

## Macaron/Tonus retreat presentation

# Mesh-based methods and physically informed learning

**Authors:**  
LECOURTIER Frédérique

**Supervisors:**  
FRANCK Emmanuel  
DUPREZ Michel  
LLERAS Vanessa

February 6-7, 2024

# Introduction

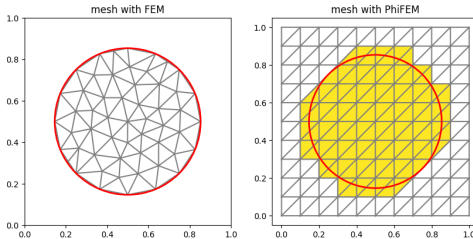
# Scientific context

**Context :** Create real-time digital twins of an organ, described by a levelset function.

→ This levelset function can easily be obtained from medical images.

**$\phi$ -FEM Method :** New fictitious domain finite element method.

- domain given by a level-set function  $\Rightarrow$  don't require a mesh fitting the boundary
- allow to work on complex geometries
- ensure geometric quality of the mesh



*Practical cases:* Real-time simulation, shape optimization...

# Objective

**Current Objective :** Develop hybrid finite element / neural network methods.

## OFFLINE :

Several Geometries



+

Several Forces

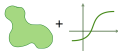


Train a PINNs



## ONLINE :

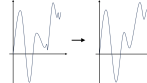
1 Geometry - 1 Force



Get PINNs prediction



Correct prediction with  $\phi$ -FEM



## Evolution :

- Geometry : 2D, simple, fixed (as circle, ellipse..)  $\rightarrow$  3D / complex / variable
- PDE : simple, static (Poisson problem)  $\rightarrow$  complex / dynamic (elasticity, hyper-elasticity)
- Neural Network : simple and defined everywhere (PINNs)  $\rightarrow$  Neural Operator

# Problem considered

## Elliptic problem with Dirichlet conditions :

Find  $u : \Omega \rightarrow \mathbb{R}^d (d = 1, 2, 3)$  such that

$$\begin{cases} L(u) = -\nabla \cdot (A(x)\nabla u(x)) + c(x)u(x) = f(x) & \text{in } \Omega, \\ u(x) = g(x) & \text{on } \partial\Omega \end{cases} \quad (1)$$

with  $A$  a definite positive coercivity condition and  $c$  a scalar. We consider  $\Delta$  the Laplace operator,  $\Omega$  a smooth bounded open set and  $\Gamma$  its boundary.

*Remark :* For simplicity, we will not consider 1st order terms.

## Weak formulation :

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

with

$$a(u, v) = \int_{\Omega} (A(x)\nabla u(x)) \cdot \nabla v(x) + c(x)u(x)v(x) dx$$
$$l(v) = \int_{\Omega} f(x)v(x) dx$$

# Numerical methods

**Objective :** Show that the philosophy behind most of the methods is the same.

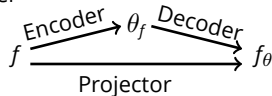
Mesh-based methods // Physically informed learning

**Numerical methods :** Discretize an infinite-dimensional problem (unknown = function) and solve it in a finite-dimensional space (unknown = vector).

- **Encoding :** we encode the problem in a finite-dimensional space
- **Approximation :** solve the problem in finite-dimensional space
- **Decoding :** bring the solution back into infinite dimensional space

Encoding	Approximation	Decoding
$f \rightarrow \theta_f$	$\theta_f \rightarrow \theta_u$	$\theta_u \rightarrow u_\theta$

**Projector :** Encoder + Decoder



# Mesh-based methods

Encoding/Decoding  
Approximation

# Mesh-based methods

Encoding/Decoding

Approximation



# Encoding/Decoding - FEMs

- **Decoding**: Linear combination of piecewise polynomial function  $\varphi_i$ .

$$u_\theta(x) = \mathcal{D}_{\theta_u}(x) = \sum_{i=1}^N (\theta_u)_i \varphi_i(x)$$

⇒ linear decoding ⇒ approximation space  $V_N$  = vectorial space

⇒ existence and uniqueness of the orthogonal projector

- **Encoding**: Optimization process.

$$\theta_f = \mathcal{E}(f) = \arg \min_{\theta \in \mathbb{R}^N} \int_{\Omega} \|f_\theta(x) - f(x)\|^2 dx$$

⇔ Orthogonal projection on vector space  $V_N = \text{Vect}\{\varphi_1, \dots, \varphi_N\}$ .

$$\theta_f = \mathcal{E}(f) = M^{-1}b(f)$$

with  $M_{ij} = \int_{\Omega} \varphi_i(x) \varphi_j(x)$  and  $b_i(f) = \int_{\Omega} \varphi_i(x) f(x)$ . Appendix 1

# Mesh-based methods

Encoding/Decoding

Approximation

# Approximation

**Idea :** Project a certain form of the equation onto the vector space  $V_N$ .

We introduce the residual of the equation defined by

$$R(v) = R_{in}(v)\mathbb{1}_{\Omega} + R_{bc}(v)\mathbb{1}_{\partial\Omega}$$

with

$$R_{in}(v) = L(v) - f \quad \text{and} \quad R_{bc}(v) = v - g$$

which respectively define the residues inside  $\Omega$  and on the boundary  $\partial\Omega$ .

