Macaron/Tonus retreat presentation

Mesh-based methods and physically informed learning

Authors:
Frédérique LECOURTIER

Supervisors:
Emmanuel FRANCK
Michel DUPREZ
Vanessa LLERAS

February 6-7, 2024

MIMESIS

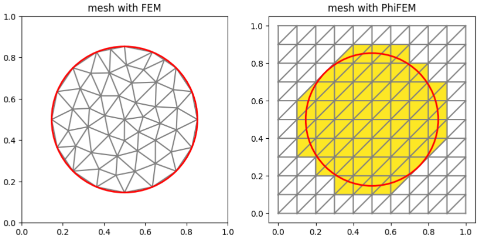
Introduction

Scientific context

Context : Create real-time digital twins of an organ, described by a levelset function.
$\rightarrow$ This levelset function can easily be obtained from medical images.

ϕ**-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...

Neural Network : Obtain a solution quickly.

MIMESIS

Problem considered

Elliptic problem with Dirichlet conditions :

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

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

with A a definite positive coercivity condition and c a scalar. We consider Δ the Laplace operator, Ω a smooth bounded open set and Γ its boundary.

Weak formulation :

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

with

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

$$l(v) = \int_\Omega f(x)v(x)\, dx$$

MIMESIS

Aim of the talk

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

$$\begin{tikzcd} & \theta_f \arrow[dr, "\text{Decoder}"] & \\ f \arrow[ur, "\text{Encoder}"] \arrow[rr, "\text{Projector}"'] & & f_\theta \end{tikzcd}$$

MIMESIS

Mesh-based methods (FEM)

Encoding/Decoding

Approximation

Mesh-based methods (FEM)

Encoding/Decoding

Approximation

Encoding/Decoding - FEMs

- **Decoding :** Linear combination of piecewise polynomial function φ_i.

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

$\Rightarrow$ linear decoding $\Rightarrow$ approximation space V_N = vectorial space
$\Rightarrow$ existence and uniqueness of the orthogonal projector

- **Encoding :** Optimization process.

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

$\Leftrightarrow$ Orthogonal projection on vector space $V_N = Vect\{\varphi_1, \ldots, \varphi_N\}$.

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

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

MIMESIS

Mesh-based methods (FEM)

Encoding/Decoding

Approximation

Approximation

Idea : Project a certain form of the equation onto the vector space V_N.
We introduce the residual inside Ω and on the boundary $\partial\Omega$ defined by

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

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

$$u = \underset{v \in H_1^0(\Omega)}{\operatorname{argmin}} J(v) \quad \longrightarrow \quad \theta_u = \underset{\theta \in \mathbb{R}^N}{\operatorname{argmin}} J(\theta)$$

with J a functional to minimize.

Variants : Depends on the problem form used for projection.

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

Energetic form

Discrete Minimization Problem :

$$u_\theta(x) = \underset{v \in V_N}{\operatorname{argmin}} J(v), \qquad J(v) = J_{in}(v) + J_{bc}(v) \tag{2}$$

with

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

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

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

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

Appendix 2

u_θ sol of (2) $\Leftrightarrow$ $\begin{cases} \nabla_v J_{in}(u_\theta) = 0 \\ \nabla_v J_{bc}(u_\theta) = 0 \end{cases}$ $\Leftrightarrow$ $\begin{cases} R_{in}(u_\theta) = 0 \text{ in } \Omega \\ u_\theta = g \text{ on } \partial\Omega \end{cases}$ $\Leftrightarrow$ u_θ approx sol of (1)

Discrete min pb **PDE**

MIMESIS

Galerkin Projection

DoFs minimization Problem :

$$\theta_u = \underset{\theta \in \mathbb{R}^N}{\mathrm{argmin}}\, J(\theta), \qquad J(\theta) = J_{in}(\theta) = \frac{1}{2}\int_\Omega L(v_\theta)v_\theta - \int_\Omega f v_\theta \tag{3}$$

Remark : Here, we are only interested in the minimisation problem on Ω.

