

HyperTop

Workshop on Advances in Shape and Topology
Optimization for Hyperelastic Materials

Barcelona, 23 – 25 September 2026

Programming shape-morphing ultra-soft magneto-active structures via topology optimization

R. Ortigosa, J. Martínez-Frutos

Universidad Politécnica de Cartagena

jesus.martinez@upct.es



MULTISIMO LAB

Multiphysics Simulation and Optimisation Lab · UPCT



OUTLINE

Contents



01 Our initial contact on TO in hyperelasticity

02 Previous work on Topology Optimization of active materials

03 Residual Mechanical Anisotropy in ultra-soft hMREs

04 Computational platform to Program Magneto-Mechanical Actuation

05 Multimaterial and Topology optimization algorithm

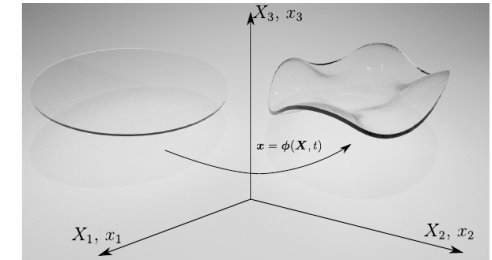
06 Julia implementation - HyperFEM



Nonlinear continuum mechanics and polyconvexity

- Strong form (PDE) in the Lagrangian formalism

$$\begin{aligned}\nabla_0 \times \mathbf{F} &= \mathbf{0}; & \text{in } \Omega_0 & \Rightarrow \mathbf{F} = \nabla_0 \phi \\ \nabla_0 \cdot \mathbf{P}(\mathbf{F}) + \mathbf{b}_0 &= \mathbf{0}; & \text{in } \Omega_0 & \mathbf{P} = \sigma \mathbf{H} \\ \phi &= \phi^*; & \text{on } \partial\Omega_{0,D} & \\ \mathbf{P}\mathbf{N} &= \mathbf{t}_0; & \text{on } \partial\Omega_{0,N} &\end{aligned}$$



- Constitutive relation through nonlinear stored energy functional $e(\mathbf{F})$: $\mathbf{P}(\mathbf{F}) = \frac{\partial e(\mathbf{F})}{\partial \mathbf{F}}$

- Polyconvexity condition [Ball, 1976; Schröder and Neff, 2004...] $e(\mathbf{F}) = W(\mathbf{F}, \mathbf{H}(\mathbf{F}), J(\mathbf{F}))$ $\mathbf{H} = \frac{1}{2} \mathbf{F} \times \mathbf{F}$; $J = \frac{1}{6} (\mathbf{F} \times \mathbf{F}) : \mathbf{F}$

- Polyconvex constitutive models:

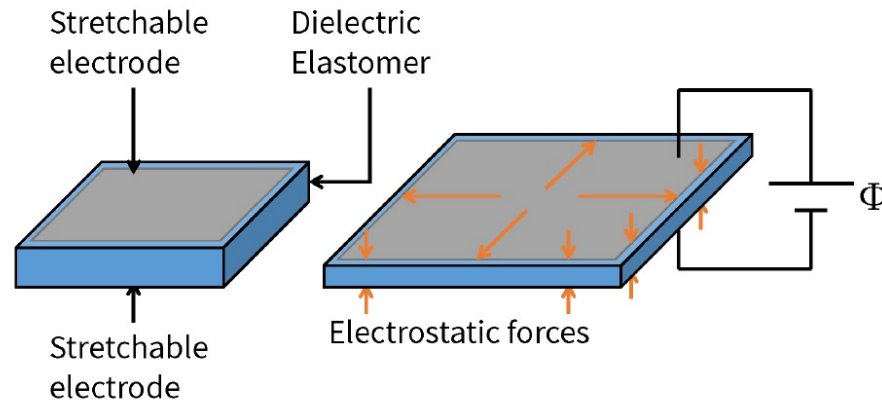
$$\begin{aligned}\text{NH } W(\mathbf{F}, J) &= \frac{\mu}{2} \|\mathbf{F}\|^2 + f(J); & f(J) &= -\mu \ln J + \frac{\lambda}{2} (J - 1)^2 \\ \text{MR } W(\mathbf{F}, \mathbf{H}, J) &= \frac{\alpha}{2} \|\mathbf{F}\|^2 + \frac{\beta}{2} \|\mathbf{H}\|^2 + g(J); & g(J) &= -(\alpha + 2\beta) \ln J + \frac{\lambda}{2} (J - 1)^2 \\ \text{Gent } W(\mathbf{F}, J) &= -\frac{\mu J_m}{2} \ln \left(1 - \frac{\|\mathbf{F}\|^2 - 3}{J_m} \right) + f(J); & &\end{aligned}$$

- Classical and polyconvexity-based expression for the first Piola-Kirchhoff stress tensor $\mathbf{P} = \partial_{\mathbf{F}} e$; $\mathbf{P} = \partial_{\mathbf{F}} W + \partial_{\mathbf{H}} W \times \mathbf{F} + \partial_J W \mathbf{H}$
- Fourth order elasticity tensor ($\mathcal{C} := \partial_{\mathbf{F}\mathbf{F}}^2 e$):

$$\mathcal{C} = \partial_{\mathbf{F}\mathbf{F}}^2 W_1 + \mathbf{F} \times \partial_{\mathbf{H}\mathbf{H}}^2 W_2 \times \mathbf{F} + \partial_J^2 W_3 \mathbf{H} \otimes \mathbf{H} + \mathcal{I} \times (\partial_{\mathbf{H}} W_2 + \partial_J W_3 \mathbf{F})$$

with $\mathcal{I}_{ijkl} = \delta_{ij} \delta_{kl}$ and $(\mathcal{A} \times \mathcal{A})_{ijkl} = \mathcal{A}_{ilpq} \mathcal{E}_{jpq} \mathcal{E}_{kpq} \mathcal{A}_{qQ}$ for $\mathbf{A} \in \mathbb{R}^{3 \times 3 \times 3 \times 3}$ and $\mathbf{A}^{3 \times 3}$

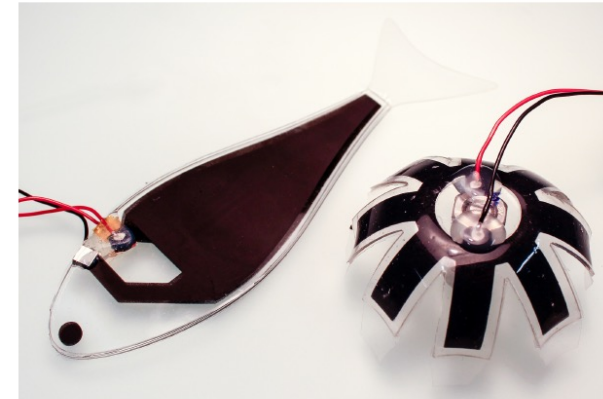
Electro-Active Polymers: working principle



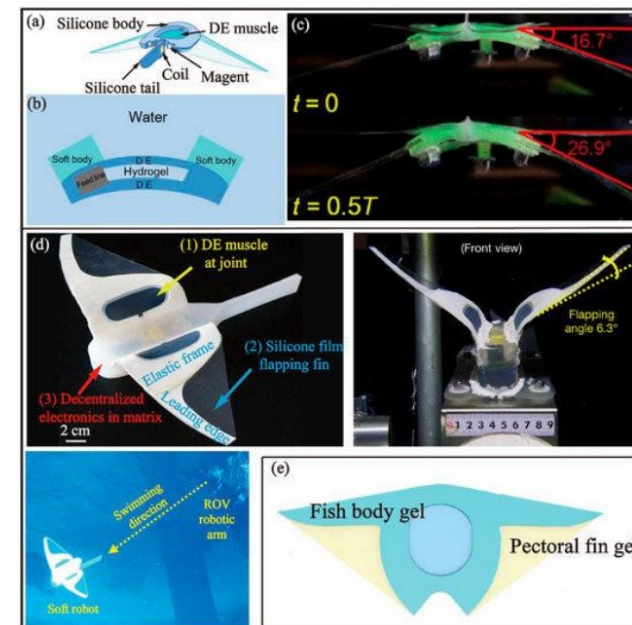
Physical principle: Coulomb forces



Giant electrically induced deformations in EAP membrane
[Suo 2013 (Harvard Lab), Max Plank Press 2016]