**Discretization :** Degrees of freedom problem (which also has a unique solution)

$$u = \arg \min_{v \in V_N} J(v) \quad \longrightarrow \quad \theta_u = \arg \min_{\theta \in \mathbb{R}^N} J(\theta)$$

with  $J$  a functional to minimize.

**Variants :** Depends on the problem form used for projection.

**Symmetric spatial PDE**  
Problem - Energetic form  
Galerkin projection

**Any type of PDE**  
Problem - Least-square form  
Galerkin Least-square projection

# Energetic form

## Minimization Problem :

$$u_\theta(x) = \arg \min_{v \in V_N} J(v), \quad J(v) = J_{in}(v) + J_{bc}(v) \quad (2)$$

with

$$J_{in}(v) = \frac{1}{2} \int_{\Omega} L(v)v - \int_{\Omega} f v \quad \text{and} \quad J_{bc}(v) = \frac{1}{2} \int_{\partial\Omega} R_{bc}(v)^2$$

*Remark :* This form of the problem is due to the Lax-Milgram theorem as  $a$  is symmetrical.

**Minimization Problem (2)  $\Leftrightarrow$  PDE (1) :**

$$\nabla_v J(v) = R(v)$$

Appendix 2

$$\begin{array}{ccc} u_\theta \text{ sol} & \Leftrightarrow \nabla_{u_\theta} J(u_\theta) = 0 \Leftrightarrow \begin{cases} R_{in}(u_\theta) = 0 \text{ in } \Omega \\ u_\theta = g \text{ on } \partial\Omega \end{cases} \Leftrightarrow & u_\theta \text{ sol} \\ \text{of (2)} & & \text{of (1)} \end{array}$$

**Min pb**

**PDE**

# Galerkin Projection

## Discrete minimization Problem :

$$\theta_u = \arg \min_{\theta \in \mathbb{R}^N} J(\theta), \quad J(\theta) = J_{in}(\theta) = \frac{1}{2} \int_{\Omega} L(v_{\theta}) v_{\theta} - \int_{\Omega} f v_{\theta} \quad (3)$$

*Remark :* In practice, boundary conditions can be imposed in different ways. We are therefore only interested in the minimization problem in  $\Omega$ .

**Galerkin projection :** Consists in resolving

$$\langle R_{in}(u_{\theta}(x)), \varphi_i \rangle_{L^2} = 0, \quad \forall i \in \{1, \dots, N\} \quad (4)$$

**Galerkin Projection (4)  $\Leftrightarrow$  PDE (1) :**

$$\nabla_{\theta} J(\theta) = \left( \int_{\Omega} R_{in}(v_{\theta}) \varphi_i \right)_{i=1, \dots, N}$$

Appendix 3

$$u_{\theta} \text{ sol of (1)} \Leftrightarrow u_{\theta} \text{ sol of (2)} \Leftrightarrow \theta_u \text{ sol of (3)} \Leftrightarrow \nabla_{\theta} J(\theta) = 0 \Leftrightarrow u_{\theta} \text{ sol of (4)}$$

**PDE**

**Min pb**

**Discrete  
min pb**

**Galerkin  
projection**

# Least-Square form

## Minimization Problem :

$$u_\theta(x) = \arg \min_{v \in V_N} J(v), \quad J(v) = J_{in}(v) + J_{bc}(v) \quad (5)$$

with

$$J_{in}(v) = \frac{1}{2} \int_{\Omega} R_{in}(v)^2 \quad \text{and} \quad J_{bc}(v) = \frac{1}{2} \int_{\partial\Omega} R_{bc}(v)^2$$

**Minimization Problem (5)  $\Leftrightarrow$  PDE (1) :**

$$\nabla_v J(v) = L(R(v)) \mathbb{1}_{\Omega} + (v - g) \mathbb{1}_{\partial\Omega}$$

Appendix 4

$$u_\theta \text{ sol of (5)} \Leftrightarrow \nabla_{u_\theta} J(u_\theta) = 0 \Leftrightarrow \begin{cases} L(R(u_\theta)) = 0 \text{ in } \Omega \\ R(u_\theta) = 0 \text{ on } \partial\Omega \end{cases} \Leftrightarrow R(u_\theta) = 0 \Leftrightarrow u_\theta \text{ sol of (1)}$$

**Min pb**

**PDE**

# Least-Square Galerkin Projection

## Discrete minimization Problem :

$$\theta_u = \arg \min_{\theta \in \mathbb{R}^N} J(\theta), \quad J(\theta) = J_{in}(\theta) = \frac{1}{2} \int_{\Omega} (L(v_{\theta}) - f)^2 \quad (6)$$

*Remark :* In practice, boundary conditions can be imposed in different ways. We are therefore only interested in the minimization problem in  $\Omega$ .

