ϕ-FEM-FNO: A NEW APPROACH TO TRAIN A NEURAL OPERATOR AS A FAST PDE SOLVER FOR VARIABLE GEOMETRIES.

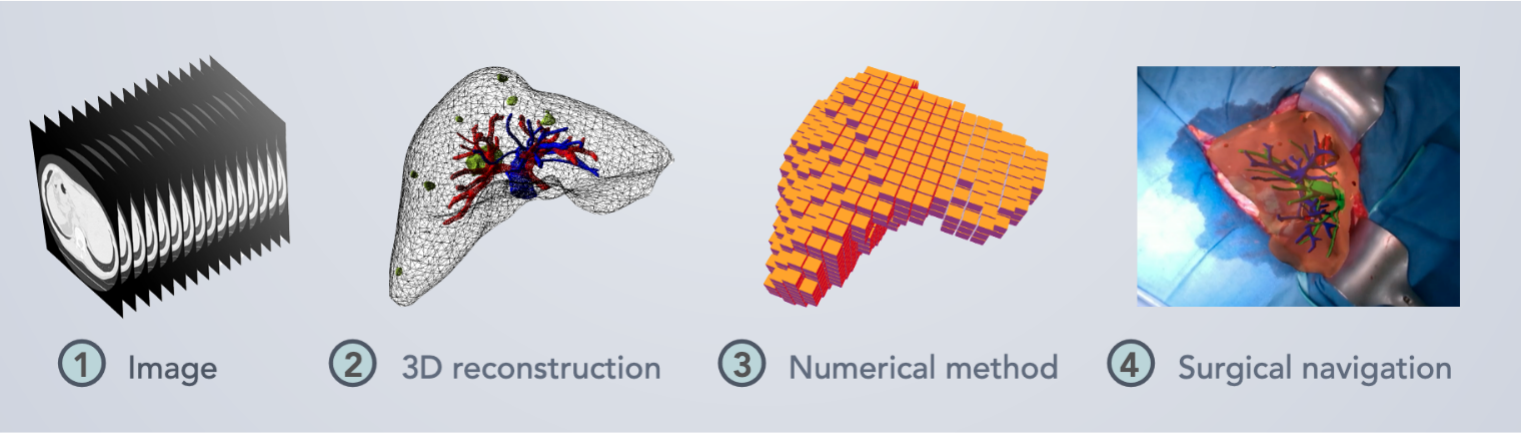
04/06/2024 ECCOMAS 2024, Lisbon

Michel Duprez[1], Vanessa Lleras[2], Alexei Lozinski[3], Vincent Vigon[4] and **Killian Vuillemot**[1,2]

[1] Inria, Strasbourg, France, [2] IMAG, Montpellier, France,
[3] LMB, Besançon, France, [4] IRMA, Strasbourg, France.

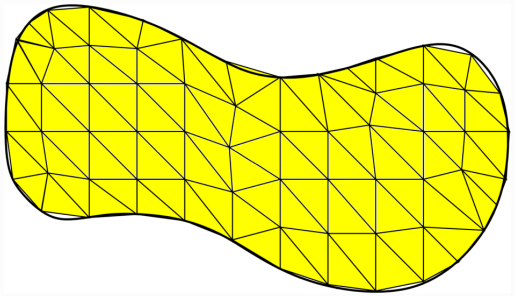
Objectives

Develop **real-time, patient specific digital twins** for computer-aided surgical interventions.

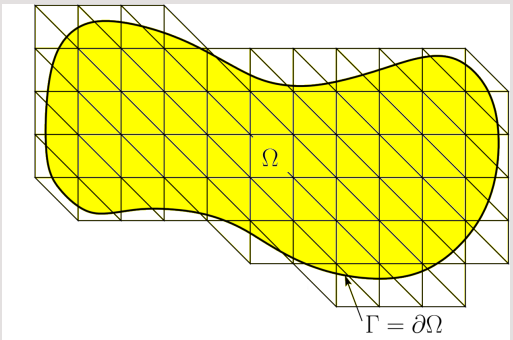


- Simulation of the deformations of organs : PDEs $\longrightarrow$ FEMs,
- Complex geometries $\longrightarrow$ Unfitted FEMs,
- Real-time constructions $\longrightarrow$ machine learning techniques.

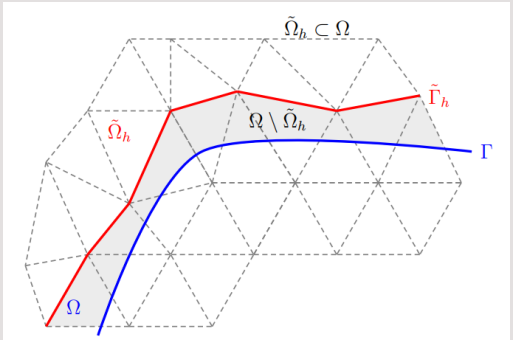
Very short story of FEMs



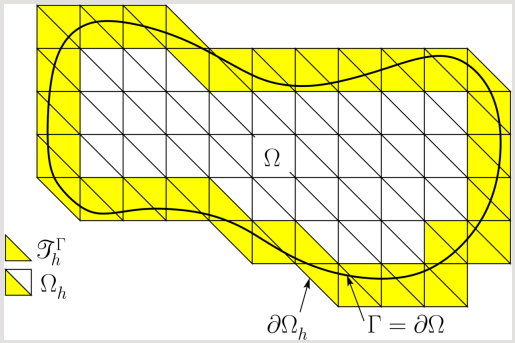
(a) Standard FEM (Clough 60s).



(b) XFEM (Moes and al., 2006)
$\longrightarrow$ Non-classical shape functions,
CutFEM (Burman, Hansbo, 2010-2014)
$\longrightarrow$ cut cells and partial integrals.



(c) Shifted Boundary method (Main, Scovazzi, 2017)
$\longrightarrow$ Taylor development near the boundary.



