Macaron/Tonus retreat presentation

Mesh-based methods and physically informed learning

Authors:
LECOURTIER Frédérique

Supervisors:
DUPREZ Michel
FRANCK Emmanuel
LLERAS Vanessa

February 6-7, 2024

MIMESIS

Introduction

Scientific context

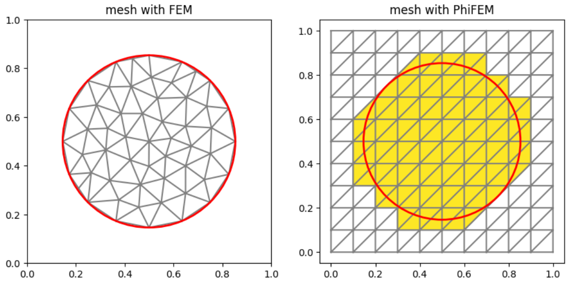
Context : Create real-time digital twins of an organ (such as the liver).

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

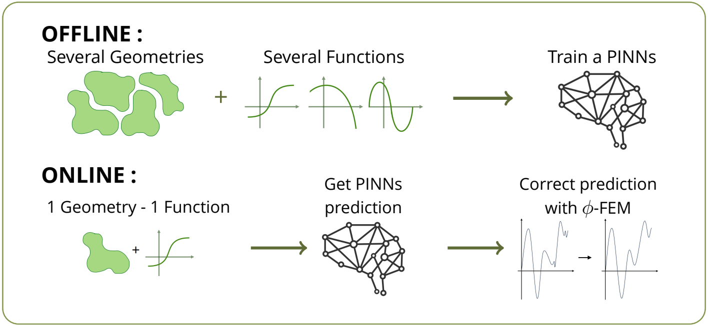
➙ ensure geometric quality



Practical case: Real-time simulation, shape optimization...

Objective

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



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

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

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

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

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

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

MIMESIS

Numerical methods

Objective : Show that the philosophy behind most ofd the methods are the same.

Mesh-based methods // Physically informed learning

Numerical methods : Discrete 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
$f \rightarrow \theta_f$	$\theta_f \rightarrow \theta_u$	$\theta_u \rightarrow u_\theta$

MIMESIS

Mesh-based methods

Encoding/Decoding

Approximation

Mesh-based methods

Encoding/Decoding

Approximation

Encoding/Decoding - FEMs

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

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

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

- **Encoding :** Orthogonal projection on vector space $V_N = Vect\{\varphi_1, \dots, \varphi_N\}$.

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

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

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

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

which respectively define the residues inside Ω 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.

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

MIMESIS

Energetic form

Minimization Problem :

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

with

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

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

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

Min pb **PDE**

MIMESIS

Galerkin Projection

Discrete minimization Problem :

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

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

PDE		Min pb		Discrete min pb				Galerkin projection
u_θ sol of (1)	$\Leftrightarrow$	u_θ sol of (2)	$\Leftrightarrow$	u_θ sol of (3)	$\Leftrightarrow$	$\nabla_\theta J(\theta) = 0$	$\Leftrightarrow$	u_θ sol of (4)

MIMESIS

Least-Square form

Minimization Problem :

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

with

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

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

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_θ 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_θ sol of (1)

Min pb **PDE**

A modifier !

MIMESIS

Least-Square Galerkin Projection

Discrete minimization Problem :

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

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

Galerkin projection : Consists in resolving

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

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

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

Appendix 5

u_θ sol of (1)	$\Leftrightarrow$	u_θ sol of (5)	$\Leftrightarrow$	u_θ sol of (6)	$\Leftrightarrow$	$\nabla_\theta J(\theta) = 0$	$\Leftrightarrow$	u_θ sol of (7)
PDE		**Min pb**		**Discrete min pb**				**LS Galerkin projection**

MIMESIS

Steps Decomposition - FEMs

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

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 V_N = finite-dimensional variety
$\Rightarrow$ there is no unique projector

- **Encoding :** Optimization process.

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

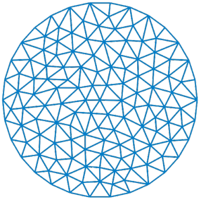
MIMESIS

Non-Linear Decoder

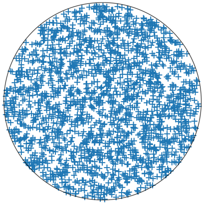
Advantages :

- 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
$\Rightarrow$ local precision
$\Rightarrow$ use meshes



NN models
$\Rightarrow$ global precision
$\Rightarrow$ no need to use meshes



MIMESIS

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).