**Least-Square Galerkin projection :** Consists in resolving

$$\langle R_{in}(u_{\theta}(x)), (\nabla_{\theta} R_{in}(u_{\theta}(x)))_i \rangle_{L^2} = 0, \quad \forall i \in \{1, \dots, N\} \quad (7)$$

**Least-Square Galerkin Projection (7)  $\Leftrightarrow$  PDE (1) :**

$$\nabla_{\theta} J(\theta) = \left( \int_{\Omega} L(R_{in}(v_{\theta})) \varphi_i \right)_{i=1, \dots, N}$$

Appendix 5

$$u_{\theta} \text{ sol of (1)} \Leftrightarrow u_{\theta} \text{ sol of (5)} \Leftrightarrow \theta_u \text{ sol of (6)} \Leftrightarrow \nabla_{\theta} J(\theta) = 0 \Leftrightarrow u_{\theta} \text{ sol of (7)}$$

**PDE**

**Min pb**

**Discrete  
min pb**

**LS Galerkin  
projection**

# Steps Decomposition - FEMs

Encoding	Approximation		Decoding
$f \rightarrow \theta_f$	$\theta_f \rightarrow \theta_u$		$\theta_u \rightarrow u_\theta$
$\theta_f = \mathcal{E}(f)$ $= M^{-1}b(f)$	Galerkin	LS Galerkin	$u_\theta(x) = \mathcal{D}_\theta(x)$ $= \sum_{i=1}^N (\theta_u)_i \varphi_i$
	$\langle R(u_\theta), \varphi_i \rangle_{L^2} = 0$	$\langle R(u_\theta), (\nabla_\theta R(u_\theta))_i \rangle_{L^2} = 0$	
	$A\theta_u = B$		

**Example :** Galerkin projection.

For  $i \in \{1, \dots, N\}$ ,

$$\langle R(u_\theta), \varphi_i \rangle_{L^2} = 0$$

$$\iff \int_{\Omega} L(u_\theta) \varphi_i = \int_{\Omega} f \varphi_i$$

$$\iff \sum_{j=1}^N (\theta_u)_j \int_{\Omega} \varphi_i L(\varphi_j) = \int_{\Omega} f \varphi_i$$

$$A\theta_u = B \text{ with}$$

$$A_{i,j} = \int_{\Omega} \varphi_i L(\varphi_j) \quad , \quad B_i = \int_{\Omega} f \varphi_i$$



# Physically Informed Learning

Encoding/Decoding  
Approximation

# Physically Informed Learning

Encoding/Decoding

Approximation

# Encoding/Decoding - NNs

- **Decoding**: Implicit neural representation.

$$u_{\theta}(x) = \mathcal{D}_{\theta_u}(x) = u_{NN}(x)$$

with  $u_{NN}$  a neural network (for example a MLP).

⇒ non-linear decoding ⇒ approximation space  $\mathcal{M}_N$  = finite-dimensional variety

⇒ there is no unique projector

- **Encoding**: Optimization process.

$$\theta_f = \mathcal{E}(f) = \arg \min_{\theta \in \mathbb{R}^N} \int_{\Omega} \|f_{\theta}(x) - f(x)\|^2 dx$$

# Neural Network Decoder

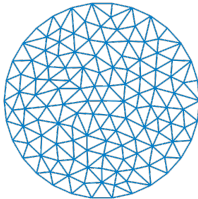
## Advantages of a non-linear decoder :

- We gain in the richness of the approximation
- We can hope to significantly reduce the number of degrees of freedom
- This avoids the need to use meshes.

polynomial models

⇒ local precision

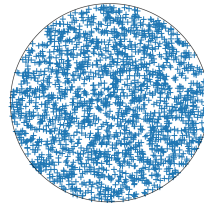
⇒ use meshes



NN models

⇒ global precision

⇒ no need to use meshes



# Physically Informed Learning

Encoding/Decoding  
Approximation

# Approximation

**Idea :** Project a certain form of the equation onto the variety  $\mathcal{M}_N$ .

**Discretization :** Degrees of freedom problem (no mesh).

$$u = \arg \min_{v \in \mathcal{M}_N} J(v) \longrightarrow \theta_u = \arg \min_{\theta \in \mathbb{R}^N} J(\theta)$$

with  $J$  a functional to minimize.

**Variants :** Depends on the problem form used for projection.

## Symmetric spatial PDE

Problem - Energetic form

Deep-Ritz

(Galerkin projection)

## Any type of PDE

Problem - Least-square form

Standard PINNs

(Galerkin Least-square projection)

# Deep-Ritz

**Discrete minimization Problem :** Considering the energetic form of our PDE, our discrete problem is

$$\theta_u = \arg \min_{\theta \in \mathbb{R}^N} J_{in}(\theta) + J_{bc}(\theta) \quad (8)$$

with

$$J_{in}(\theta) = \frac{1}{2} \int_{\Omega} L(v_{\theta}) v_{\theta} - \int_{\Omega} f v_{\theta} \quad \text{and} \quad J_{bc}(\theta) = \frac{1}{2} \int_{\partial\Omega} (v_{\theta} - g)^2$$

**Monte-Carlo method :** Discretize the cost function by random process.

- $(x_1, \dots, x_n)$  randomly drawn according to  $\mu(x)$  defined on  $\Omega$

$$J_{in}(\theta) = \frac{1}{2n} \sum_{i=1}^n L(v_{\theta}(x_i)) v_{\theta}(x_i) - \frac{1}{n} \sum_{i=1}^n f(x_i) v_{\theta}(x_i)$$

- $(y_1, \dots, y_{n_b})$  randomly drawn according to  $\mu_b(x)$  defined on  $\partial\Omega$

$$J_{bc}(\theta) = \frac{1}{2n_b} \sum_{i=1}^{n_b} (v_{\theta}(y_i) - g(y_i))^2$$

*Remark :* → Two different random generation processes (to have enough boundary points)