(d) **ϕ-FEM (Duprez and Lozinski, 2020)**
$\longrightarrow$ **Level-set function**

Problems on complex shapes $\longrightarrow$ unfitted FEMs

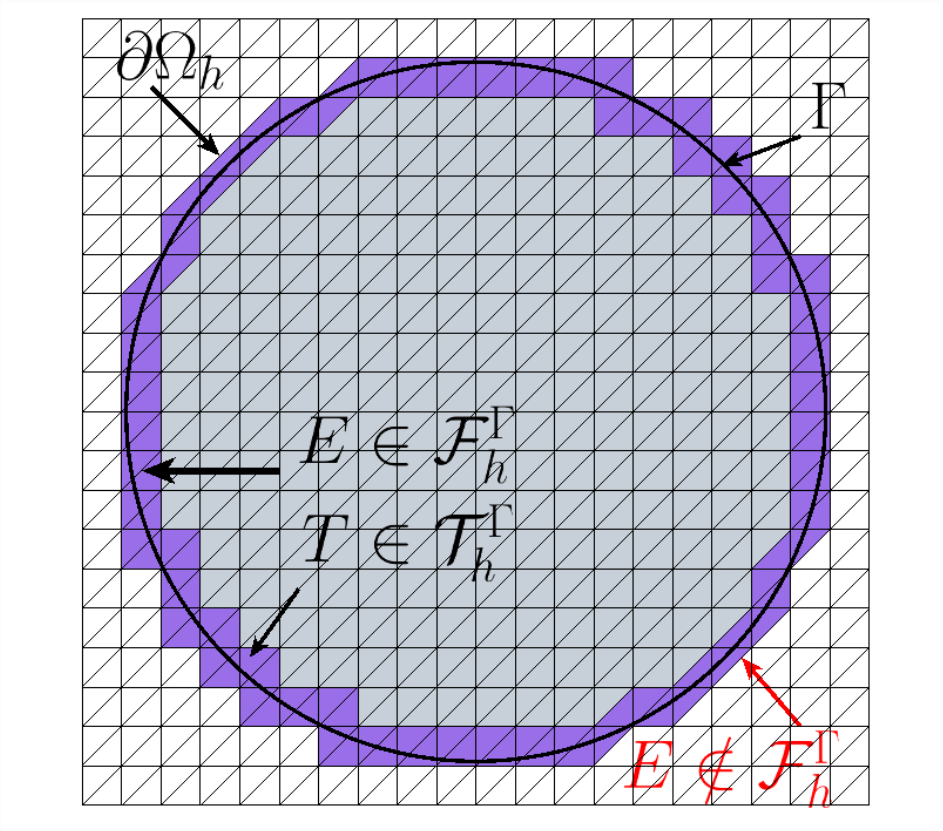
The idea of ϕ-FEM

Level-set function

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

The spaces

- $\mathcal{T}_h$: ϕ-FEM mesh,
- $\mathcal{T}_h^\Gamma$: cells of $\mathcal{T}_h$ cut by the boundary (purple triangles),
- $\mathcal{F}_h^\Gamma$: internal facets of $\mathcal{T}_h^\Gamma$.



Example with $\phi(x, y) = -1 + x^2 + y^2$.

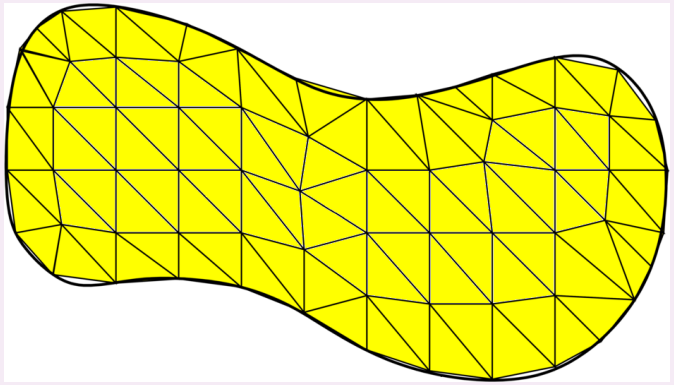
Standard FEM VS ϕ-FEM

Example (Poisson-Dirichlet equation)

$$-\Delta u = f \quad \text{in } \Omega, \qquad u = g \quad \text{on } \Gamma\,.$$

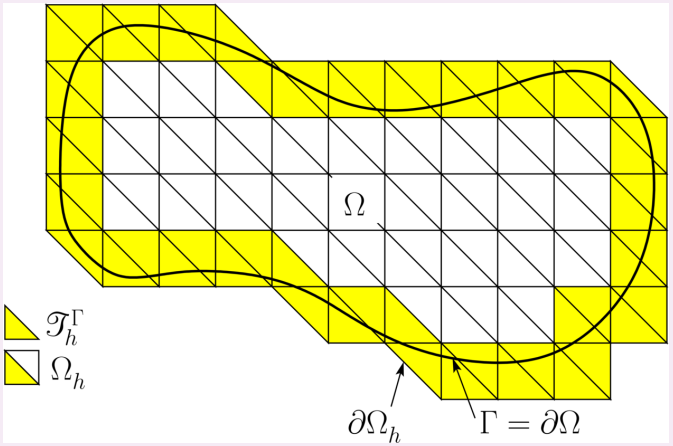
Standard FEM

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



ϕ-FEM

$$\text{Find } w \text{ s.t. } \begin{cases} -\Delta u = f\,, \text{ in } \Omega_h\,, \\ u = \phi w + g\,, \text{ in } \Omega_h\,. \end{cases}$$



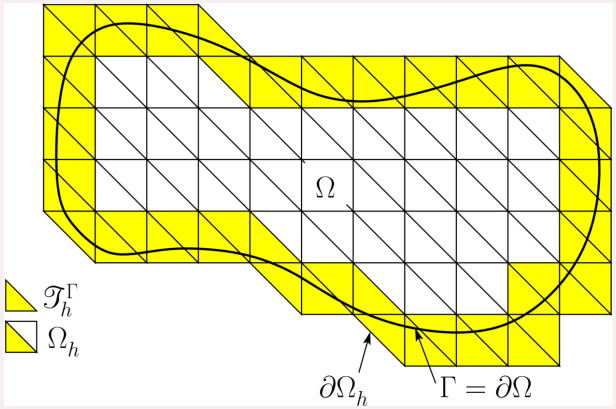
Example (Poisson-Dirichlet equation)

$$-\Delta u = f \quad \text{in } \Omega, \quad u = g \quad \text{on } \Gamma\,.$$

ϕ-FEM scheme

Find w_h such that for all v_h,

$$\int_{\Omega_h} \nabla(\phi_h w_h + g_h) \cdot \nabla(\phi_h v_h) - \int_{\partial\Omega_h} \frac{\partial}{\partial n}(\phi_h w_h + g_h)\phi_h v_h + \boxed{\text{stabs}} = \int_{\Omega_h} f_h \phi_h v_h - \boxed{\text{stabs}}\,.$$



Who are «stabs»?