Galerkin projection : Consists in resolving

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

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_θ approx sol of (1)	$\Leftrightarrow$	u_θ sol of (2)	$\Leftrightarrow$	θ_u sol of (3)	$\Leftrightarrow$	$\nabla_\theta J(\theta) = 0$	$\Leftrightarrow$	u_θ sol of (4)
PDE		**Discrete min pb**		**DoFs min pb**				**Galerkin projection**

MIMESIS

Least-Square form

Discrete Minimization Problem :

$$u_\theta(x) = \operatorname*{argmin}_{v \in V_N} J(v), \qquad J(v) = J_{in}(v) + J_{bc}(v)$$

with

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

DoFs minimization Problem :

$$\theta_u = \operatorname*{argmin}_{\theta \in \mathbb{R}^N} J(\theta), \qquad J(\theta) = J_{in}(\theta) = \frac{1}{2}\int_\Omega (L(v_\theta) - f)^2$$

Least-Square Galerkin projection : Consists in resolving

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

MIMESIS

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 $\langle R(u_\theta), \varphi_i\rangle_{L^2} = 0$	LS Galerkin $\langle R(u_\theta), (\nabla_\theta R(u_\theta))_i\rangle_{L^2} = 0$	$u_\theta(x) = \mathcal{D}(\theta_u)(x) = \sum_{i=1}^{N} (\theta_u)_i \varphi_i$
	$A\theta_u = B$		

Example : Galerkin projection.
For $i \in \{1, \dots, N\}$,

$$
\begin{aligned}
& \langle R(u_\theta), \varphi_i\rangle_{L^2} = 0 \\
\iff \quad & \int_\Omega L(u_\theta)\varphi_i = \int_\Omega f\varphi_i \\
\iff \quad & \sum_{j=1}^{N} (\theta_u)_j \int_\Omega \varphi_i L(\varphi_j) = \int_\Omega f\varphi_i
\end{aligned}
$$

$A\theta_u = B$ with

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

MIMESIS

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).
$\Rightarrow$ non-linear decoding $\Rightarrow$ approximation space $\mathcal{M}_N$ = finite-dimensional manyfold
$\Rightarrow$ there is no unique projector

- **Encoding :** Optimization process.

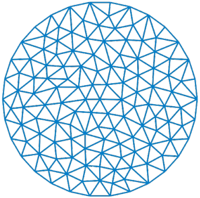
$$\theta_f = \mathcal{E}(f) = \underset{\theta \in \mathbb{R}^N}{\text{argmin}} \int_\Omega ||f_\theta(x) - f(x)||^2 dx$$

Neural Network Decoder

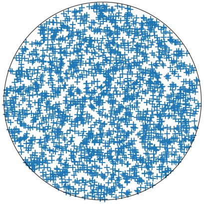
Advantages of a non-linear decoder :

- 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
$\Rightarrow$ use meshes



NN models
$\Rightarrow$ no need to use meshes



MIMESIS

Physically Informed Learning

Encoding/Decoding

Approximation

Approximation

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

Discretization : Degrees of freedom problem (no mesh).

$$u = \underset{v \in H_1^0(\Omega)}{\operatorname{argmin}} J(v) \quad \longrightarrow \quad \theta_u = \underset{\theta \in \mathbb{R}^N}{\operatorname{argmin}} J(\theta)$$

with J a functional to minimize.

Variants : Depends on the problem form used for projection.

Symmetric spatial PDE	**Any type of PDE**
Problem - Energetic form	Problem - Least-square form
Deep-Ritz	Standard PINNs
(Galerkin projection)	(Galerkin Least-square projection)

MIMESIS

Deep-Ritz

DoFs Minimization Problem : Considering the energetic form of our PDE, our discrete problem is

$$\theta_u = \underset{\theta \in \mathbb{R}^N}{\operatorname{argmin}} \, \alpha J_{in}(\theta) + \beta J_{bc}(\theta) \tag{5}$$

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.

- $(x_1, \ldots, x_n)$ randomly drawn on Ω

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

- $(y_1, \ldots, y_{n_b})$ randomly drawn 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 α and β still need to be determined

MIMESIS

Standard PINNs

DoFs Minimization Problem : Considering the least-square form of our PDE, our discrete problem is

$$\theta_u = \underset{\theta \in \mathbb{R}^N}{\operatorname{argmin}}\, \alpha J_{in}(\theta) + \beta J_{bc}(\theta) \tag{6}$$

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.