→ Weights in front of the cost functions still need to be determined

# Standard PINNs

**Discrete minimization Problem :** Considering the least-square form of our PDE, our discrete problem is

$$\theta_u = \arg \min_{\theta \in \mathbb{R}^N} J_{in}(\theta) + J_{bc}(\theta) \quad (9)$$

with

$$J_{in}(\theta) = \frac{1}{2} \int_{\Omega} (L(v_{\theta}) - f)^2 \quad \text{and} \quad J_{bc}(\theta) = \frac{1}{2} \int_{\partial\Omega} (v_{\theta} - g)^2$$

**Monte-Carlo method :** Discretize the cost function by random process.

- $(x_1, \dots, x_n)$  randomly drawn according to  $\mu(x)$  defined on  $\Omega$

$$J_{in}(\theta) = \frac{1}{2n} \sum_{i=1}^n (L(v_{\theta}(x_i)) - f(x_i))^2$$

- $(y_1, \dots, y_{n_b})$  randomly drawn according to  $\mu_b(x)$  defined on  $\partial\Omega$

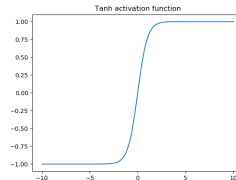
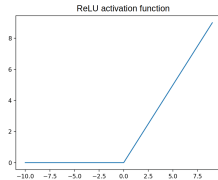
$$J_{bc}(\theta) = \frac{1}{2n_b} \sum_{i=1}^{n_b} (v_{\theta}(y_i) - g(y_i))^2$$

*Remark :* Same.



# In practice...

- ➔ Stochastic gradient descent method (by mini-batch) - ADAM method Appendix 6
- ➔ Use regular model, derivable several times (and automatic differentiation)
- ➔ Activation functions regular enough to be derived 2 times (due to the Laplacian)
  - ⇒ Tangent Hyperbolic rather than ReLU
- (or adaptive methods where we parameterize the activation functions)



## To go further :

- ➔ Standard PINNs : possibility of adding a  $J_{data}$  cost function
  - to approximate already known solutions
- ➔ Impose boundary conditions using a LevelSet function

# Steps Decomposition - NNs

Encoding	Approximation	Decoding
$f \rightarrow \theta_f$	$\theta_f \rightarrow \theta_u$	$\theta_u \rightarrow u_\theta$

Mesh-based Methods			
$\theta_f = \mathcal{E}(f)$ $= M^{-1}b(f)$	Galerkin	LS Galerkin	$u_\theta(x) = \mathcal{D}_\theta(x)$ $= \sum_{i=1}^N (\theta_u)_i \varphi_i$
	$\langle R(u_\theta), \varphi_i \rangle = 0$	$\langle R(u_\theta), (\nabla_\theta R(u_\theta))_i \rangle = 0$	
	$A\theta_u = B$		

Physically informed learning			
$\theta_f = \min_{\theta \in \mathbb{R}^N} \int_{\Omega}   f_\theta - f  ^2$	Deep-Ritz	Standard PINNs	$u_\theta(x) = u_{NN}(x)$
	Energetic Form	LS Form	
	$\theta_u = \arg \min_{\theta \in \mathbb{R}^N} J(\theta)$		

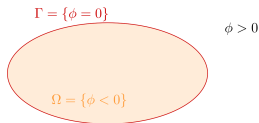
# Our hybrid method

# $\phi$ -FEM Method

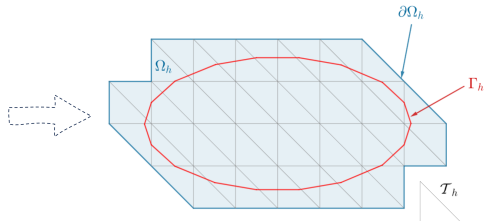
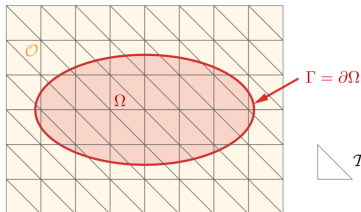
## Main ideas :

### Appendix 7

- Domain defined by a LevelSet Function  $\phi$ .



- Mesh of a fictitious domain containing  $\Omega$ .



- We are looking for  $w$  such that  $u = \phi w + g$ . Thus, the decoder is written as

$$u_\theta(x) = \mathcal{D}_{\theta_w}(x) = \phi(x) \sum_{i=1}^N (\theta_w)_i \varphi_i + g(x)$$

# Impose exact BC in PINNs

Considering the least squares form of our PDE, we impose the exact boundary conditions by writing our solution as

$$u_\theta = \phi w_\theta + g$$

where  $w_\theta$  is our decoder (defined by a neural network such as an MLP). We then consider the same minimization problem by removing the cost function associated with the boundary

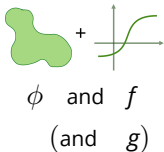
$$\theta_u = \arg \min_{\theta \in \mathbb{R}^N} J_{in}(\theta) + \cancel{J_{bc}(\theta)}$$

with

$$J_{in}(\theta) = \frac{1}{2} \int_{\Omega} (L(\phi w_\theta + g) - f)^2 \quad \text{and} \quad \cancel{J_{bc}(\theta) = \frac{1}{2} \int_{\partial\Omega} (v_\theta - g)^2}$$