- First order : Ghost penalty ($\mathcal{S}_1$),
- Second order : least square imposition of the strong formulation on $\mathcal{T}_h^\Gamma$ ($\mathcal{S}_2$).

$$\int_{\Omega_h} \nabla(\phi_h w_h + g_h) \cdot \nabla(\phi_h v_h) - \int_{\partial\Omega_h} \frac{\partial}{\partial n}(\phi_h w_h + g_h)\phi_h v_h$$

$$\underbrace{+ \sigma h \sum_{E \in \mathcal{F}_\Gamma} \int_E \left[\frac{\partial}{\partial n}(\phi_h w_h + g_h)\right]\left[\frac{\partial}{\partial n}(\phi_h v_h)\right]}_{(\mathcal{S}_1)\text{ : Ghost penalty, jump over the facets}}$$

$$\underbrace{+ \sigma h^2 \sum_{T \in \mathcal{T}_h^\Gamma} \int_T \Delta(\phi_h w_h + g_h)\Delta(\phi_h w_h)}_{(\mathcal{S}_2)\text{ : least square imposition of the governing equation}} = \int_{\Omega_h} f_h \phi_h v_h$$

$$\underbrace{-\sigma h^2 \sum_{T \in \mathcal{T}_h^\Gamma} \int_T f\Delta(\phi_h w_h)}_{\mathcal{S}_2\text{'s friend}} .$$

Interests of the method

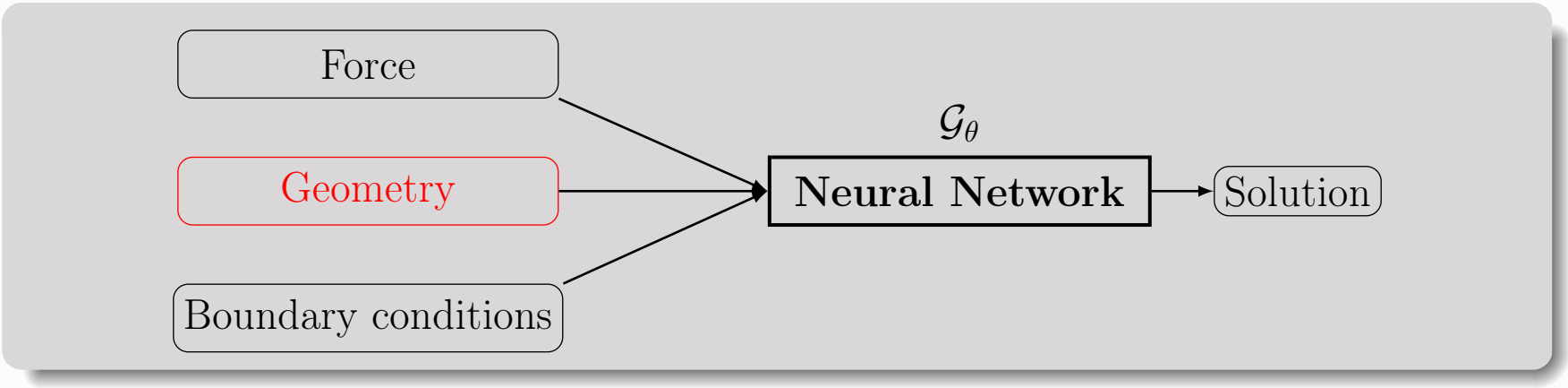
- Optimal convergence in L^2 and H^1 norms,
- Easy to implement : standard shape functions, no cut cells $\longrightarrow$ standard quadrature rules,
- Acceptable conditioning of the finite element matrix,
- No need to generate a conforming mesh.

Other schemes

- Dirichlet or Neumann boundary conditions,
- Linear elasticity problems,
- Stokes problem,
- Heat equation.

In the context of real-time simulations, we need
quasi-instantaneous results.

- ϕ-FEM : precise but slow $\longrightarrow$ Not real-time
- Neural Networks $\longrightarrow$ Real time
- ϕ-FEM $+$ Neural Networks $\longrightarrow$ Precise and real-time method ?

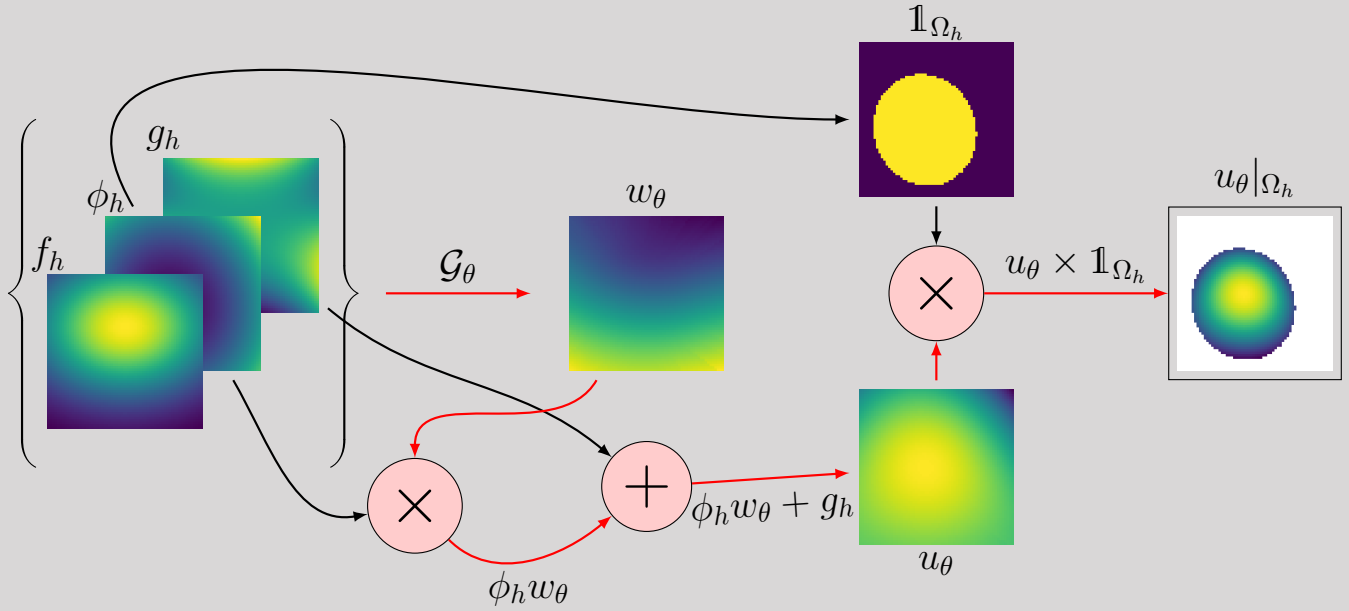


Our framework

In the context of real-time simulations, we need
quasi-instantaneous results.

- ϕ-FEM : precise but slow $\longrightarrow$ Not real-time
- Neural Networks $\longrightarrow$ Real time
- ϕ-FEM $+$ Neural Networks $\longrightarrow$ Precise and real-time method ?