Steps Decomposition - NNs

<table>
<tr><th>Encoding</th><th colspan="2">Approximation</th><th>Decoding</th></tr>
<tr><td colspan="4">Mesh-based Methods</td></tr>
<tr><td rowspan="2">$\theta_f = \mathcal{E}(f) = M^{-1}b(f)$</td><td>Galerkin
$\langle R(u_\theta), \varphi_i \rangle = 0$</td><td>LS Galerkin
$\langle R(u_\theta), (\nabla_\theta R(u_\theta))_i \rangle = 0$</td><td rowspan="2">$u_\theta(x) = \mathcal{D}(\theta_u)(x) = \sum_{i=1}^{N} (\theta_u)_i \varphi_i$</td></tr>
<tr><td colspan="2">$A\theta_u = B$</td></tr>
<tr><td colspan="4">Physically informed learning</td></tr>
<tr><td rowspan="2">$\theta_f = \min_{\theta \in \mathbb{R}^N} \int_\Omega \lvert\lvert f_\theta - f \rvert\rvert^2$</td><td>Deep-Ritz
Energetic Form</td><td>Standard PINNs
LS Form</td><td rowspan="2">$u_\theta(x) = u_{NN}(x)$</td></tr>
<tr><td colspan="2">$\theta_u = \mathrm{argmin}_{\theta \in \mathbb{R}^N} J(\theta)$</td></tr>
</table>

Connection : Mesh-Based Methods // Physically Informed Learning

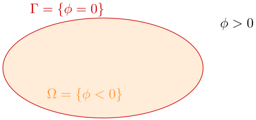
MIMESIS

Our hybrid method

ϕ-FEM Method

Main ideas : Appendix 8

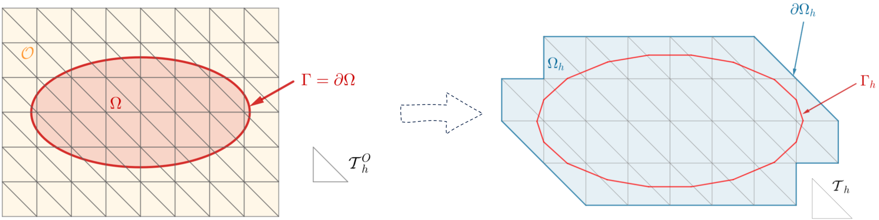
- Domain defined by a LevelSet Function ϕ.



- Mesh of a fictitious domain containing Ω.

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

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



MIMESIS

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_θ 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

$$\theta_u = \underset{\theta \in \mathbb{R}^N}{\operatorname{argmin}} J_{in}(\theta) + \cancel{J_{bc}(\theta)}$$

with

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

Connection : ϕ-FEM // Exact BC in PINNs

MIMESIS

Conclusion

Conclusion

What has been seen ?

- ”Physical Informed Learning” = extension of classic numerical methods
  $\rightarrow$ where decoder belongs to a manyfold
- advantage in high dimensions (parametric PDEs)
- advantage in the context of complex geometries (mesh-free methods)

Our hybrid approach : Appendix 7

- It combines
  - ➙ Speed of neural networks in predicting a solution
  - ➙ Precision of FEM methods to correct and certify the prediction of the NN (which can be completely wrong, on an unknown dataset for example)
- Encouraging results on simple geometries Appendix 9
- Difficulties on complex geometries - Important that its derivatives don’t explode $\rightarrow$ Next step: learning levelset functions (Eikonal equation)

MIMESIS

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. ϕ-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 ϕ-FEM approach for problems with natural boundary conditions*. 2020.

[5] Michel Duprez, Vanessa Lleras, and Alexei Lozinski. ϕ-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. ϕ-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*.

