MIMESIS

UNIVERSITY OF STRASBOURG

MASTER CSMI

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

Authors:
Frédérique Lecourtier

Supervisors:
Michel Duprez
Emmanuel Franck
Vanessa Lleras

Inria

Date: July 11, 2023

Contents

PLAN :

1. Intro : Ce stage est un stage de M2, dans le cadre du master CSMI. C'est la suite d'un projet effectué pendant le premier semestre.... explication rapide du projet (colab entre Cemosis et Mimesis dans le but de blablabla)
   (a) Présentation de Mimesis
   (b) Contexte : Mimesis est spécialisé dans la simulation en temps réel de blabla. C'est pourquoi ils ont développé une méthode appelée PhiFEM (explication des intérêts de PhiFEM).
   (c) Objectifs : On cherche ici à améliorer la précision de la solution ainsi que les temps pour l'obtenir en passant par le biais d'un FNO que l'on va entraîner avec des solutions PhiFEM qui sont bien adapté à ce type de réseau de neurones, due aux grilles cartésienne.
2. Finite Element Methods (FEM)
   (a) Standard FEM
   (b) PhiFEM
3. Fourier Neural Operator (FNO)
4. Correction :
   Présentation des différentes méthodes de Correction (2 cas tests : solution analytique, sol FNO) / Legendre / Modèle Dense ?
   On veut mettre un schéma qui explique Entrainement du FNO -> Sortie -> Correction (-> Legendre ?)
5. Résultats ?
6. Conclu
7. Bibliography
8. Appendix (Organisation of the repository, Documentation, Github actions ...)

1 Introduction

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

1.1 Scientific Context

Finite element methods (FEM) are used to solve partial differential equations numerically. These can, for example, represent analytically the dynamic behavior of certain physical systems (mechanical, thermodynamic, acoustic, etc.). Among other things, it is a discrete algorithm for determining the approximate solution of a partial differential equation (PDE) on a compact domain with boundary conditions.
The standard FEM method, which requires precise meshing of the domain under consideration and, in particular, fitting with its boundary, has its limitations. In particular, in the medical field, meshing complex and evolving geometries such as organs (e.g. the liver) can be very costly. More specifically, in the application context of creating real-time digital twins of an organ, the standard FEM method would require complete remeshing of the organ each time it is deformed, which in practice is not workable.
This is why other methods, known as non-conformal finite element methods, have emerged in the last few years. These include CutFEM ref 6 or XFEM ref 18, 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 ref 12. But the method has been extended to Neumann conditions ref 8 and then to solve various mechanical problems, including linear elasticity and heat transfer problems, and even to solve the Stokes problem ref 9.

1.2 Presentation of Mimesis

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

1.3 Objectives

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

1.4 Deliverables

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

2 Finite Element Methods (FEMs)

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

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

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

2.1 Standard FEM

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

2.1.1 Some notions of functional analysis.

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

2.1.2 General principle of the method

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

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

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

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

The variational problem can then be approached by :

Approach Problem :

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

As V_h is of finite dimension, we can consider a basis $(\varphi_1, \ldots, \varphi_{N_h})$ 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 :

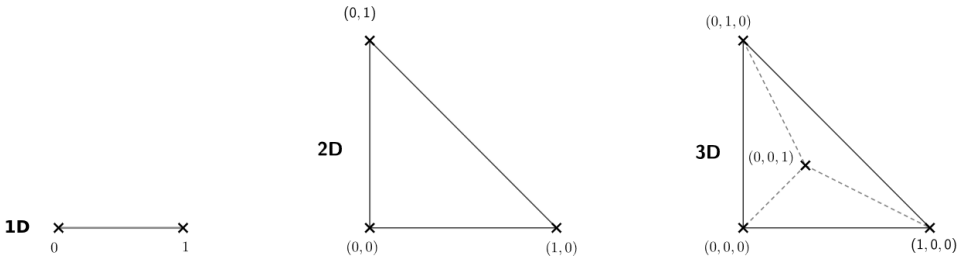

Figure 2.1: Example of reference Elements

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 (see Fig 2.2):

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

where N_e is the number of elements.
In 2D, these elements can be triangles or rectangles. In 3D, they can be tetrahedrons, parallelepipeds or prisms.

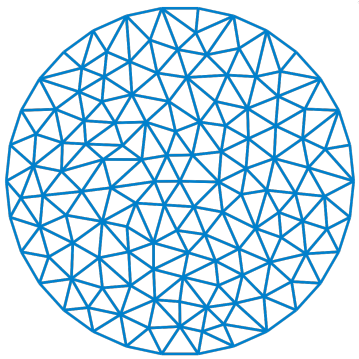

Figure 2.2: Example of a triangular mesh on a circles

2.1.3.5 Geometric transformation

A mesh is generated by

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

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

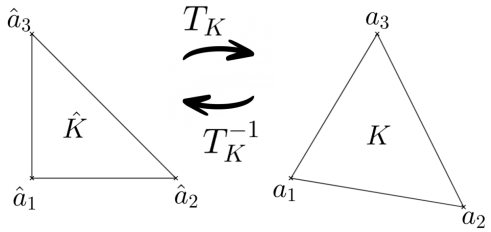

Figure 2.3: Geometric transformation applied to a triangle

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

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

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

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

In particular, we have

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

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

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

Remark. *In the following, we'll 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 ?

2.2 ϕ-FEM

TO COMPLETE !