The idea : construct an operator $\mathcal{G}_\theta$



How to combine ϕ-FEM and neural networks to obtain fast and precise results ?

$\longrightarrow$ the Fourier Neural Operator.

Why choose the FNO ?

- Neural operator : learns a mapping, not a solution,
- Uses FFT $\longrightarrow$ requires Cartesian grid, as ϕ-FEM does,
- According to the authors, more accurate than other ML-methods :

	NN	PCANN	DeepOnet	GNO	LNO	FNO
Darcy Flow	0.17	0.02	0.04	0.03	0.05	**0.01**
1D Burgers	0.47	0.04	0.06	0.06	0.02	**0.001**

L^2 errors of different methods **[Kovachki et al, 2023]**.

- Almost no need to change the underlying architecture when changing the governing PDE.

What is the FNO? (I)

Parametric application :

$$\mathcal{G}_\theta : \mathbb{R}^{n_x \times n_y \times 3} \xrightarrow{N} \mathbb{R}^{n_x \times n_y \times 3} \xrightarrow{P_\theta} \mathbb{R}^{n_x \times n_y \times n_d} \xrightarrow{\mathcal{H}_\theta^1} \mathbb{R}^{n_x \times n_y \times n_d} \xrightarrow{\mathcal{H}_\theta^2}$$

$$\ldots \xrightarrow{\mathcal{H}_\theta^4} \mathbb{R}^{n_x \times n_y \times n_d} \xrightarrow{Q_\theta} \mathbb{R}^{n_x \times n_y \times 1} \xrightarrow{N^{-1}} \mathbb{R}^{n_x \times n_y \times 1} .$$

- In and out dimensions : $X = (f_h, \phi_h, g_h) \to w_h$, with f_h, ϕ_h, g_h and w_h images of shape (n_x, n_y).
- N and N^{-1} : standardization and unstandardization (channel by channel),
- P_θ and Q_θ : « embedding and projection »,

$$P_\theta(X)_{ijk} = \sum_{k'=1}^{3} W_{kk'}^{P_\theta} X_{ijk'} + B_k^{P_\theta} \in \mathbb{R}^{n_d} ,$$

$\longrightarrow$ from original dimension 3 to « hidden dimension », $n_d >> 3$.

$$Q_\theta(X)_{ij} = \left[\sum_{k=1}^{n_Q} W_{1k}^{Q_\theta,2} \sigma \left(\sum_{k'=1}^{n_d} W_{kk'}^{Q_\theta,1} X_{ijk'} + B_k^{Q_\theta,1} \right) \right] + B^{Q_\theta,2} \in \mathbb{R}$$

$\longrightarrow$ from « hidden dimension » n_d to final dimension 1.

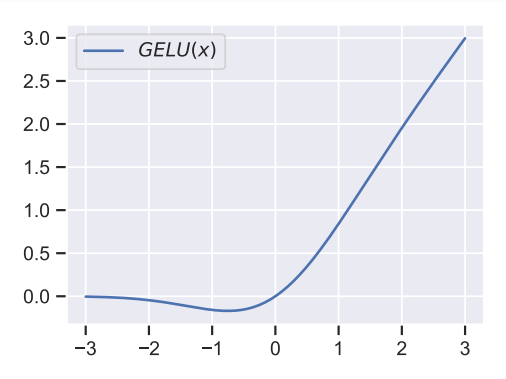
What is the FNO? (II)

Parametric application :

$$\mathcal{G}_\theta : \mathbb{R}^{n_x \times n_y \times 3} \xrightarrow{N} \mathbb{R}^{n_x \times n_y \times 3} \xrightarrow{P_\theta} \mathbb{R}^{n_x \times n_y \times n_d} \xrightarrow{\mathcal{H}_\theta^1} \mathbb{R}^{n_x \times n_y \times n_d} \xrightarrow{\mathcal{H}_\theta^2} \dots \xrightarrow{\mathcal{H}_\theta^4} \mathbb{R}^{n_x \times n_y \times n_d} \xrightarrow{Q_\theta} \mathbb{R}^{n_x \times n_y \times 1} \xrightarrow{N^{-1}} \mathbb{R}^{n_x \times n_y \times 1}.$$

Each layer $\mathcal{H}_\theta^\ell$ is defined by :

$$\mathcal{H}_\theta^\ell = \sigma\Big(\mathcal{C}_\theta^\ell(X) + \mathcal{B}_\theta^\ell(X) \Big)$$



$$\mathcal{C}_\theta^\ell(X) = \mathcal{F}^{-1}\Big(W^{\mathcal{C}_\theta^\ell}\mathcal{F}(X)\Big)$$

with $\mathcal{F}$ the real-FFT and $\mathcal{F}^{-1}$ its inverse.

The coefficients of $W^{\mathcal{C}_\theta^\ell}$ are complex trainable parameters.

$$\mathcal{B}_\theta^\ell(X)_{ijk} = \sum_{k'=1}^{n_d} W_{kk'}^{\mathcal{B}_\theta^\ell} X_{ijk'} + B_k^{\mathcal{B}_\theta^\ell}$$

The coefficients of $W^{\mathcal{B}_\theta^\ell}$ and $B^{\mathcal{B}_\theta^\ell}$ are real trainable parameters.

What is the FNO? (III)

Question :

How many trainable parameters for ϕ-FEM-FNO?

$$\mathcal{C}_\theta^l \ : \ 2 \times \overbrace{n_d}^{\text{Number of in channels}} \times \underbrace{n_d}_{\text{Number of out channels}} \times \overbrace{m_x \times m_y}^{\text{Number of Fourier modes}}$$

- $\mathcal{F}(X)$: RFFT $\rightarrow$ low frequencies stored in NW and SW corners,
- High frequencies negligible $\Longrightarrow W = 0$ outside NW and SW corners,
- Size of the corners = «number of modes»= m_x, m_y,
- Finally, for each corner : $n_d \times n_d \times m_x \times m_y$ parameters $\longrightarrow \times 2$.

Total number :

$$\overbrace{4 \times n_d}^{P_\theta \,:\, 3 \times n_d + n_d} + 4 \times \underbrace{\overbrace{(2 \times n_d^2 \times m_x \times m_y}^{\mathcal{C}_\theta^l \,:\, \text{by truncation of high frequencies}} + \overbrace{n_d^2 + n_d)}^{\mathcal{B}_\theta^l}}_{\mathcal{H}_\theta^l} + \overbrace{(n_d + 2) \times n_Q + 1}^{Q_\theta \,:\, n_d \times n_Q + n_Q + n_Q \times 1 + 1} .$$

Does not depend on the resolution of the images!