MIMESIS

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

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

with

$$M_{ij} = \langle \varphi_i, \varphi_j \rangle = \int_\Omega \varphi_i(x)\varphi_j(x)\, dx$$

$$b_i(f) = \langle f, \varphi_i \rangle = \int_\Omega f(x)\varphi_i(x)\, dx$$

MIMESIS

Appendix 2 : Energetic form I

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

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

- 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

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

And thus

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

MIMESIS

Appendix 2 : Energetic form II

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

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

Then

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

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 θ with

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

First, we define

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

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

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

MIMESIS

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.

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

And thus

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

MIMESIS

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

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

And thus

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

Finally

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

MIMESIS

Appendix 5 : LS Galerkin Projection

Let's compute the gradient of J with respect to θ with

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

First, we define

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

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

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

MIMESIS

Physically-Informed Learning

Appendix 6 : ADAM Method

ADAM = "Adaptive Moment Estimation" - Combine the idea of Moment and RMSProp.

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

with

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

MIMESIS

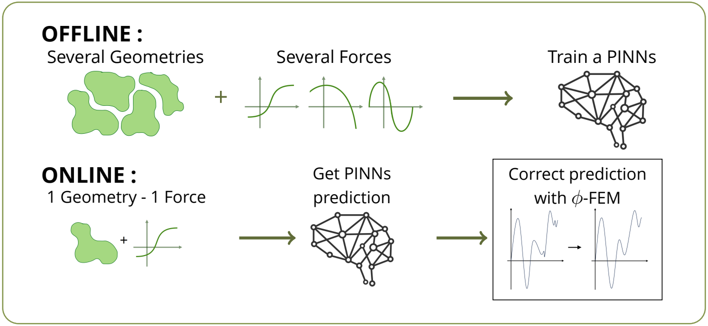
Our hybrid method

Appendix 7 : Description
Appendix 8 : ϕ-FEM Method
Appendix 9 : Results

Our hybrid method

Appendix 7 : Objective

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

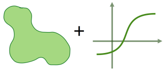


On going work :

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

Appendix 7 : Correction

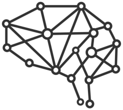
1 Geometry - 1 Force



ϕ and f

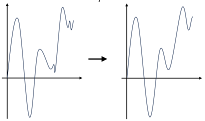
(and g)

Get PINNs prediction



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

Correct prediction with ϕ-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

$$\tilde{u} = u_{NN} + \underbrace{\boxed{\tilde{C}}}_{\ll 1}$$

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 ϕ-FEM method.

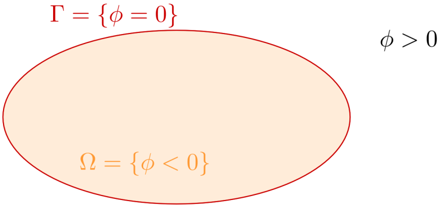
Our hybrid method

Appendix 8 : Problem

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

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

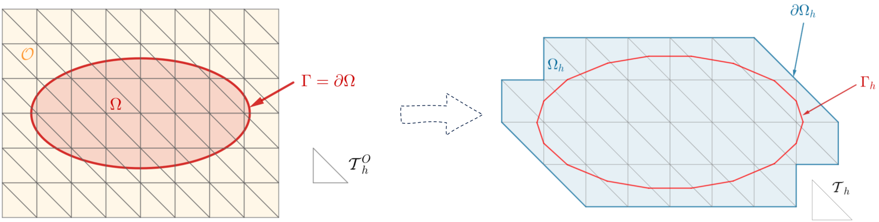
where ϕ is the level-set function and Ω and Γ are given by :



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

Remark : Thanks to ϕ and g, the boundary conditions are respected.

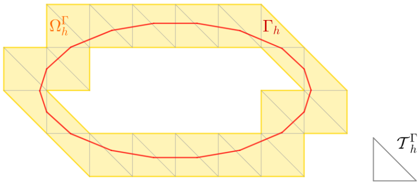
Appendix 8 : Fictitious domain

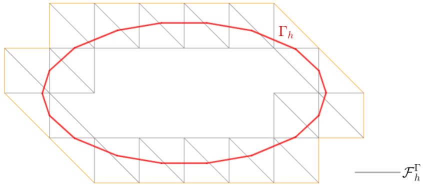


- ϕ_h : approximation of ϕ
- $\Gamma_h = \{\phi_h = 0\}$: approximate boundary of Γ
- Ω_h : computational mesh
- $\partial\Omega_h$: boundary of Ω_h ($\partial\Omega_h \neq \Gamma_h$)

Remark : n_{vert} will denote the number of vertices in each direction

Appendix 8 : Facets and Cells sets





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

Appendix 8 : 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

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

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

Stabilization terms

and

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

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 Ω_h.