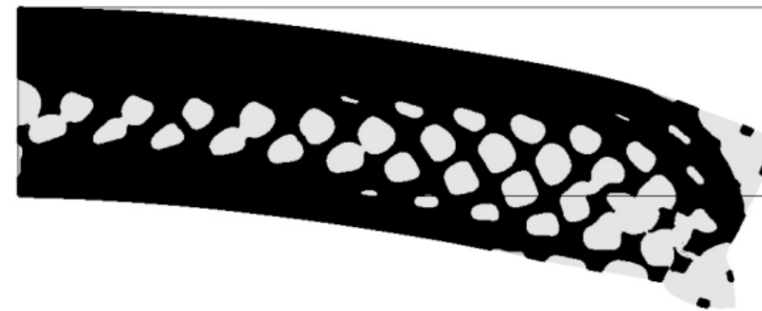
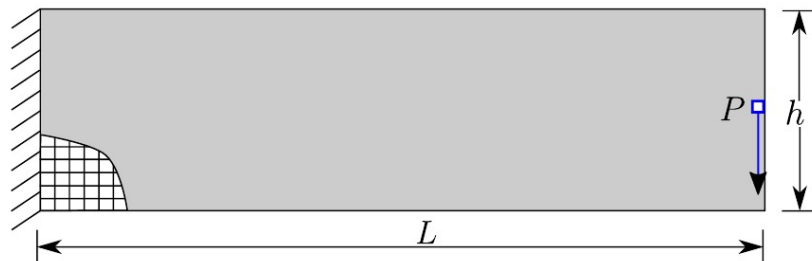
soft gripper



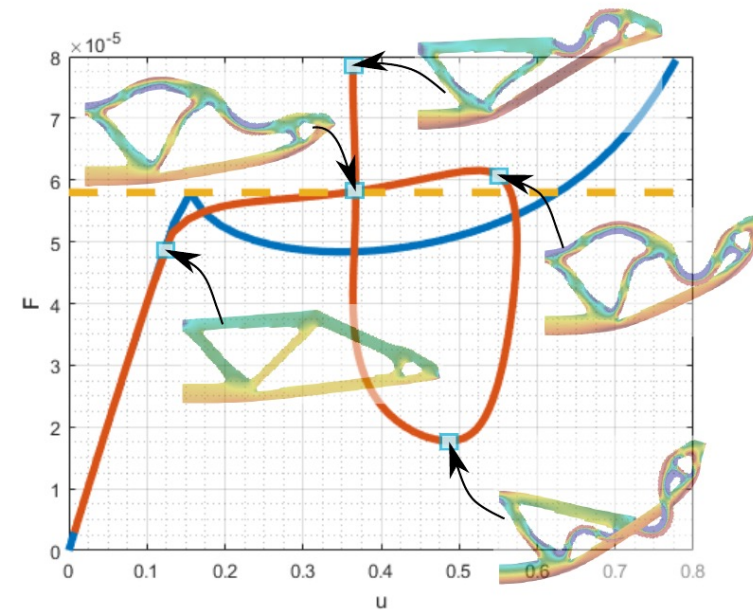
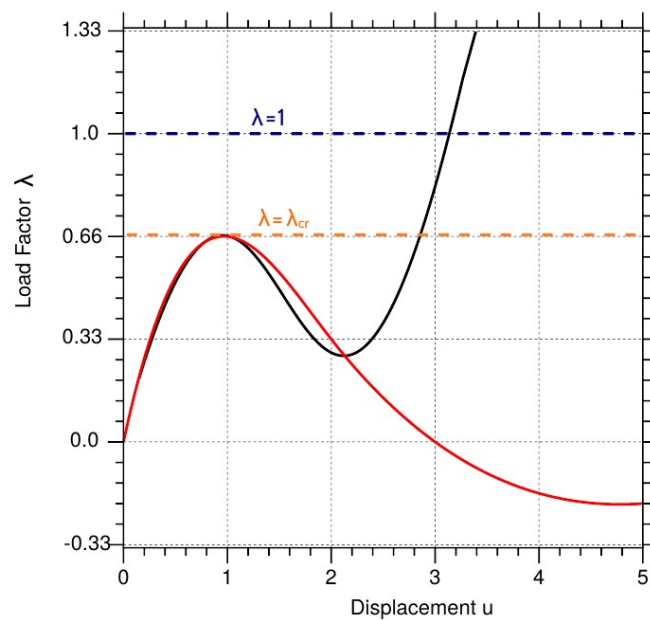
DE propelled manta ray

TO in hyperelasticity: stabilisation approach I

- **Remember:** Polyconvex strain energies are **not convex!**



- Can we use a standard **arc-length method** to capture the post-buckling response?



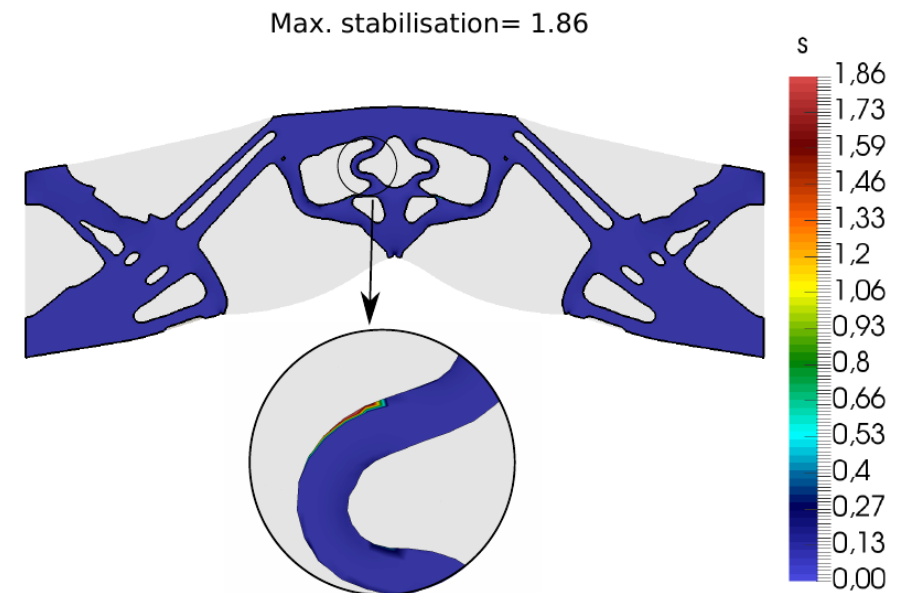
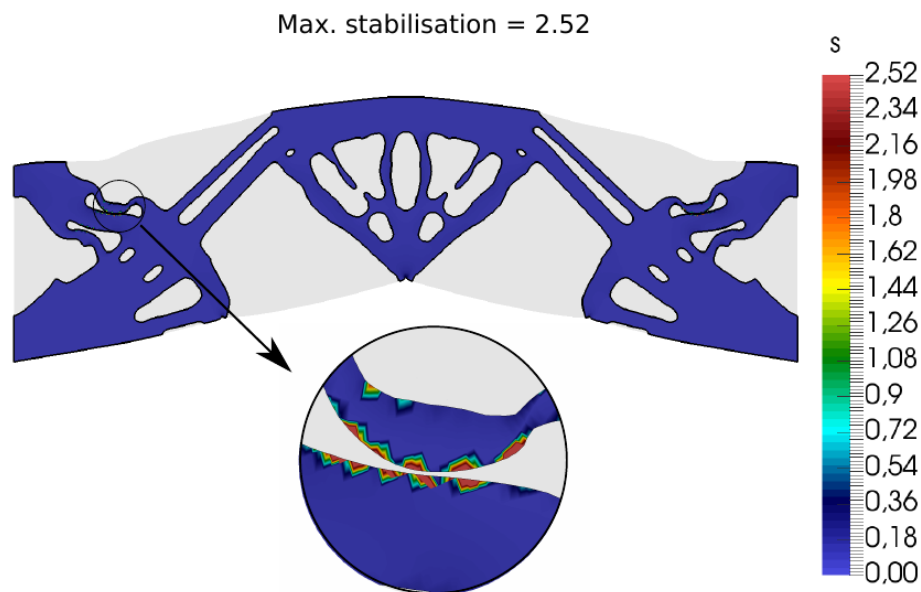
TO in hyperelasticity: stabilisation approach II

- **Beyond instability:** First order Taylor series expansion for P

$$P_{n+1} := P|_n + \mathcal{C}_{\text{stb}}|_n : \nabla_0 (u_{n+1} - u_n)$$

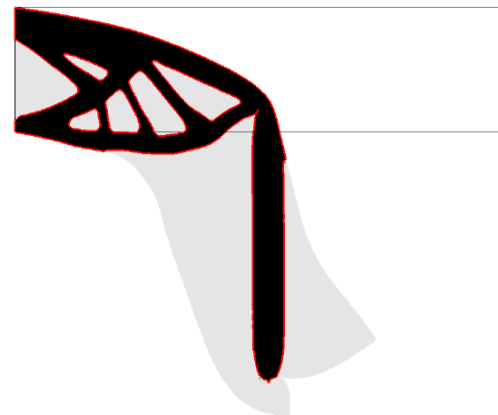
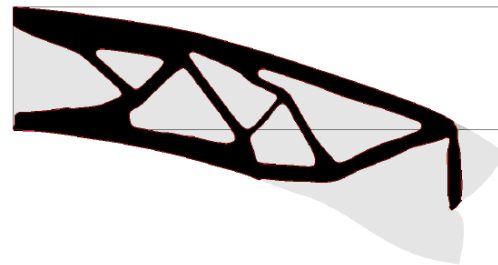
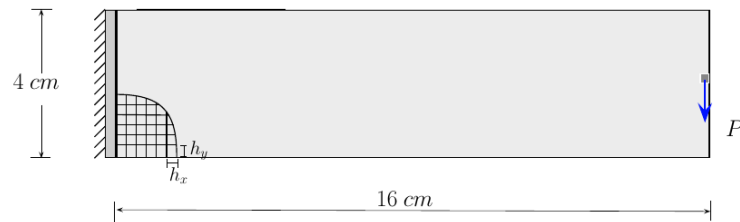
- **Stabilised** elasticity tensor

$$\mathcal{C}_{\text{stb}}|_n = \mathcal{C}|_n + \Delta \mathcal{C}_{\text{stb}}|_n; \quad V : \mathcal{C}_{\text{stb}}|_n : V \geq 0; \quad V \in \mathbb{R}^{3 \times 3}; \quad \Delta \mathcal{C}_{\text{reg}}|_n = (1 + \delta) \Gamma \mathcal{I},$$