ϕ-FEM-FNO : RANDOM ELLIPSES

First test case

$$-\Delta u = f \text{ , in } \Omega, \; u = g \text{ , on } \Gamma \text{ ,}$$

- Ω is a random rotated ellipse defined using the signed distance ϕ,
- f is a random gaussian force centered in Ω with constant amplitude

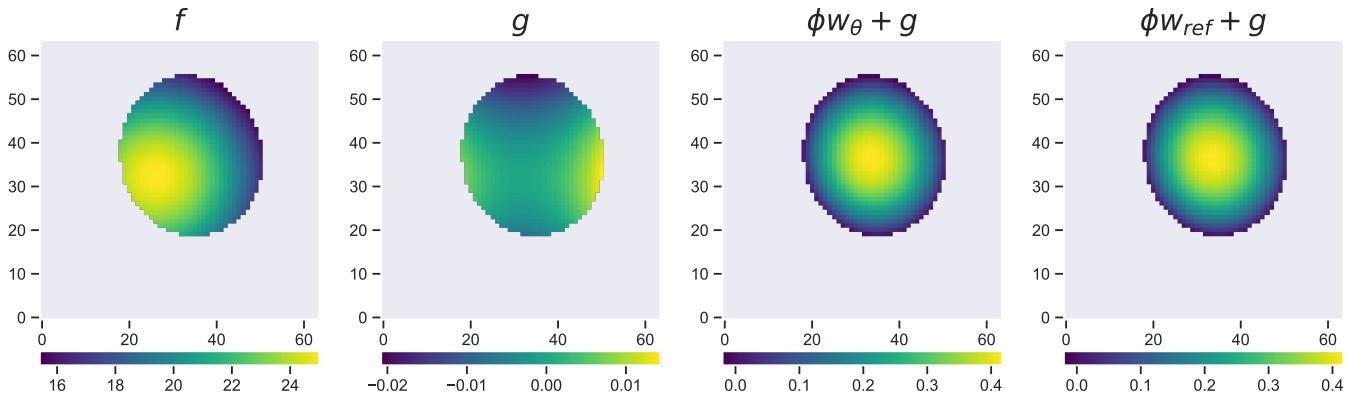
$$\Longrightarrow f \in [0, 25] \text{ ,}$$

- $g_{(\alpha,\beta)}(x, y) = \alpha \left((x - 0.5)^2 - (y - 0.5)^2 \right) \cos(\beta y \pi)$

$$\Longrightarrow g \in [-0.17, 0.15] \text{ ,}$$

- $\Longrightarrow u \in [-0.17, 0.87]$.

Dataset : 1500 training data, 300 validation data.