Appendix 8 : Stabilization terms

Independent parameter of h

Jump on the interface E

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

1st order term

2nd order term

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

$-$

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

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

$\rightarrow$ Ghost penalty [Burman, 2010]

2nd term : require the solution to verify the strong form on Ω_h^Γ.

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

MIMESIS

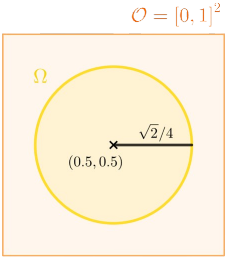
Our hybrid method

Appendix 9 : Problem considered

PDE : Poisson problem with Homogeneous Dirichlet conditions
Find $u : \Omega \to \mathbb{R}^d (d = 1, 2, 3)$ such that

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

with Δ the Laplace operator, Ω a smooth bounded open set and Γ its boundary.
Geometry : Circle - center=$(0.5, 0.5)$, radius=$\sqrt{2}/4$



→ **Level-set function :**

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

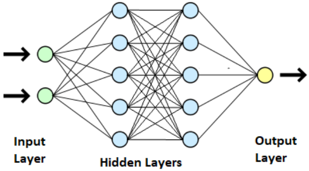
→ **Exact solution :**

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

MIMESIS

Appendix 9 : Networks

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



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

$$\boxed{\begin{array}{l} inputs = \{(x_i, y_i)\} \\ outputs = \{u_i\}_{i=1,\dots,n_{pts}} \end{array}}$$

$u_i = \phi(x_i, y_i) w_\theta(x_i, y_i)$

- n_layers=4
- n_neurons=20 (in each layer)
- n_epochs=10000
- n_pts=2000 (randomly drawn in the square $[0,1]^2$)

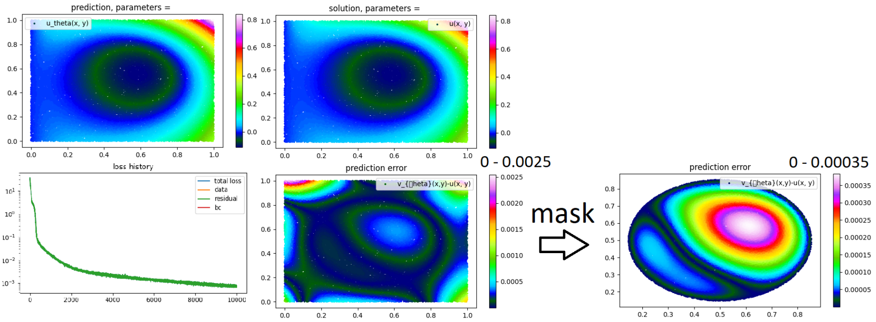
with $(x_i, y_i) \in \mathcal{O}$.

Remark : We impose exact boundary conditions.

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 Ω_h, not on Ω (training on the square for the moment).

Appendix 9 : Training

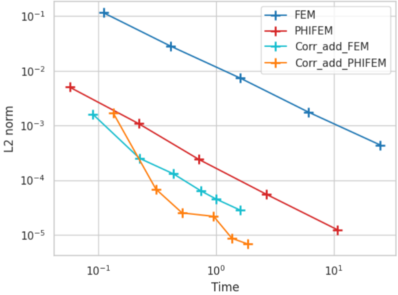


⚠ We consider a single problem (f fixed) on a single geometry (ϕ fixed).

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

Appendix 9 : Correction

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



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 ϕ-FEM would be parallelized

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

FEM / ϕ-FEM : $n_{vert} \in \{8, 16, 32, 64, 128\}$
Corr : $n_{vert} \in \{5, 10, 15, 20, 25, 30\}$
Remark : The stabilisation parameter σ of the ϕ-FEM method has a major impact on the error obtained.