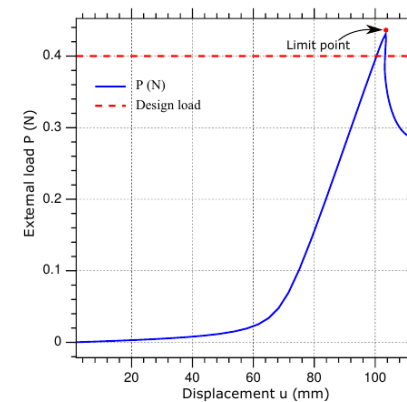
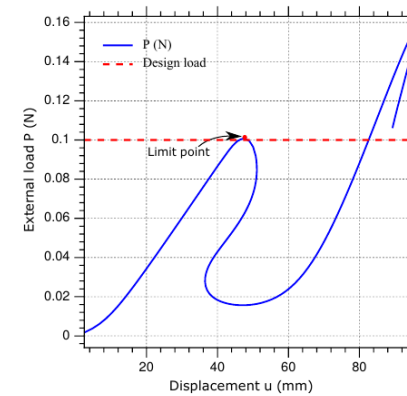
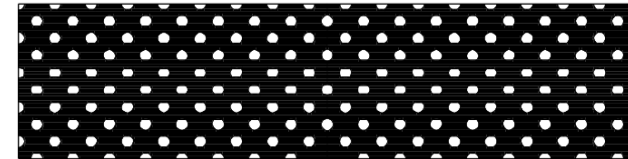


TO in hyperelasticity: numerical example 2

- Geometry and BC's of the problem

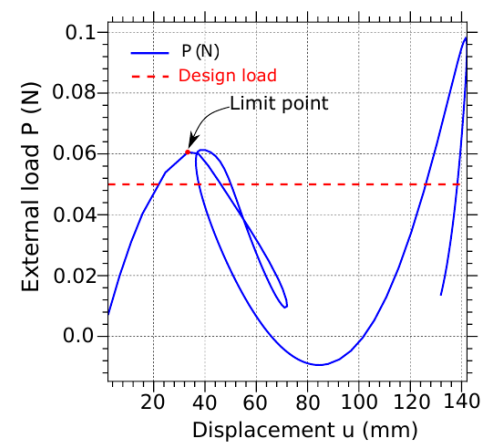
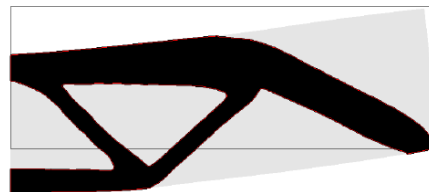
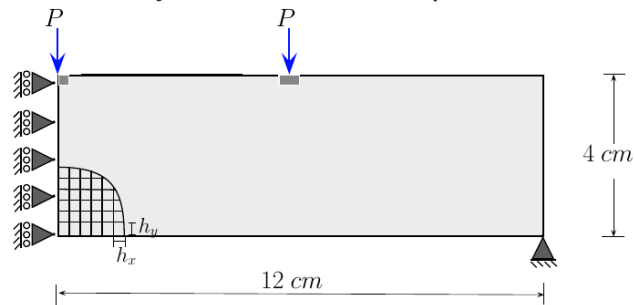


- Initial seed

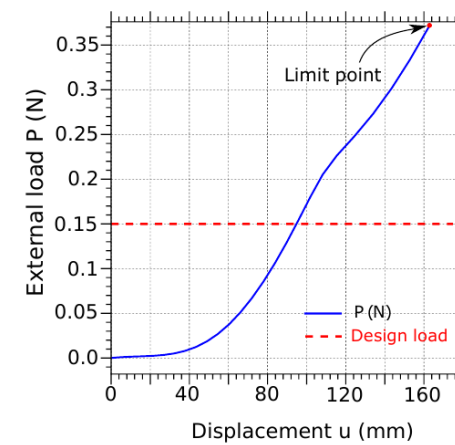
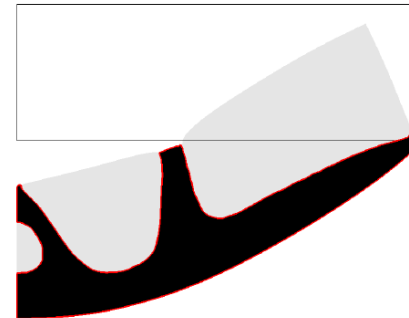
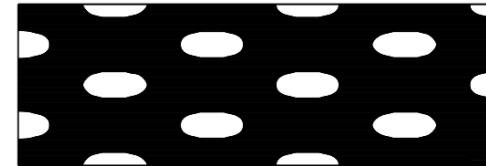