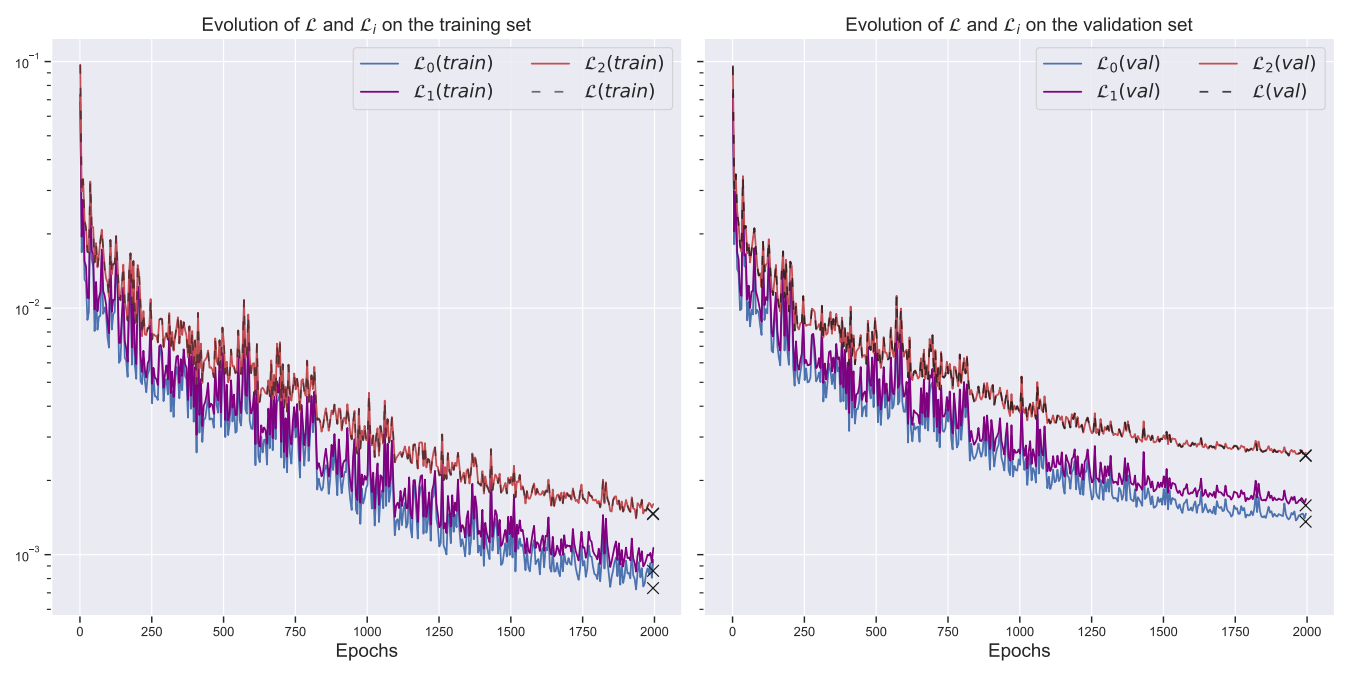
ϕ-FEM-FNO : RANDOM ELLIPSES

First test case

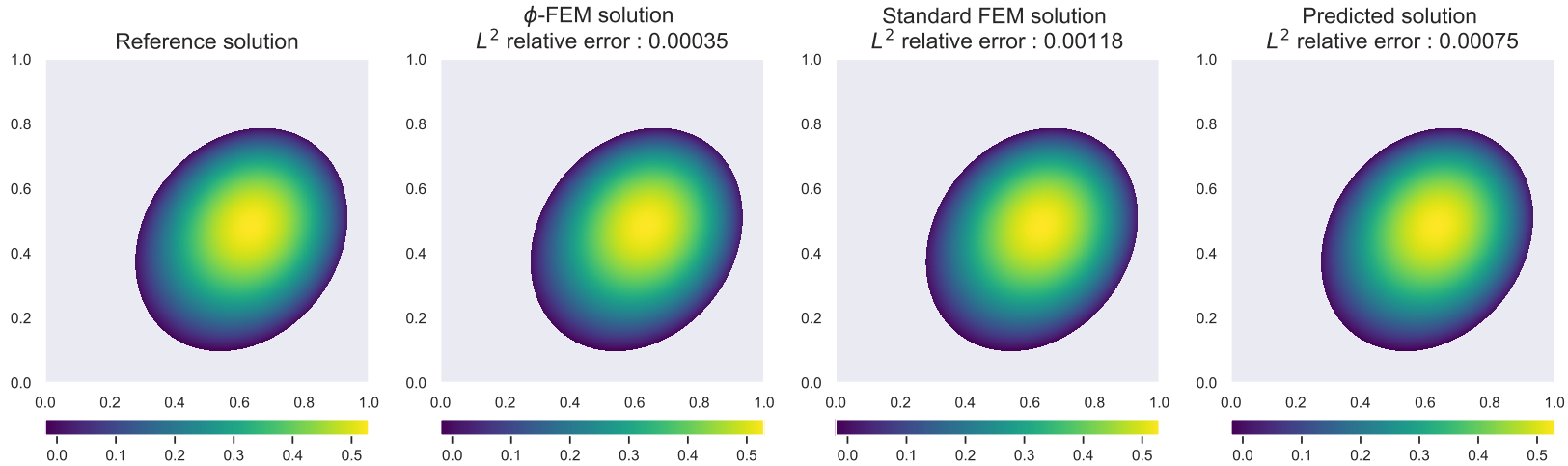
$$-\Delta u = f \text{ , in } \Omega, \; u = g \text{ , on } \Gamma \text{ ,}$$

where Ω is a random rotated ellipse.

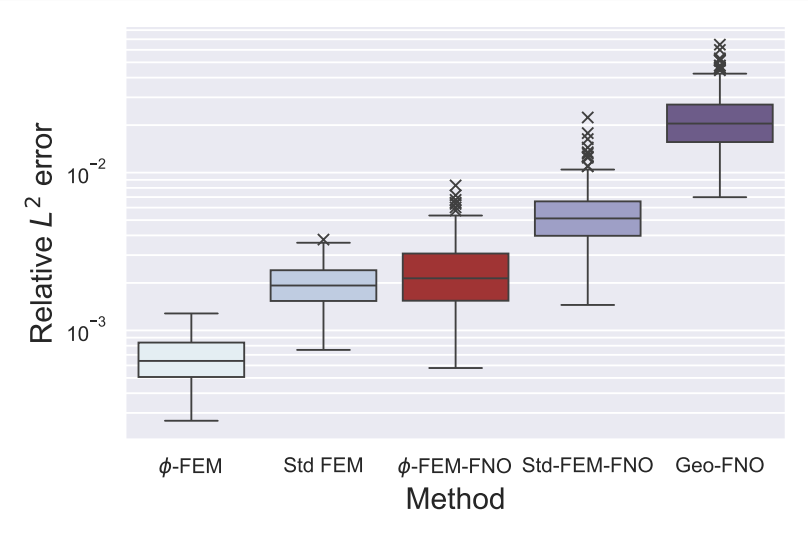
Convergence of the loss function : $\approx \| \cdot \|_{2,\Omega_h}$

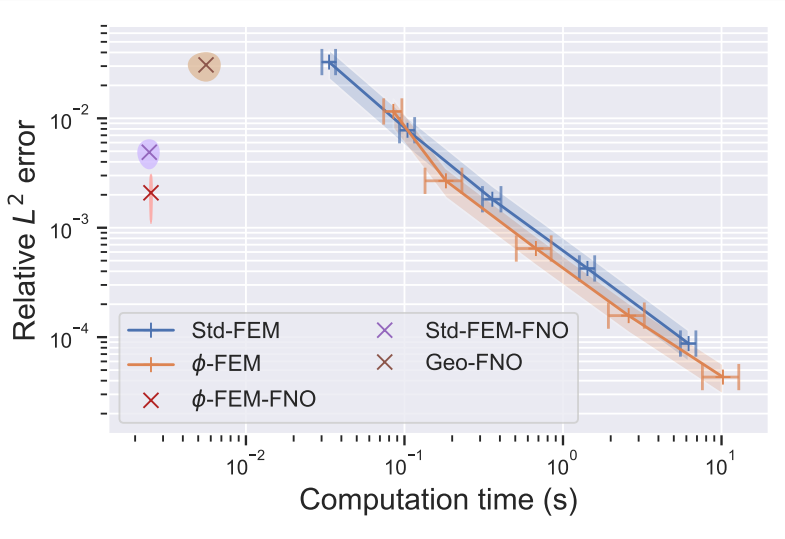


ϕ-FEM-FNO VS OTHER TECHNIQUES



Outputs of the first three methods.





Errors of the methods.