# Correct PINNs prediction with $\phi$ FEM

1 Geometry - 1 Force



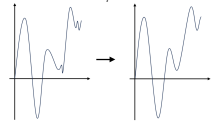
Get PINNs  
prediction



$$u_{NN} = \phi w_{NN} + g$$



Correct prediction  
with  $\phi$ -FEM



$$u_{NN} \rightarrow \tilde{u} = u_{NN} + \phi C$$

**Correct by adding :** Considering  $u_{NN}$  as the prediction of our PINNs (trained to learn the solution of the elliptic problem), the correction problem consists in writing the solution as

$$\tilde{u} = u_{NN} + \boxed{\tilde{C}}_{\approx 0}$$

and searching  $\tilde{C} : \Omega \rightarrow \mathbb{R}^d$  such that

$$\begin{cases} L(\tilde{C}) = \tilde{f}, & \text{in } \Omega, \\ \tilde{C} = 0, & \text{on } \Gamma, \end{cases}$$

with  $\tilde{f} = f - L(u_{NN})$  and  $\tilde{C} = \phi C$  for the  $\phi$ -FEM method.

# Conclusion

# Conclusion - What has been seen

- "Physical Informed Learning" methods are simply an extension of classic numerical methods such as FEM, where the decoder belongs to a variety (whose properties are different from those of vector spaces).
- These approaches have real advantages in high dimensions, particularly in the context of parametric PDEs.
- Moreover, as they are mesh-free methods, they have a major advantage in the context of complex geometries.



# Conclusion - Our hybrid approach

## Interest of our approach:

- It combines
  - ➔ Speed of neural networks in predicting a solution
  - ➔ Precision of FEM methods to correct and certify the prediction of the neural network (which can be completely wrong, on an unknown dataset for example)
- In the context of complex geometry (or in application domains such as real-time or shape optimisation), like NNs,  $\phi$ -FEM makes it possible to avoid mesh (re-)generation.

## Current results:

- Encouraging results on simple geometries [Appendix 8](#)
- Difficulties on complex geometries - Importance of the regularity of the LevelSet function
  - ➔ Next step: learning levelset functions (Eikonal equation)

Thank you !

# Bibliography

- [1] Erik Burman. Ghost penalty. *Comptes Rendus. Mathématique*.
- [2] Erik Burman, Susanne Claus, Peter Hansbo, Mats G. Larson, and André Massing. CutFEM: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*.
- [3] Stéphane Cotin, Michel Duprez, Vanessa Lleras, Alexei Lozinski, and Killian Vuillemot.  $\phi$ -FEM: an efficient simulation tool using simple meshes for problems in structure mechanics and heat transfer.
- [4] Michel Duprez, Vanessa Lleras, and Alexei Lozinski. *A new  $\phi$ -FEM approach for problems with natural boundary conditions*. 2020.
- [5] Michel Duprez, Vanessa Lleras, and Alexei Lozinski.  $\phi$ -FEM: an optimally convergent and easily implementable immersed boundary method for particulate flows and Stokes equations. *ESAIM: Mathematical Modelling and Numerical Analysis*.
- [6] Michel Duprez and Alexei Lozinski.  $\phi$ -FEM: A Finite Element Method on Domains Defined by Level-Sets. *SIAM Journal on Numerical Analysis*.
- [7] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. *Implicit Neural Representations with Periodic Activation Functions*. 2020.
- [8] N. Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*.

# Mesh-based methods

# Appendix 1 : Encoding - FEMs

We want to project  $f$  onto the vector subspace  $V_N$  so that  $f_\theta = p_{V_N}(f)$   
 then  $\forall i \in \{1, \dots, N\}$ , we have

$$\begin{aligned} \langle f_\theta - f, \varphi_i \rangle &= 0 \\ \iff \langle f_\theta, \varphi_i \rangle &= \langle f, \varphi_i \rangle \\ \iff \sum_{j=1}^N (\theta_f)_j \langle \varphi_j, \varphi_i \rangle &= \langle f, \varphi_i \rangle \\ \iff M \theta_f &= b(f) \\ \iff \theta_f &= M^{-1} b(f) \end{aligned}$$

with

$$\begin{aligned} M_{ij} &= \langle \varphi_i, \varphi_j \rangle = \int_{\Omega} \varphi_i(x) \varphi_j(x) dx \\ b_i(f) &= \langle f, \varphi_i \rangle = \int_{\Omega} f(x) \varphi_i(x) dx \end{aligned}$$

## Appendix 2 : Energetic form I

Let's compute the gradient of  $J$  with respect to  $v$  with

$$J(v) = J_{in}(v) + J_{bc}(v) = \left( \frac{1}{2} \int_{\Omega} L(v)v - \int_{\Omega} f v \right) + \left( \frac{1}{2} \int_{\partial\Omega} R_{bc}(v)^2 \right)$$

- First, let's calculate the differential of  $J_{in}$  with respect to  $v$ .

$$J_{in}(v + \epsilon h) = \frac{1}{2} \int_{\Omega} (A \nabla(v + \epsilon h)) \cdot \nabla(v + \epsilon h) + c(v + \epsilon h)^2 - \int_{\Omega} f(v + \epsilon h)$$

By bilinearity of the scalar product and by symmetry of  $A$ , we finally obtain

$$\mathcal{D}J_{in}(v) \cdot h = \lim_{\epsilon \rightarrow 0} \frac{J_{in}(v + \epsilon h) - J_{in}(v)}{\epsilon} = \int_{\Omega} (-\nabla \cdot (A \nabla v) + cv - f)h$$