TO in hyperelasticity: numerical example 3

- Geometry and BC's of the problem



- Initial seed



OUTLINE

Contents



01 Our initial contact on TO in hyperelasticity

02 Previous work on Topology Optimization of active materials

03 Residual Mechanical Anisotropy in ultra-soft hMREs

04 Computational platform to Program Magneto-Mechanical Actuation

05 Multimaterial and Topology optimization algorithm

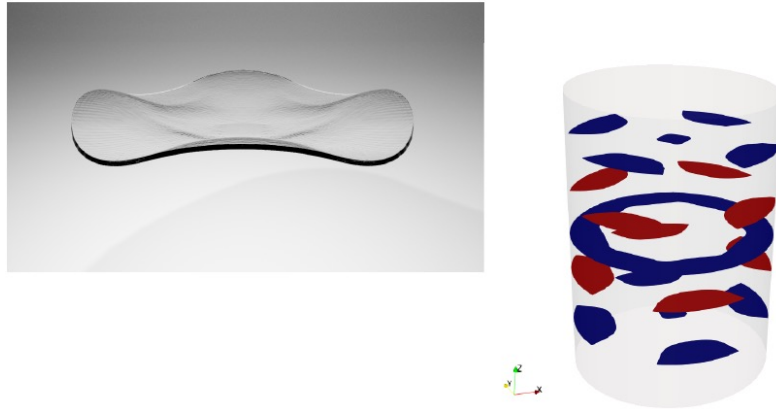
06 Julia implementation - HyperFEM



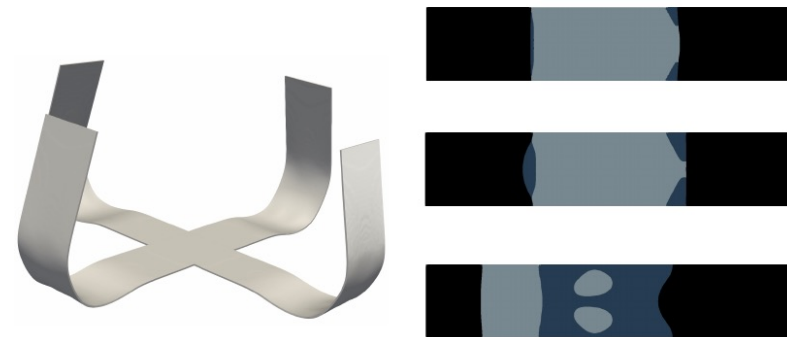
HyperTop

TO-driven design of active materials

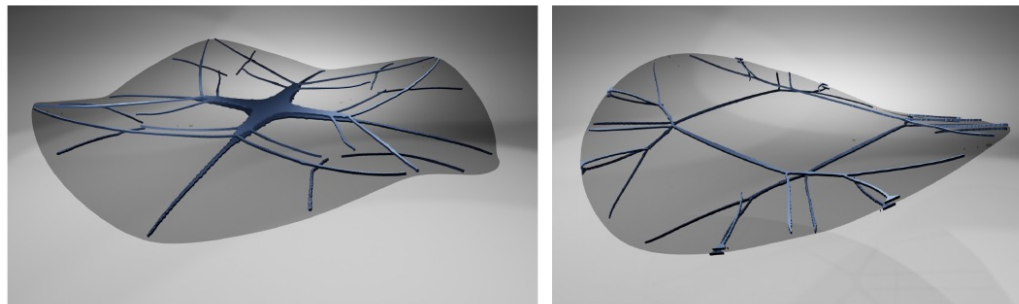
TO-driven electrode design



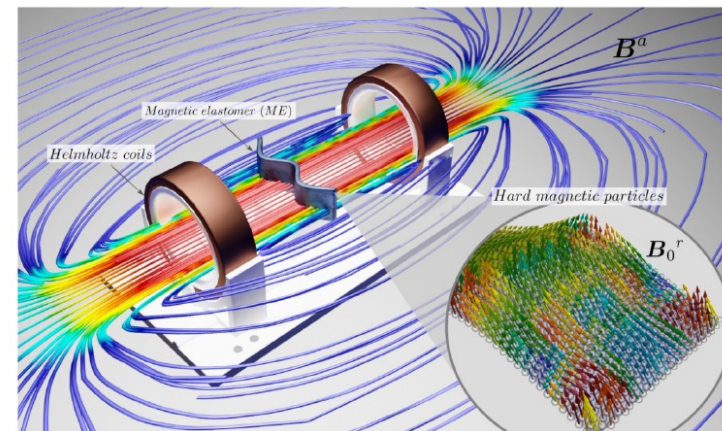
TO-driven multimaterial actuators



TO-driven stiffeners design



TO-driven residual magnetization design



OUTLINE

Contents



01 Our initial contact on TO in hyperelasticity

02 Previous work on Topology Optimization of active materials

03 Residual Mechanical Anisotropy in ultra-soft hMREs

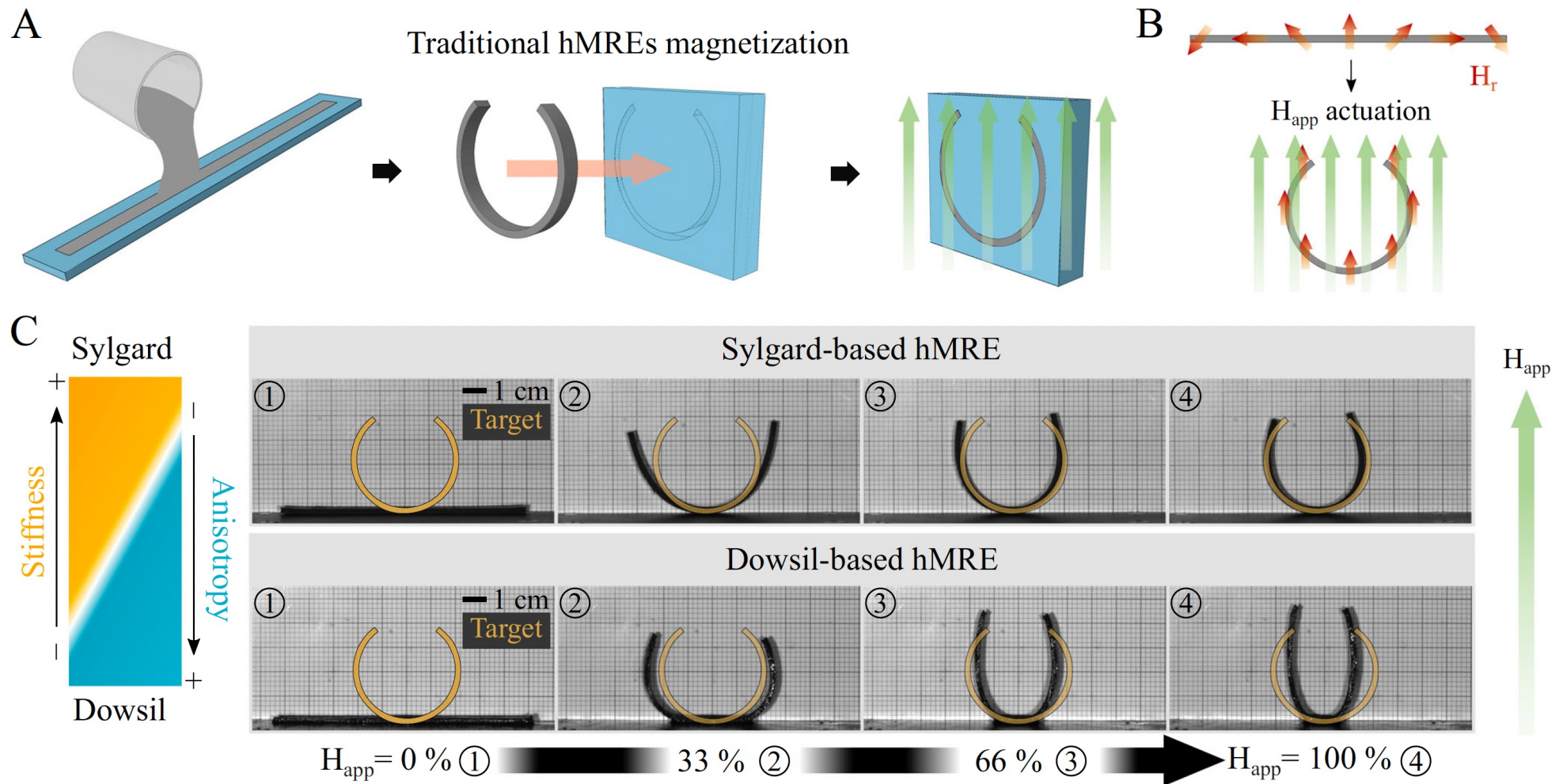
04 Computational platform to Program Magneto-Mechanical Actuation

05 Multimaterial and Topology optimization algorithm

06 Julia implementation - HyperFEM



Shape programming of ultra-soft hMRES



Sylgard 184 (≈ 2 MPa)

f SéNeCa⁽⁺⁾
Agencia de Ciencia y Tecnología
Región de Murcia

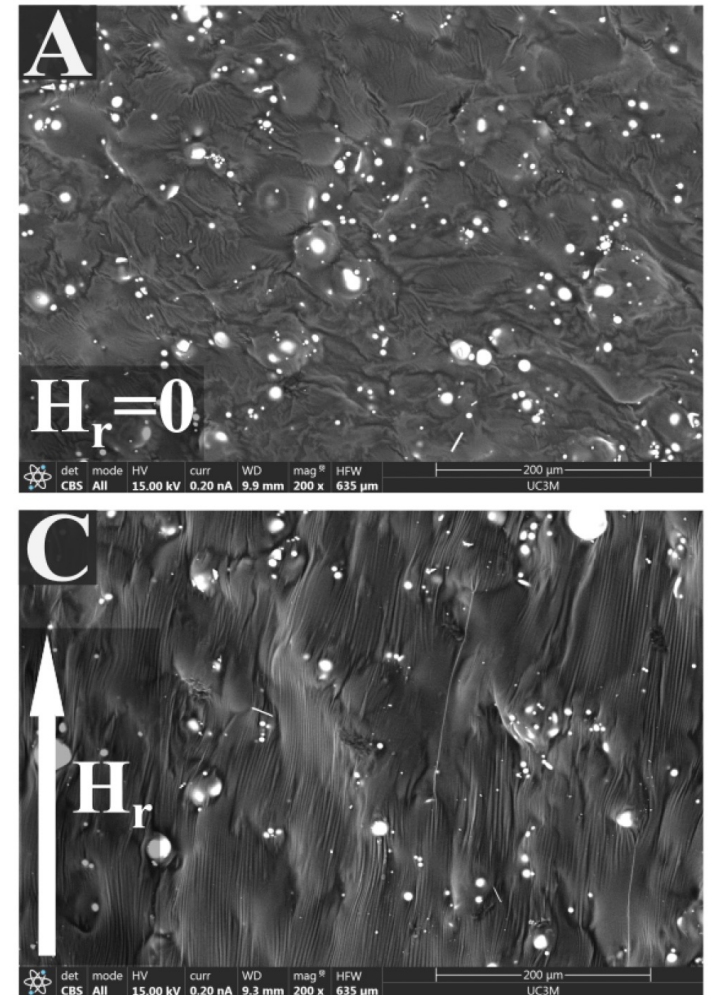
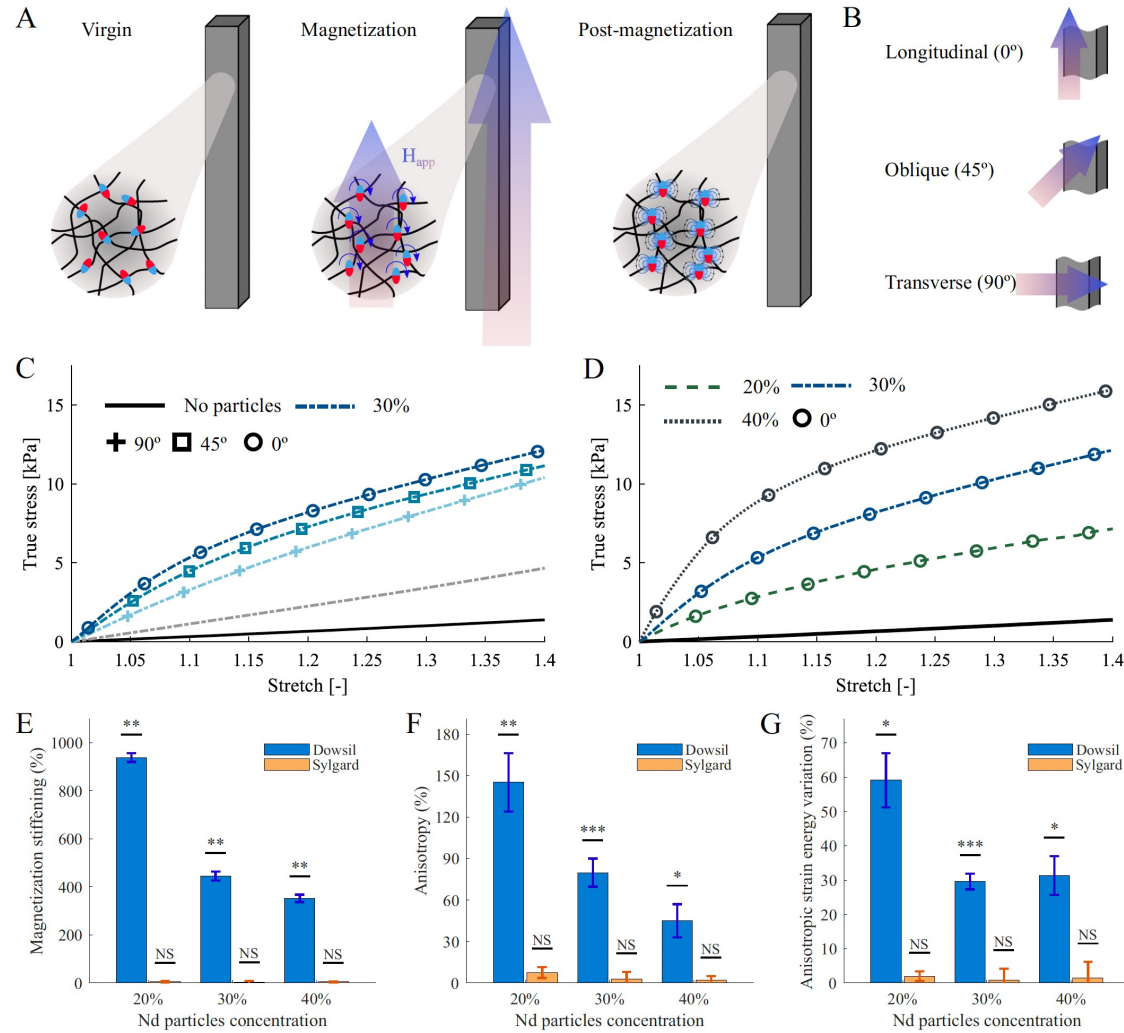