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

with J a functional to minimize.

Variants : Depends on the problem form used for projection.

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

Deep-Ritz

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

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

with

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

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

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

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

Standard PINNs

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

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

with

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

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

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

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

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

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

MIMESIS

In practice...

- Two different random generation processes (to have enough boundary points)
- Weights in front of the cost functions still need to be determined
- Use regular model, derivable several times (and automatic differentiation)
- Activation functions regular enough to be derived 2 times (due to the Laplacian)
  $\Rightarrow$ Tangent Hyperbolic rather than ReLU
  (or adaptive methods where we parameterize the activation functions)
- Stochastic gradient descent method (by mini-batch) - ADAM method Appendix 6

To go further :

- Standard PINNs : possibility of adding a J_{data} cost function
  $\rightarrow$ 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 $\langle R(u_\theta), \varphi_i \rangle = 0$	LS Galerkin $\langle R(u_\theta), (\nabla_\theta R(u_\theta))_i \rangle = 0$	$u_\theta(x) = \mathcal{D}_\theta(x) = \sum_{i=1}^{N} (\theta_u)_i \varphi_i$
	$A\theta_u = B$		

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

MIMESIS

Hybrid method

ϕ-FEM Method

A compléter !

MIMESIS

Impose exact BC in PINNs

A compléter !

Correct PINNs prediction with ϕFEM

A compléter !

Conclusion

Conclusion

A compléter !

Bibliography

Bibliography

[1] Erik Burman. “Ghost penalty”. *Comptes Rendus. Mathématique* 348.21 (2010), pp. 1217–1220. ISSN: 1778-3569.

[2] Erik Burman et al. “CutFEM: Discretizing geometry and partial differential equations”. *International Journal for Numerical Methods in Engineering* 104.7 (2015), pp. 472–501. ISSN: 1097-0207.

[3] Stéphane Cotin et al. *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. “ϕ-FEM: an optimally convergent and easily implementable immersed boundary method for particulate flows and Stokes equations”. *ESAIM: Mathematical Modelling and Numerical Analysis* 57.3 (May 2023), pp. 1111–1142. ISSN: 2822-7840, 2804-7214.

[5] Michel Duprez, Vanessa Lleras, and Alexei Lozinski. “A new ϕ-FEM approach for problems with natural boundary conditions”. *Numerical Methods for Partial Differential Equations* 39.1 (2023), pp. 281–303. ISSN: 1098-2426.

[6] Michel Duprez and Alexei Lozinski. “ϕ-FEM: A Finite Element Method on Domains Defined by Level-Sets”. *SIAM Journal on Numerical Analysis* 58.2 (Jan. 2020), pp. 1008–1028. ISSN: 0036-1429.

[7] Zongyi Li et al. *Neural Operator: Graph Kernel Network for Partial Differential Equations*. Mar. 6, 2020.

[8] Zongyi Li et al. *Fourier Neural Operator for Parametric Partial Differential Equations*. May 16, 2021.

[9] Zongyi Li et al. *Fourier Neural Operator with Learned Deformations for PDEs on General Geometries*. July 11, 2022.

[10] Zongyi Li et al. *Physics-Informed Neural Operator for Learning Partial Differential Equations*. July 29, 2023.

[11] 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* 389 (Feb. 2022), p. 114333. ISSN: 00457825.

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 $\forall i \in \{1, \ldots, N\}$, 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

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

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

Appendix 2 : Energetic form I

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

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

- 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.

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

Then

$$\mathcal{D}J_{bc}(v) \cdot h = \lim_{\epsilon \to 0} \frac{J_{bc}(v + \epsilon h) - J_{bc}(v)}{\epsilon} = \int_\Omega v^2 - hg$$

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

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

- 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.

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

Then

$$\mathcal{D}J_{bc}(v)\cdot h = \lim_{\epsilon\to 0}\frac{J_{bc}(v+\epsilon h) - J_{bc}(v)}{\epsilon} = \int_\Omega v^2 - hg$$

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

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

Physically Informed Learning

Appendix 6 : ADAM Method

A compléter !