And thus

$$\nabla_v J_{in}(v) = L(v) - f = R_{in}(v)$$

## Appendix 2 : Energetic form II

- In the same way, we can compute the differential of  $J_{bc}$  with respect to  $v$ .

$$J_{bc}(v + \epsilon h) = \frac{1}{2} \int_{\partial\Omega} v^2 + 2\epsilon v h + \epsilon^2 h^2 - 2v g - 2\epsilon h g + g^2$$

Then

$$\mathcal{D}J_{bc}(v) \cdot h = \lim_{\epsilon \rightarrow 0} \frac{J_{bc}(v + \epsilon h) - J_{bc}(v)}{\epsilon} = \int_{\partial\Omega} (v - g)h$$

And thus

$$\nabla_v J_{bc}(v) = (v - g) = R_{bc}(v)$$

Finally

$$\nabla_v J(v) = \nabla_v J_i(v) + \nabla_v J_{bc}(v) = R(v)$$

## Appendix 3 : Galerkin Projection

Let's compute the gradient of  $J$  with respect to  $\theta$  with

$$J(\theta) = J_{in}(\theta) = \frac{1}{2} \int_{\Omega} L(u_{\theta}) v_{\theta} - \int_{\Omega} f v_{\theta}$$

First, we define

$$v_{\theta} = \sum_{i=1}^N \theta_i \varphi_i = \theta \cdot \varphi \quad \text{and} \quad v_{\theta+\epsilon h} = (\theta + \epsilon h) \cdot \varphi = v_{\theta} + \epsilon v_h$$

Then since  $A$  is symmetric

$$\mathcal{D}J(\theta) \cdot h = \int_{\Omega} R(v_{\theta}) v_h = \sum_{i=1}^N h_i \int_{\Omega} R(v_{\theta}) \varphi_i$$

Finally

$$\nabla_{\theta} J(\theta) = \left( \int_{\Omega} R(v_{\theta}) \varphi_i \right)_{i=1, \dots, N}$$



## Appendix 4 : Least-Square form I

Let's compute the gradient of  $J$  with respect to  $v$  with

$$J(v) = J_{in}(v) + J_{bc}(v) = \left( \frac{1}{2} \int_{\Omega} R_{in}(v)^2 \right) + \left( \frac{1}{2} \int_{\partial\Omega} R_{bc}(v)^2 \right)$$

- First, let's calculate the differential of  $J_{in}$  with respect to  $v$ .

$$\begin{aligned} \mathcal{D}J_{in}(v) \cdot h &= \langle \nabla \cdot (A \nabla h), \nabla \cdot (A \nabla v) - cv + f \rangle + \langle ch, -\nabla \cdot (A \nabla v) + cv - f \rangle \\ &= -\langle \nabla \cdot (A \nabla h), R_{in}(v) \rangle + \langle ch, R_{in}(v) \rangle \\ &= \langle -\nabla \cdot (A \nabla R_{in}(v)) + c R_{in}(v), h \rangle \\ &= \langle L(R_{in}(v)), h \rangle \end{aligned}$$

And thus

$$\nabla_v J_{in}(v) = L(R_{in}(v))$$

## Appendix 4 : Least-Square form II

- In the same way, we can compute the differential of  $J_{bc}$  with respect to  $v$ .

$$J_{bc}(v + \epsilon h) = \frac{1}{2} \int_{\partial\Omega} v^2 + 2\epsilon v h + \epsilon^2 h^2 - 2vg - 2\epsilon h g + g^2$$

Then

$$\mathcal{D}J_{bc}(v) \cdot h = \lim_{\epsilon \rightarrow 0} \frac{J_{bc}(v + \epsilon h) - J_{bc}(v)}{\epsilon} = \int_{\partial\Omega} (v - g)h$$

And thus

$$\nabla_v J_{bc}(v) = (v - g) = R_{bc}(v)$$

Finally

$$\nabla_v J(v) = L(R(v))\mathbb{1}_{\Omega} + (v - g)\mathbb{1}_{\partial\Omega}$$

# Appendix 5 : LS Galerkin Projection

Let's compute the gradient of  $J$  with respect to  $\theta$  with

$$J(\theta) = J_{in}(\theta) = \frac{1}{2} \int_{\Omega} (L(u_{\theta}) - f)^2$$

First, we define

$$v_{\theta} = \sum_{i=1}^N \theta_i \varphi_i = \theta \cdot \varphi \quad \text{and} \quad v_{\theta+\epsilon h} = (\theta + \epsilon h) \cdot \varphi = v_{\theta} + \epsilon v_h$$

Then since  $A$  is symmetric

$$\mathcal{D}J(\theta) \cdot h = \int_{\Omega} L(R(v_{\theta})) v_h = \sum_{i=1}^N h_i \int_{\Omega} L(R(v_{\theta})) \varphi_i$$

Finally

$$\nabla_{\theta} J(\theta) = \left( \int_{\Omega} L(R(v_{\theta})) \varphi_i \right)_{i=1, \dots, N}$$

# Physically Informed Learning

# Appendix 6 : ADAM Method

Adam = Adaptive Moment Estimation" - combine les idées du moment et de RMSProp.

$$\begin{aligned}
 1 : \quad m &\leftarrow \frac{\beta_1 m + (1 - \beta_1) \nabla f_x}{1 - \beta_1^T} \\
 2 : \quad s &\leftarrow \frac{\beta_2 s + (1 - \beta_2) \nabla^2 f_x}{1 - \beta_2^T} \\
 3 : \quad x &\leftarrow x - \ell \frac{m}{\sqrt{s + \epsilon}}
 \end{aligned}$$

with

- $T$  the number of iteration (starting at 1)
- $\epsilon$  a smoothing parameter
- $\beta_i \in ]0, 1[$  which converge quickly to 0.