Dowsil CY52-276 (≈ 3 kPa)

f SéNeCa⁽⁺⁾
Agencia de Ciencia y Tecnología
Región de Murcia



Residual Mechanical Anisotropy in ultra-soft hMREs



OUTLINE

Contents



01 Our initial contact on TO in hyperelasticity

02 Previous work on Topology Optimization of active materials

03 Residual Mechanical Anisotropy in ultra-soft hMREs

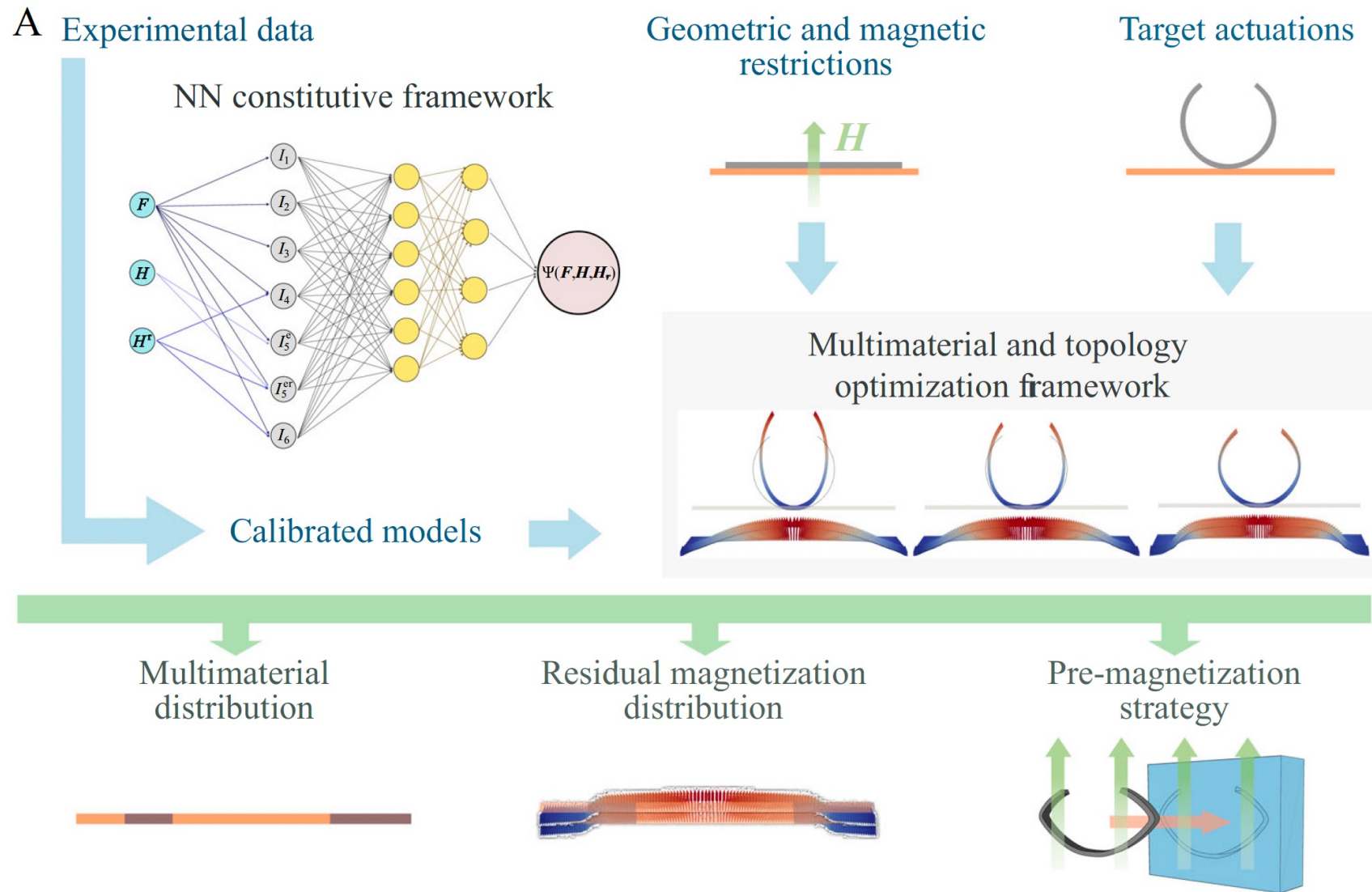
04 Computational platform to Program Magneto-Mechanical Actuation

05 Multimaterial and Topology optimization algorithm

06 Julia implementation - HyperFEM

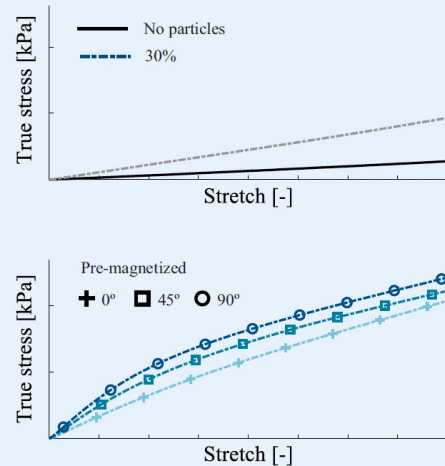


Computational framework



Physics-augmented neural networks constitutive model

A Input: Experimental data



Characterization on samples:

- Non-magnetized
- Pre-magnetized
- Pre-magnetized in different directions

Physical fields

F

H

H_r

- Mechanics
- Magnetics
- Coupling

Input of NN

I_1

I_2

I_3

I_4

I_5^e

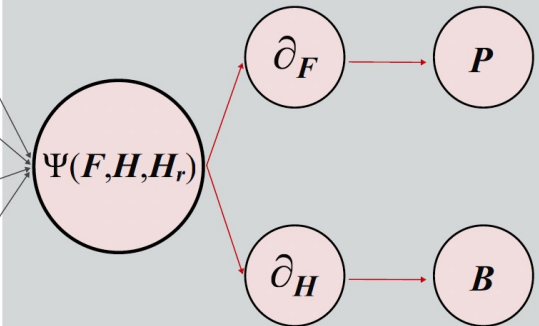
I_5^{er}

I_6

- Objectivity
- Transverse isotropy

Architecture of NN

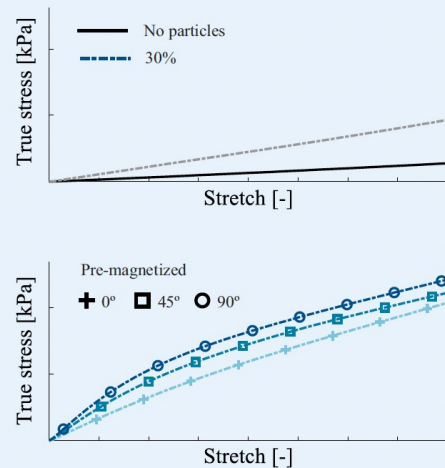
Output of NN: Constitutive Model



- Sobolev training
- Policonvex energy function
- Stress-free at origin ($F=I, H=0$)

Physics-augmented neural networks constitutive model

A Input: Experimental data



Characterization on samples:

- Non-magnetized
- Pre-magnetized
- Pre-magnetized in different directions

Physical fields

F

H

H_r

- Mechanics
- Magnetics
- Coupling

Input of NN

I_1

I_2

I_3

I_4

I_5^e

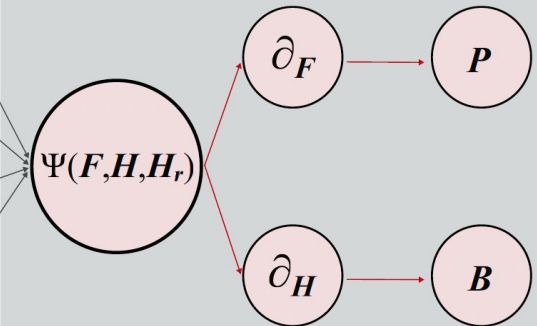
I_5^{er}

I_6

- Objectivity
- Transverse isotropy

Architecture of NN

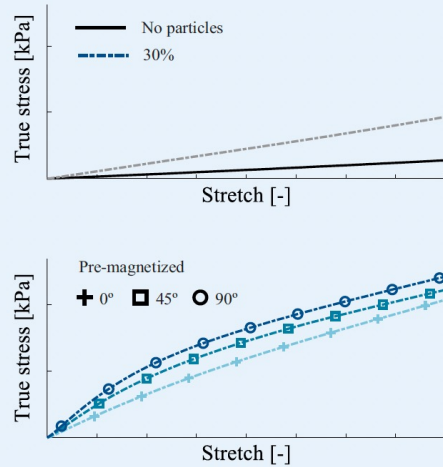
Output of NN: Constitutive Model



- Sobolev training
- Policonvex energy function
- Stress-free at origin ($F=I, H=0$)

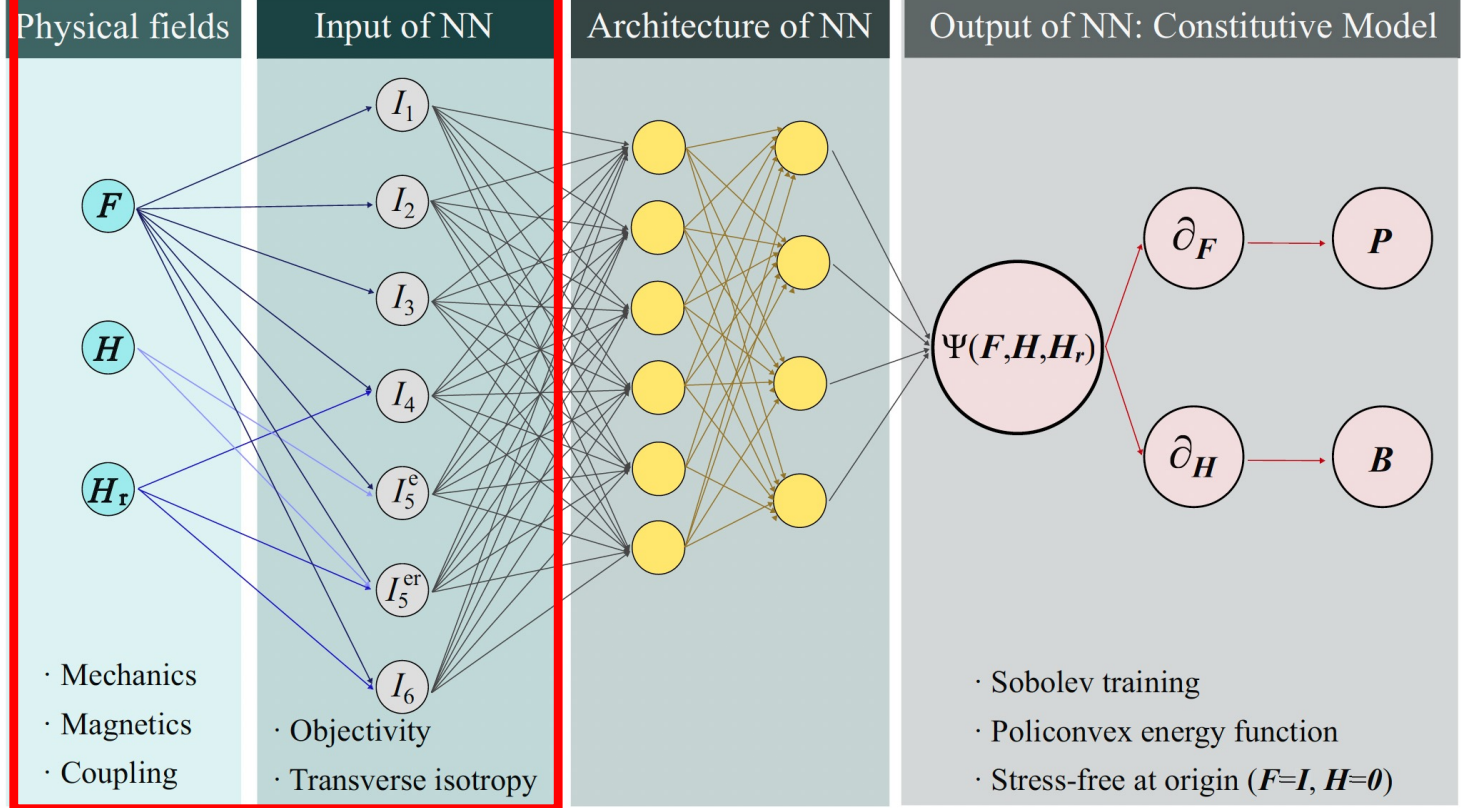
Physics-augmented neural networks constitutive model

A Input: Experimental data



Characterization on samples:

- Non-magnetized
- Pre-magnetized
- Pre-magnetized in different directions



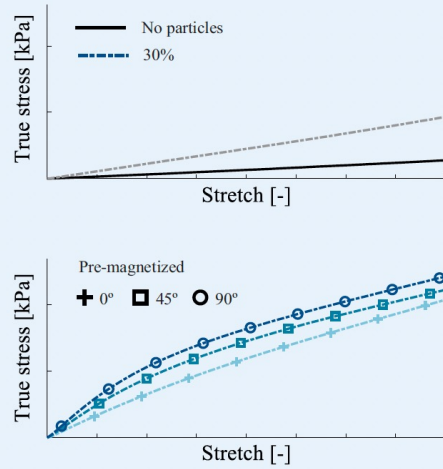
$$I_1 = F : F; \quad I_2 = \text{Cof}F : \text{Cof}F; \quad I_3 = \det F;$$

$$I_4 = FH_r \cdot FH_r; \quad I_6 = \text{Cof}FH_r \cdot \text{Cof}FH_r;$$

$$I_5^e = H \cdot C^{-1} \cdot H \quad I_5^{er} = H \cdot C^{1/2} \cdot H_r$$

Physics-augmented neural networks constitutive model

A Input: Experimental data



Characterization on samples:

- Non-magnetized
- Pre-magnetized
- Pre-magnetized in different directions

Physical fields

F

H

H_r

- Mechanics
- Magnetics
- Coupling

Input of NN

I_1

I_2

I_3

I_4

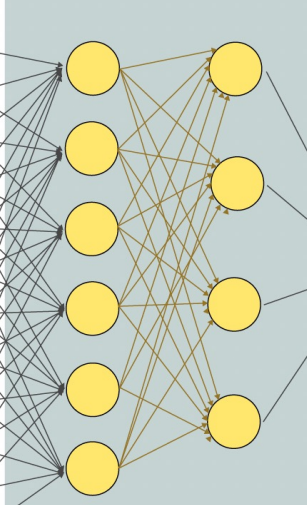
I_5^e

I_5^{er}

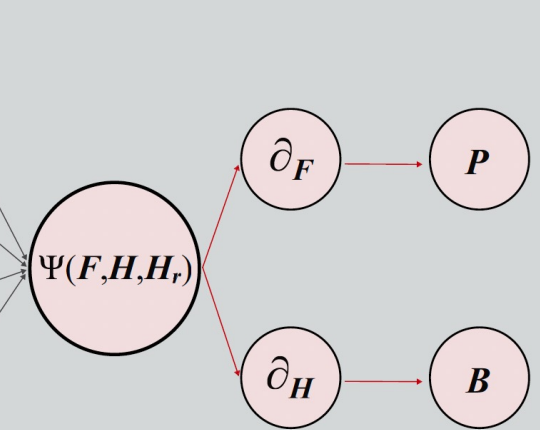
I_6

- Objectivity
- Transverse isotropy

Architecture of NN



Output of NN: Constitutive Model



- Sobolev training
- Policonvex energy function
- Stress-free at origin ($F=I, H=0$)

$$\Psi(F, H, \mathcal{W}) = \mathbb{U}(I_1, I_2, I_3, I_4, I_5, I_5^{er}, I_6; \mathcal{W})$$

$$\mathcal{W} = \{\mathbf{W}_1, \dots, \mathbf{W}_{L+1}, \mathbf{b}_1, \dots, \mathbf{b}_{L+1}\}$$

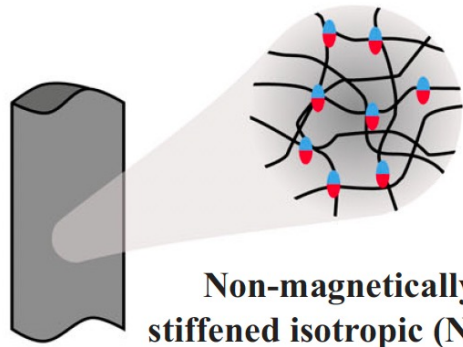
$$\mathcal{L}(\mathcal{W}) = \frac{\sum_{i=1}^{n_d} \|\mathbf{p}^i - \partial_F \Psi(F^i, \mathbf{0}, \mathcal{W})\|^2}{\sum_{i=1}^{n_d} \|\mathbf{p}^i\|^2} + \frac{\kappa_r}{2} \sum_i (\min((\mathcal{W}_v)_i, 0))^2$$

Physics-augmented neural networks constitutive model

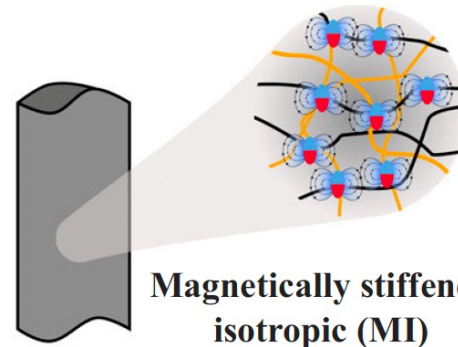
B

Constitutive model assumptions

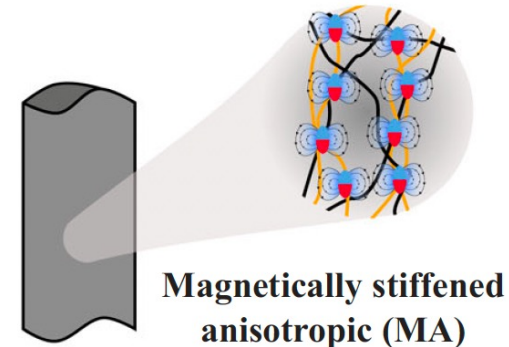
① H_r does not modulate mechanical properties of hMREs under $H = 0$



② H_r modulates stiffness

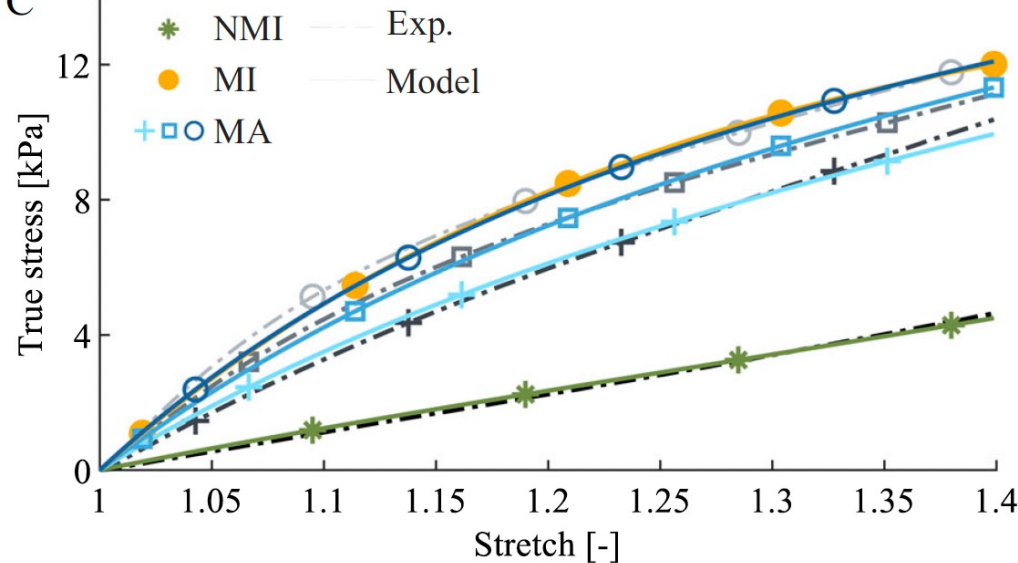


③ H_r modulates stiffness and introduces anisotropy

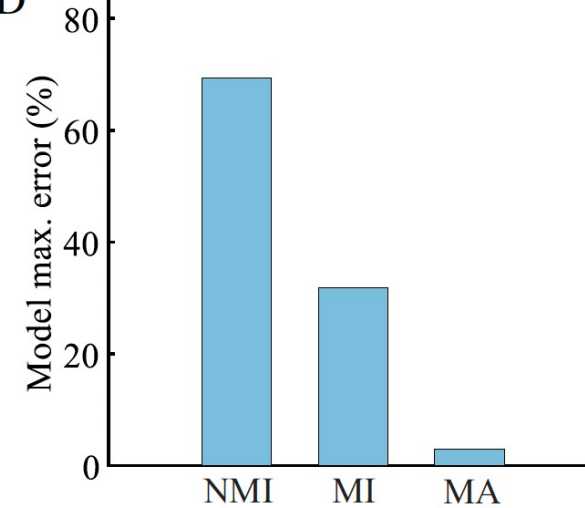


Unloaded chain Loaded chain Non-magnetic interactions Magnetic interactions

C



D



Governing equations in the whole medium



Magnetostatics in $\Omega_0 \cup \Omega_{v0}$: $\varphi \in \mathbb{V}^\varphi$

$$\begin{aligned} \text{DIV } \mathbf{B} &= 0 & \text{in } \Omega_0 \cup \Omega_{v0}, \\ \mathbf{H} &= -\nabla_0 \varphi, \\ \mathbf{B} &= -\frac{\partial \Psi}{\partial \mathbf{H}} & \text{in } \Omega_0, \\ \mathbf{B} &= -\frac{\partial \Psi_{Maxw}}{\partial \mathbf{H}} & \text{in } \Omega_{v0}, \\ \llbracket \mathbf{B} \rrbracket \cdot \mathbf{N} &= 0 & \text{on } \partial\Omega_0 \cap \partial\Omega_{v0}, \\ \varphi &= \varphi^* & \text{on } \partial\Omega_{v0}\varphi. \end{aligned}$$

Mechanics: $\phi \in \mathbb{V}_s^\phi$

$$\begin{aligned} \text{DIV } \mathbf{P} &= \mathbf{F}_0 & \text{in } \Omega_0, \\ \mathbf{F} &= \nabla_0 \phi, \\ \mathbf{P} &= \frac{\partial \Psi}{\partial \mathbf{F}} & \text{in } \Omega_0, \\ \mathbf{P} &= \frac{\partial \Psi_{Maxw}}{\partial \mathbf{F}} & \text{in } \Omega_{v0}, \\ \llbracket \mathbf{P} \rrbracket \mathbf{N} &= \mathbf{0} & \text{on } \partial\Omega_0 \cap \partial\Omega_{v0}, \\ \phi &= \phi^* & \text{on } \partial\Omega_{v0}\phi. \end{aligned}$$

hMRE (Ω_0): physics-augmented NN in terms of invariants

$$\Psi(\mathbf{F}, \mathbf{H}, \mathbf{H}_r) = \mathbb{U}(I_1, I_2, I_3, I_4, I_5^e, I_5^{er}, I_6; \mathcal{W})$$

$$\begin{aligned} I_1 &= \mathbf{F} : \mathbf{F}, & I_2 &= \text{Cof} \mathbf{F} : \text{Cof} \mathbf{F}, & I_3 &= \det \mathbf{F}, \\ I_4 &= \mathbf{F} \mathbf{H}_r \cdot \mathbf{F} \mathbf{H}_r, & I_6 &= \text{Cof} \mathbf{F} \mathbf{H}_r \cdot \text{Cof} \mathbf{F} \mathbf{H}_r, \\ I_5^e &= \mathbf{H} \cdot \mathbf{C}^{-1} \mathbf{H}, & I_5^{er} &= \mathbf{H} \cdot \mathbf{C}^{1/2} \mathbf{H}_r \end{aligned}$$