3 Fourier Neural Operator (FNO)

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

3.1 Architecture of the FNO

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

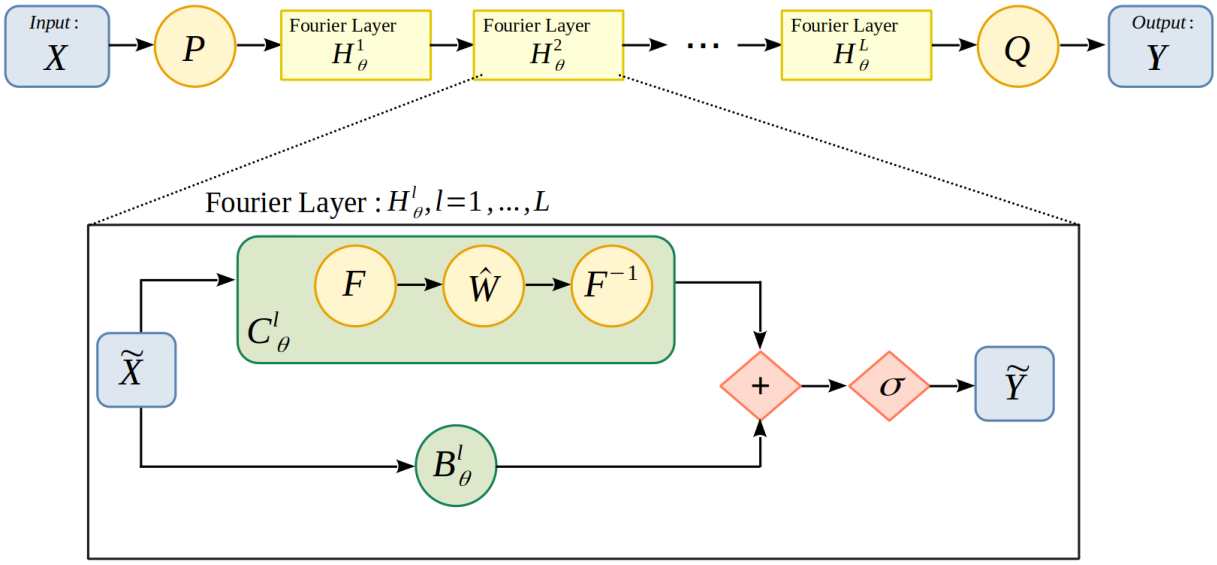

Figure 3.1: Architecture of the FNO

The architecture of the FNO is as follows:

$$G_\theta = Q \circ \mathscr{H}_\theta^L \circ \cdots \circ \mathscr{H}_\theta^1 \circ P$$

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

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

3.2 Fourier Layer structure

Each Fourier layer is divided into two sublayers:

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

where

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

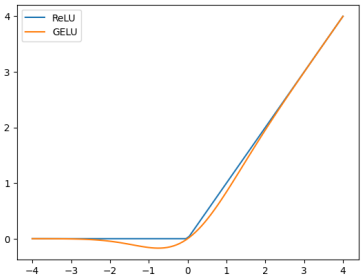

Figure 3.2: Activation functions used

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

3.2.1 Convolution sublayer

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

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

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

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

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

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

The 2D DFT is defined by :

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

The inverse of the 2D DFT is defined by :

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

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

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

Let

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

Thus

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

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

  as the sum of a geometric sequence.

We deduce that

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

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

3.2.2 Bias subLayer

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

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

3.3 Some details on the convolution sublayer

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

3.3.1 Border issues

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

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

with

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

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

3.3.2 FFT

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

3.3.3 Real DFT

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

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

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

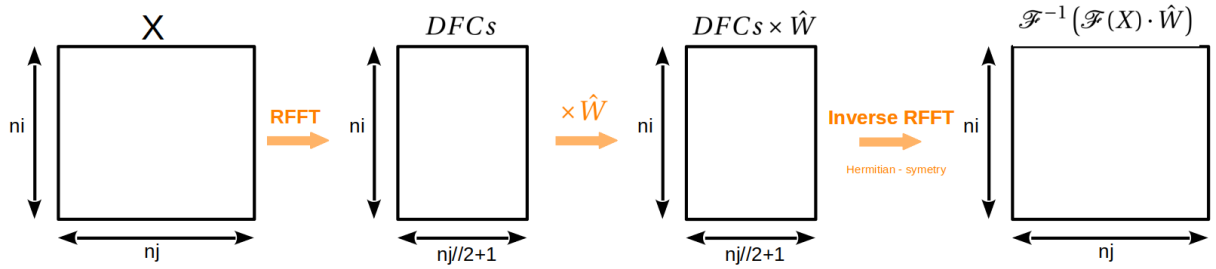

Figure 3.3: RFFT with Hermitian-symmetry scheme

3.3.4 Low pass filter

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

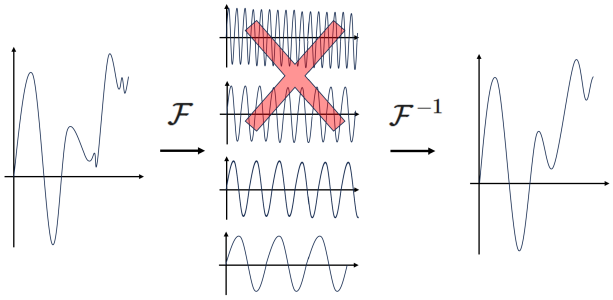

Figure 3.4: Low pass filter

3.3.5 Global aspect of the FNO

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

3.4 Application

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

4 Correction

Réfléchir présenter sol considéré et domaines !

5 Numerical Results ?

6 Conclusion

7 Biblio