# Our hybrid method

Appendix 7 :  $\phi$ -FEM Method

Appendix 8 : Results

# Our hybrid method

Appendix 7 :  $\phi$ -FEM Method

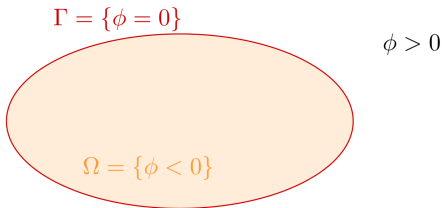
Appendix 8 : Results

## Appendix 7 : Problem

Let  $u = \phi w + g$  such that

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

where  $\phi$  is the level-set function and  $\Omega$  and  $\Gamma$  are given by :

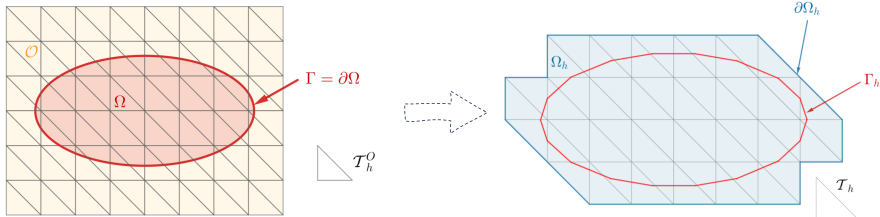


The level-set function  $\phi$  is supposed to be known on  $\mathbb{R}^d$  and sufficiently smooth. For instance, the signed distance to  $\Gamma$  is a good candidate.

*Remark* : Thanks to  $\phi$  and  $g$ , the conditions on the boundary are respected.



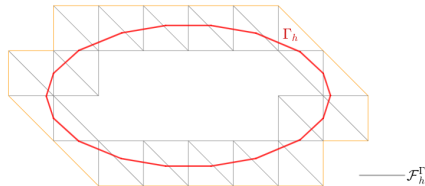
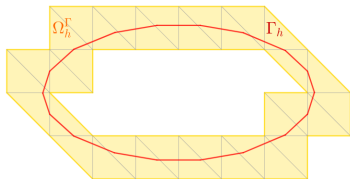
# Appendix 7 : Fictitious domain



- $\phi_h$  : approximation of  $\phi$
- $\Gamma_h = \{\phi_h = 0\}$  : approximate boundary of  $\Gamma$
- $\Omega_h$  : computational mesh
- $\partial\Omega_h$  : boundary of  $\Omega_h$  ( $\partial\Omega_h \neq \Gamma_h$ )

Remark :  $n_{vert}$  will denote the number of vertices in each direction for  $\mathcal{O}$

# Appendix 7 : Facets and Cells sets



- $\mathcal{T}_h^\Gamma$  : mesh elements cut by  $\Gamma_h$
- $\mathcal{F}_h^\Gamma$  : collects the interior facets of  $\mathcal{T}_h^\Gamma$   
(either cut by  $\Gamma_h$  or belonging to a cut mesh element)

# Appendix 7 : Poisson problem

**Approach Problem :** Find  $w_h \in V_h^{(k)}$  such that

$$a_h(w_h, v_h) = l_h(v_h) \quad \forall v_h \in V_h^{(k)}$$

where

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

$$l_h(v) = \int_{\Omega_h} f \phi_h v + \boxed{G_h^{hs}(v)} \quad \text{Stabilization terms}$$

and

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

For the non homogeneous case, we replace

$$u = \phi w \quad \rightarrow \quad u = \phi w + g$$

by supposing that  $g$  is currently given over the entire  $\Omega_h$ .

# Appendix 7 : Stabilization terms

Independent parameter of  $h$       Jump on the interface  $E$

$$G_h(w, v) = \underbrace{\sigma h \sum_{E \in \mathcal{F}_h^\Gamma} \int_E \left[ \frac{\partial}{\partial n}(\phi_h w) \right] \left[ \frac{\partial}{\partial n}(\phi_h v) \right]}_{\text{1st order term}} + \underbrace{\sigma h^2 \sum_{T \in \mathcal{T}_h^\Gamma} \int_T \Delta(\phi_h w) \Delta(\phi_h v)}_{\text{2nd order term}}$$

$$G_h^{rhs}(v) = \underbrace{-\sigma h^2 \sum_{T \in \mathcal{T}_h^\Gamma} \int_T f \Delta(\phi_h v)}_{\text{2nd order term}} - \sigma h^2 \sum_{T \in \mathcal{T}_h^\Gamma} \int_T (\Delta(\phi_h w) + f) \Delta(\phi_h v)$$

1st term : ensure continuity of the solution by penalizing gradient jumps.

→ Ghost penalty [Burman, 2010]

2nd term : require the solution to verify the strong form on  $\Omega_h^\Gamma$ .

