5.65e-01	6.86e-02	6.93e-03	6.93e-04	6.93e-05	1.78e-10

Figure 4.31: Correction by multiplying on the Square with ϕ-FEM.

In view of the results obtained, it would appear that the conclusions are the same as for the homogeneous case. Except for the case where $f = f_p$, because in the case where the solution is non-homogeneous (i.e. $u_{ex} = g$ on Γ), the perturbation P is no longer equal to the solution u_{ex} because $P = 0$ on Γ.

4.3.2.2 Results on the elevated problem

On testera que en homogène ici
Résultats sur le rehaussement + montrer que quand la solution s'annule c'est bcp mieux

4.3.3 Correction on ϕ-FEM solution

Then we'll look at correcting a disturbed solution for which we don't know the form of the perturbation. A ϕ-FEM solution can be considered for injection into correction solvers.

4.3.4 Correction with FNO

4.3.5 Correction with other networks

4.3.5.1 Multiperceptron

4.3.5.2 PINNs

5 Conclusion

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

Bibliography

References

[1] Allal Bedlaoui. *Les éléments finis : de la théorie à la pratique*. URL: https://www.academia.edu/36497995/Les_%C3%A9l%C3%A9ments_finis_de_la_th%C3%A9orie_%C3%A0_la_pratique (visited on 08/16/2023).

[2] Allal Bedlaoui. *Les éléments finis : de la théorie à la pratique*. URL: https://www.academia.edu/36497995/Les_%C3%A9l%C3%A9ments_finis_de_la_th%C3%A9orie_%C3%A0_la_pratique (visited on 08/16/2023).

[3] Erik Burman. "Ghost penalty". In: *Comptes Rendus. Mathématique* 348.21 (2010). Number: 21-22, 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).

[4] Erik Burman et al. "CutFEM: Discretizing geometry and partial differential equations". In: *International Journal for Numerical Methods in Engineering* 104.7 (2015). Number: 7 _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).

[5] Stéphane Cotin et al. *ϕ-FEM: an efficient simulation tool using simple meshes for problems in structure mechanics and heat transfer*.

[6] 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). Number: 3, 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).

[7] 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). Number: 1 _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).

[8] 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). Number: 2 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).

[9] Zongyi Li et al. *Neural Operator: Graph Kernel Network for Partial Differential Equations*. Mar. 6, 2020. DOI: 10.48550/arXiv.2003.03485. arXiv: 2003.03485[cs,math,stat]. URL: http://arxiv.org/abs/2003.03485 (visited on 08/17/2023).

[10] Zongyi Li et al. *Fourier Neural Operator for Parametric Partial Differential Equations*. Issue: arXiv:2010.08895. 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).

[11] Zongyi Li et al. *Fourier Neural Operator with Learned Deformations for PDEs on General Geometries*. July 11, 2022. DOI: 10.48550/arXiv.2207.05209. arXiv: 2207.05209[cs,math]. URL: http://arxiv.org/abs/2207.05209 (visited on 08/17/2023).

[12] Zongyi Li et al. *Physics-Informed Neural Operator for Learning Partial Differential Equations*. July 29, 2023. DOI: 10.48550/arXiv.2111.03794. arXiv: 2111.03794[cs,math]. URL: http://arxiv.org/abs/2111.03794 (visited on 08/17/2023).

[13] *Méthodes numériques pour les EDP*. URL: https://feelpp.github.io/csmi-edp/#/ (visited on 08/16/2023).

[14] Nicolas Moës and Ted Belytschko. "X-FEM, de nouvelles frontières pour les éléments finis". In: *Revue Européenne des Éléments Finis* 11.2 (Jan. 2002). Number: 2-4, 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).

[15] Alfio Quarteroni et al. *Méthodes numériques: algorithmes, analyse et applications*. Ed. revue et augmentée. Milan: Springer, 2007. ISBN: 978-88-470-0495-5.