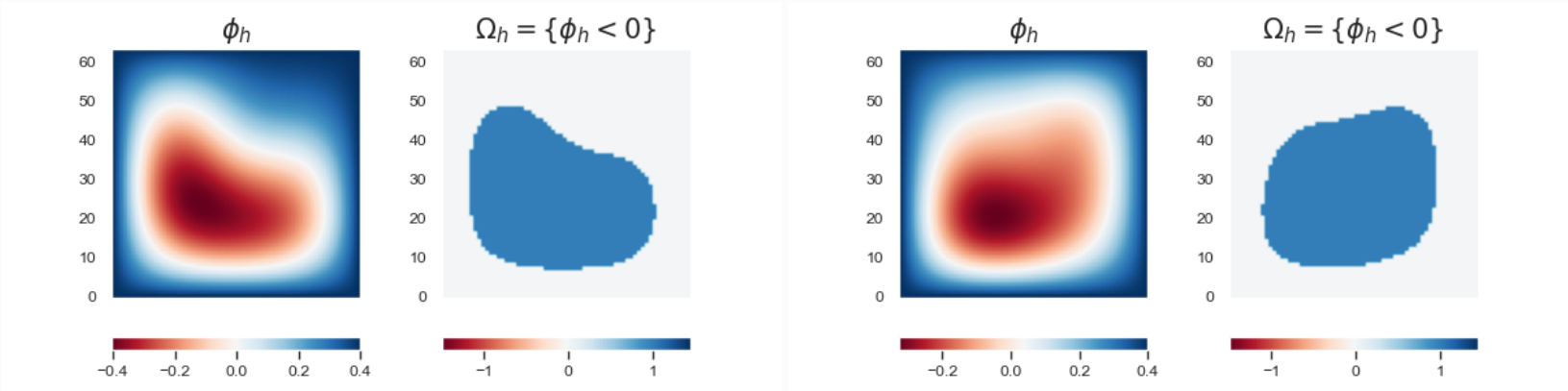
Second test case

$$-\Delta u = f \text{ , in } \Omega, \; u = g \text{ , on } \Gamma \text{ ,}$$

where Ω is defined using Fourier series,

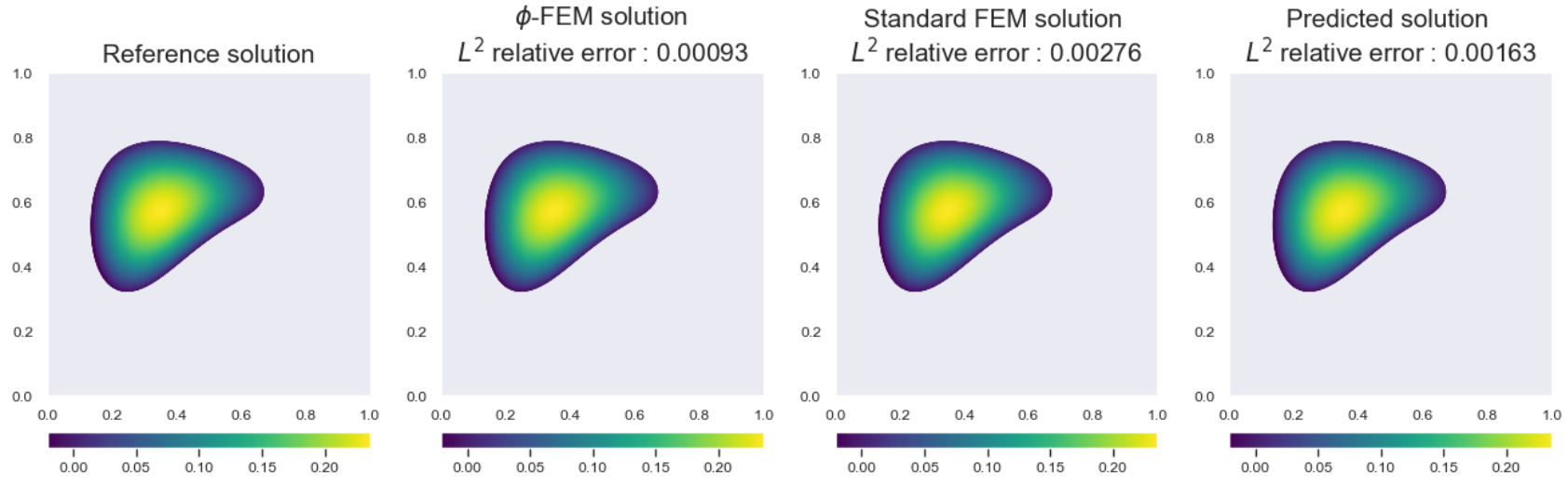
$$\phi(x, y) = 0.4 - \sum_{k=1}^{3} \sum_{l=1}^{3} \alpha_{kl} \sin(k\pi x) \sin(l\pi y),$$

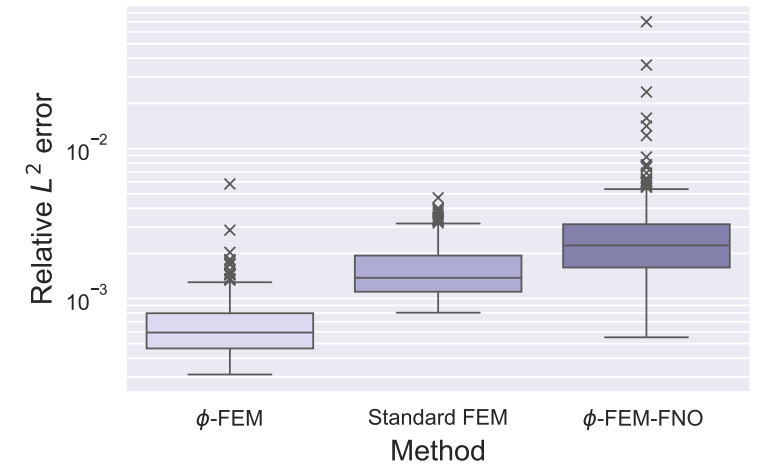
Dataset : 2200 training data, 467 validation data.

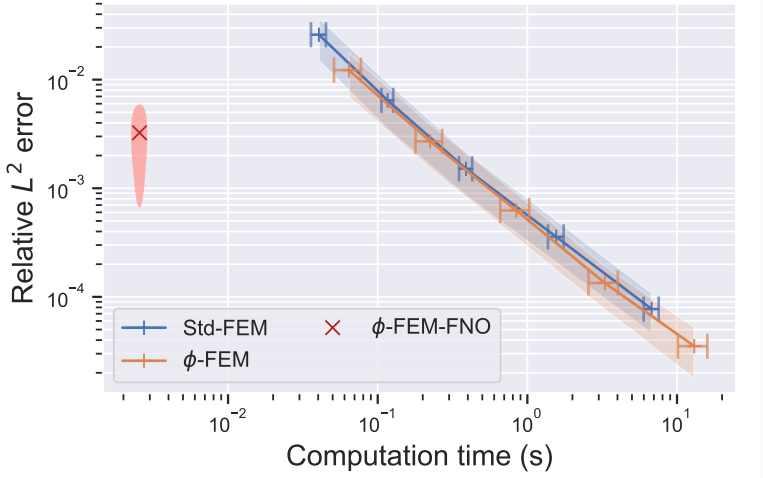


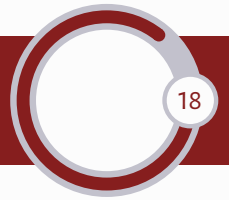
Examples of level-set functions and corresponding domains.

ϕ-FEM AND FNO : COMPLEX SHAPES









Results

- Real-time : $\approx$ 100 times faster than FEM solvers,
- $\approx$ 5 times faster and more precise than the previous work Geo-FNO.

Perspectives :

- More realistic test cases,
- Mixed Dirichlet-Neumann boundary conditions, Time-Dependant PDE's, Linear and non-linear elasticity, ...
- 3D problems,
- Validation on organ geometries.

- 05/06/2024, 10 :30 - 12 :30, MS064B $\longrightarrow$ **A. Lozinski** (Keynote Lecture) : « ϕ-FEM : a fictitious domain finite element method on geometries defined by level-sets » ;
- 06/06/2024, 10 :30 - 12 :30, MS064E $\longrightarrow$ **M. Duprez** : « A finite difference scheme with an optimal convergence for elliptic PDEs on domains defined by a level-set function » ;
- 06/06/2024, 14 :30 - 16 :30, MS061B $\longrightarrow$ **V. Lleras** : « A new immersed boundary method (ϕ-FEM) to treat Stokes equations and particulate ows ».