**Purpose :**

- ➔ reduce the errors created by the "fictitious" boundary
- ➔ ensure the correct condition number of the finite element matrix
- ➔ restore the coercivity of the bilinear scheme

# Our hybrid method

Appendix 7 :  $\phi$ -FEM Method

Appendix 8 : Results

# Appendix 8 : Problem considered

**PDE :** Poisson problem with Homogeneous Dirichlet conditions

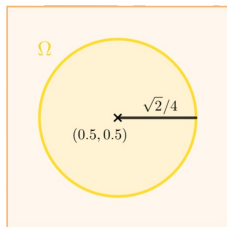
Find  $u : \Omega \rightarrow \mathbb{R}^d (d = 1, 2, 3)$  such that

$$\begin{cases} -\Delta u = f, & \text{in } \Omega, \\ u = 0, & \text{on } \Gamma, \end{cases}$$

with  $\Delta$  the Laplace operator,  $\Omega$  a smooth bounded open set and  $\Gamma$  its boundary.

**Geometry :** Circle - center=(0.5, 0.5) , radius= $\sqrt{2}/4$

$$\mathcal{O} = [0, 1]^2$$



→ **Level-set function :**

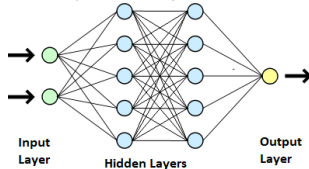
$$\phi(x, y) = -1/8 + (x - 1/2)^2 + (y - 1/2)^2$$

→ **Exact solution :**

$$u_{ex}(x, y) = \phi(x, y) \sin(x) \exp(y)$$

# Appendix 8 : Networks

**PINNs** : Multi-Layer Perceptron (MLP, Fully connected) with a physical loss



$$loss = mse(\Delta(\phi(x_i, y_i)w_{\theta, i}) + f_i)$$

$$\begin{aligned} \text{inputs} &= \{(x_i, y_i)\} \\ \text{outputs} &= \{u_i\} \\ &\quad i = 1, \dots, n_{pts} \\ u_i &= \phi(x_i, y_i)w_{\theta, i} \end{aligned}$$

- n\_layers=4
- n\_neurons=20 (in each layer)
- n\_epochs=10000
- n\_pts=2000 (randomly drawn in the square  $[0, 1]^2$ )

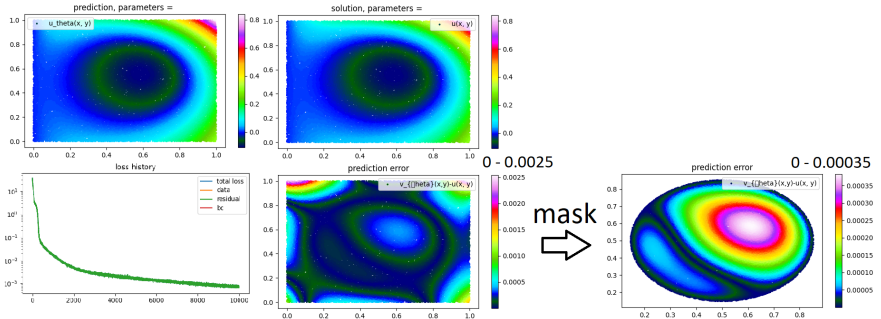
**Some important points :**

- Need  $u_{NN} \in \mathbb{P}^k$  of high degree (PINNs Ok)
- Need the derivatives to be well learn (PINNs Ok)
- For the correction : Need a correct solution on  $\Omega_h$ , not on  $\Omega$  (training on the square for the moment).

with  $(x_i, y_i) \in \mathcal{O}$ .

*Remark* : We impose exact boundary conditions.

## Appendix 8 : Training



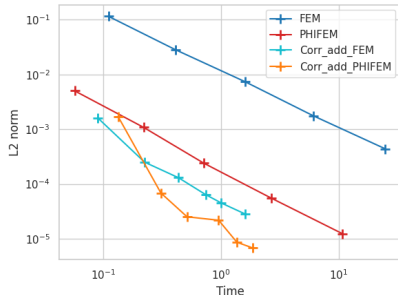
⚠ We consider a single problem ( $f$  fixed) on a single geometry ( $\phi$  fixed).

$$||u_{ex} - u_{\theta}||_{L^2(\Omega)}^{(rel)} \approx 2.81e-3$$



# Appendix 8 : Correction

$$u_\theta \in \mathbb{P}^{10} \rightarrow \tilde{u} \in \mathbb{P}^1$$



FEM /  $\phi$ -FEM :  $n_{vert} \in \{8, 16, 32, 64, 128\}$

Corr :  $n_{vert} \in \{5, 10, 15, 20, 25, 30\}$

*Remark* : The stabilisation parameter  $\sigma$  of the  $\phi$ -FEM method has a major impact on the error obtained.

Calculation time (to reach an error of  $1e-4$ )

	mesh	u_PINNs	assemble	solve	TOTAL
FEM	0,08832		29,55516	0,07272	29,71621
PhiFEM	0,33222		1,86924	0,00391	2,20537
Corr_add_FEM	0,00183	0,11187	0,46195	0,00061	0,57626
Corr_add_PhiFEM	0,03213	0,05351	0,22006	0,00040	0,30609

*Remark* : Problem with assemble and solve time

+ mesh time for  $\phi$ -FEM would be parallelized

- **mesh** - FEM : construct the mesh  
( $\phi$ -FEM : construct cell/facet sets)
- **u\_PINNs** - get  $u_\theta$  in  $\mathbb{P}^{10}$  freedom degrees
- **assemble** - assemble the FE matrix
- **solve** - resolve the linear system