Thank you for your attention !

- The standardization and unstandardization operators are applied channel by channel and are given by :

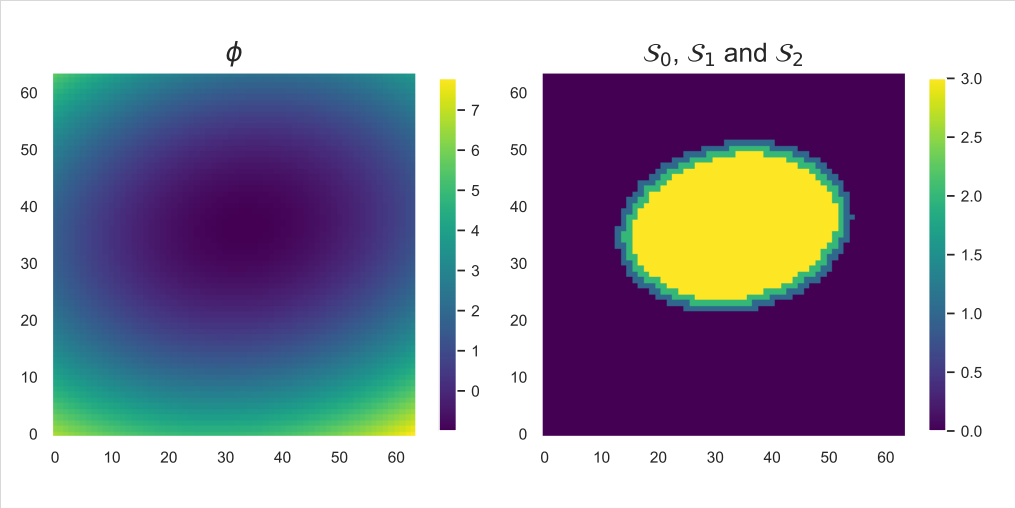
$$C^N = N(C) = \left(\frac{C - \text{mean}(C^{\text{train}})}{\text{std}(C^{\text{train}})} \right) , \qquad (N)$$

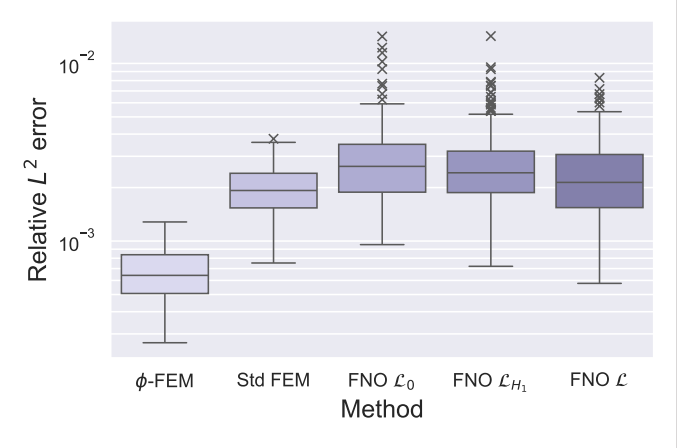
$$N^{-1}(Y) = Y \times \text{std}(Y^{\text{train}}) + \text{mean}(Y^{\text{train}}) . \qquad (N^{-1})$$

What choice for the loss $\mathcal{L}$?

H^2 norm : $\mathcal{L}^2 \approx \|\cdot\|^2_{0,\mathcal{S}_0} + |\cdot|^2_{1,\mathcal{S}_1} + |\cdot|^2_{2,\mathcal{S}_2}$

- First and second derivatives : finite differences.
- Need to reduce the computational domain :





The loss function

$$\mathcal{L}\left(U_{\text{true}}; U_\theta\right) = \frac{1}{N_{\text{data}}} \sum_{n=0}^{N_{\text{data}}} \sqrt{\frac{\sum_{i=0}^{2} \mathcal{E}_i(u^n_{\text{true}}; u^n_\theta)}{\sum_{i=0}^{2} \mathcal{N}_i(u^n_{\text{true}})}} \,,$$

where

$$\mathcal{E}_0(u^n_{\text{true}}; u^n_\theta) = \sum_{\mathcal{S}_0} \left\| u^n_{\text{true}}(i,j) - u^n_\theta(i,j) \right\|^2 ,$$

$$\mathcal{E}_1(u^n_{\text{true}}; u^n_\theta) = \sum_{\mathcal{S}_1} \left(\left\| \nabla^h_x u^n_{\text{true}}(i,j) - \nabla^h_x u^n_\theta(i,j) \right\|^2 + \left\| \nabla^h_y u^n_{\text{true}}(i,j) - \nabla^h_y u^n_\theta(i,j) \right\|^2 \right) ,$$

$$\mathcal{E}_2(u^n_{\text{true}}; u^n_\theta) = \sum_{\mathcal{S}_2} \Big(\left\| \nabla^h_x \nabla^h_x u^n_{\text{true}}(i,j) - \nabla^h_x \nabla^h_x u^n_\theta(i,j) \right\|^2$$

$$+ \left\| \nabla^h_x \nabla^h_y u^n_{\text{true}}(i,j) - \nabla^h_x \nabla^h_y u^n_\theta(i,j) \right\|^2 + \left\| \nabla^h_y \nabla^h_y u^n_{\text{true}}(i,j) - \nabla^h_y \nabla^h_y u^n_\theta(i,j) \right\|^2 \Big) ,$$

and

$$\mathcal{N}_0(u^n_{\text{true}}) = \sum_{\mathcal{S}_0} \left\| u^n_{\text{true}}(i,j) \right\|^2 ,$$

$$\mathcal{N}_1(u^n_{\text{true}}) = \sum_{\mathcal{S}_1} \left(\left\| \nabla^h_x u^n_{\text{true}}(i,j) \right\|^2 + \left\| \nabla^h_y u^n_{\text{true}}(i,j) \right\|^2 \right) ,$$

$$\mathcal{N}_2(u^n_{\text{true}}) = \sum_{\mathcal{S}_2} \left(\left\| \nabla^h_x \nabla^h_x u^n_{\text{true}}(i,j) \right\|^2 + \left\| \nabla^h_x \nabla^h_y u^n_{\text{true}}(i,j) \right\|^2 + \left\| \nabla^h_y \nabla^h_y u^n_{\text{true}}(i,j) \right\|^2 \right) .$$