Vacuum (Ω_{v0}): Maxwell energy

$$\Psi_{Maxw}(\mathbf{F}, \mathbf{H}) = -\frac{\mu_0}{2} J \mathbf{C}^{-1} : (\mathbf{H} \otimes \mathbf{H}) = -\frac{\mu_0}{2} I_3 I_5^e$$

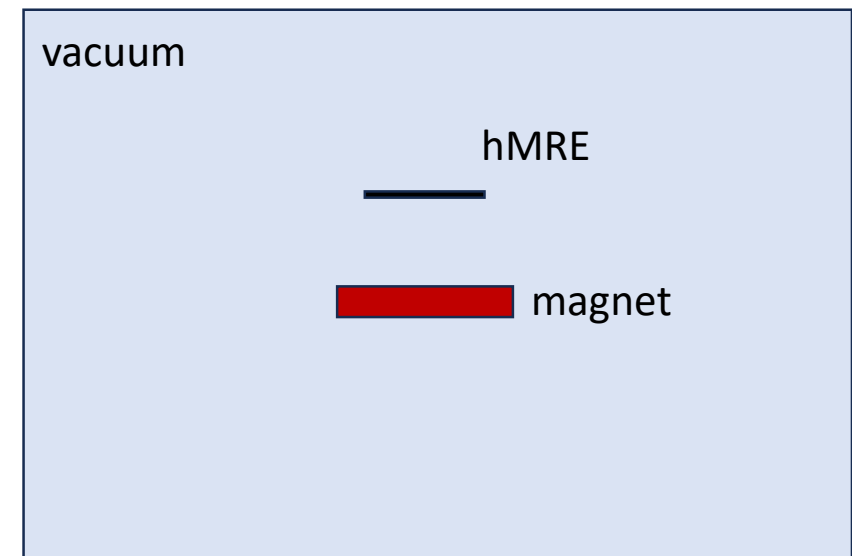
Staggered solver for magneto-mechanics in vacuum

The numerical resolution of the coupled magneto-mechanical problem was performed via a custom finite element implementation using a **staggered scheme**.

Step 1: Magnetostatic problem (scalar potential)

Solve for $\varphi \in \mathbb{V}^\varphi$ in $\Omega_0 \cup \Omega_{v0}$:

$$\begin{aligned}\text{DIV } \mathbf{B}(\mathbf{X}, \mathbf{F}, \mathbf{H}) &= 0, \\ \mathbf{H} &= -\nabla_0 \varphi, \\ \mathbf{B} &= -\frac{\partial \Psi}{\partial \mathbf{H}} \quad \text{in } \Omega_0, \\ \mathbf{B} &= -\frac{\partial \Psi_{Maxw}}{\partial \mathbf{H}} \quad \text{in } \Omega_{v0}, \\ \varphi &= \varphi^* \quad \text{on } \partial\Omega_{v0\varphi}.\end{aligned}$$



The scalar magnetic potential φ was computed in the entire domain (hMRE and vacuum). Continuity and boundary conditions were imposed across interfaces and on the boundary.

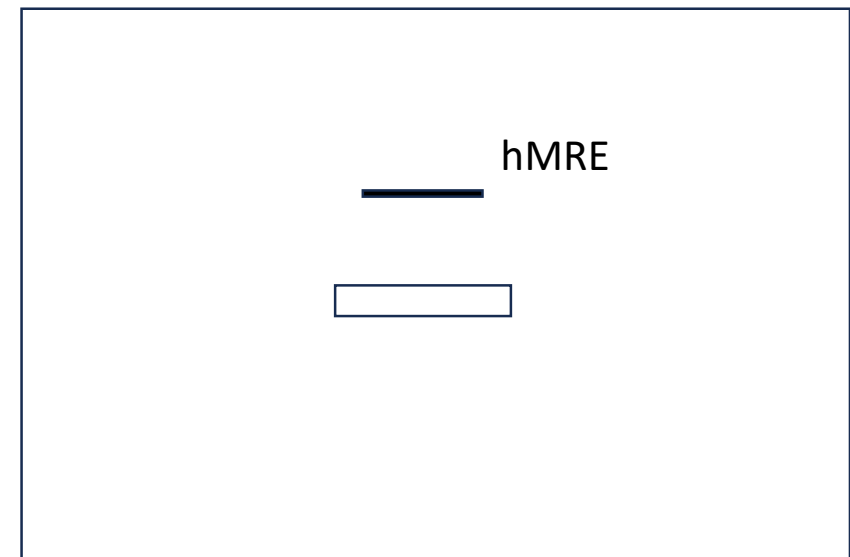
Staggered solver for magneto-mechanics in vacuum

The numerical resolution of the coupled magneto-mechanical problem was performed via a custom finite element implementation using a **staggered scheme**.

Step 2: Mechanical problem in the solid

Using $\mathbf{H}$ from Step 1, solve for $\phi \in \mathbb{V}_s^\phi$ in Ω_0 :

$$\begin{aligned}\text{DIV } \mathbf{P}(\mathbf{X}, \mathbf{F}, \mathbf{H}) &= \mathbf{F}_0, \\ \mathbf{F} &= \nabla_0 \phi, \\ \phi &= \phi^* \quad \text{on } \partial\Omega_0\phi, \\ \llbracket \mathbf{P} \rrbracket \mathbf{N} &= \mathbf{0} \quad \text{on } \partial\Omega_0 \cap \partial\Omega_{v0}, \\ \mathbf{P} &= \frac{\partial \Psi}{\partial \mathbf{F}} \quad \text{in } \Omega_0.\end{aligned}$$



The deformation mapping ϕ was computed in the solid domain by solving the equilibrium equation. Dirichlet and Neumann boundary conditions were applied as appropriate, and $\mathbf{H}$ is taken from Step 1.

Staggered solver for magneto-mechanics in vacuum

The numerical resolution of the coupled magneto-mechanical problem was performed via a custom finite element implementation using a **staggered scheme**.

Step 3: Mechanical problem in the vacuum

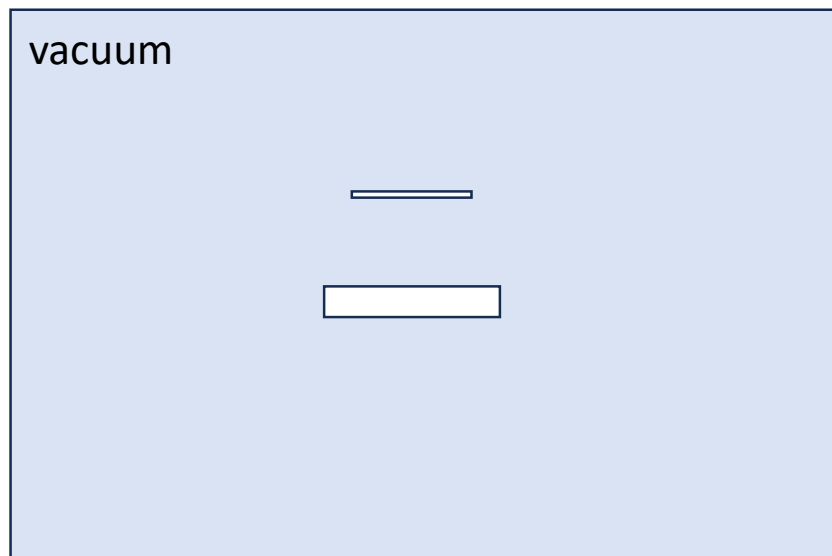
Enforce continuity with ϕ and solve for $\phi_v \in \mathbb{V}_v^\phi$ in Ω_{v0} :

$$\text{DIV} \left(\frac{\partial \Psi_{vm}}{\partial \mathbf{F}}(\mathbf{F}) \right) = \mathbf{0},$$

$$\mathbf{F} = \nabla_0 \phi_v,$$

$$\phi_v = \mathbf{0} \quad \text{on } \partial\Omega_{v0}\phi,$$

$$\phi_v = \phi \quad \text{on } \partial\Omega_0 \cap \partial\Omega_{v0}.$$



A nonlinear elastic problem was solved in the vacuum region using a potential ϕ_v , enforcing continuity with the solid deformation and applying clamped boundary conditions on its boundary.

1) Fictitious energy model

$$\Psi_{vm}(\mathbf{F}) = \frac{\mu_1}{2p_1 3^{p_1-1}} \left(\hat{I}_1 \right)^{p_1} + \frac{\mu_2}{2p_2 3^{p_2-1}} \left(\hat{I}_2 \right)^{p_2} + \frac{\kappa}{2} \left(J^{p_3} + J^{-p_3} \right)$$

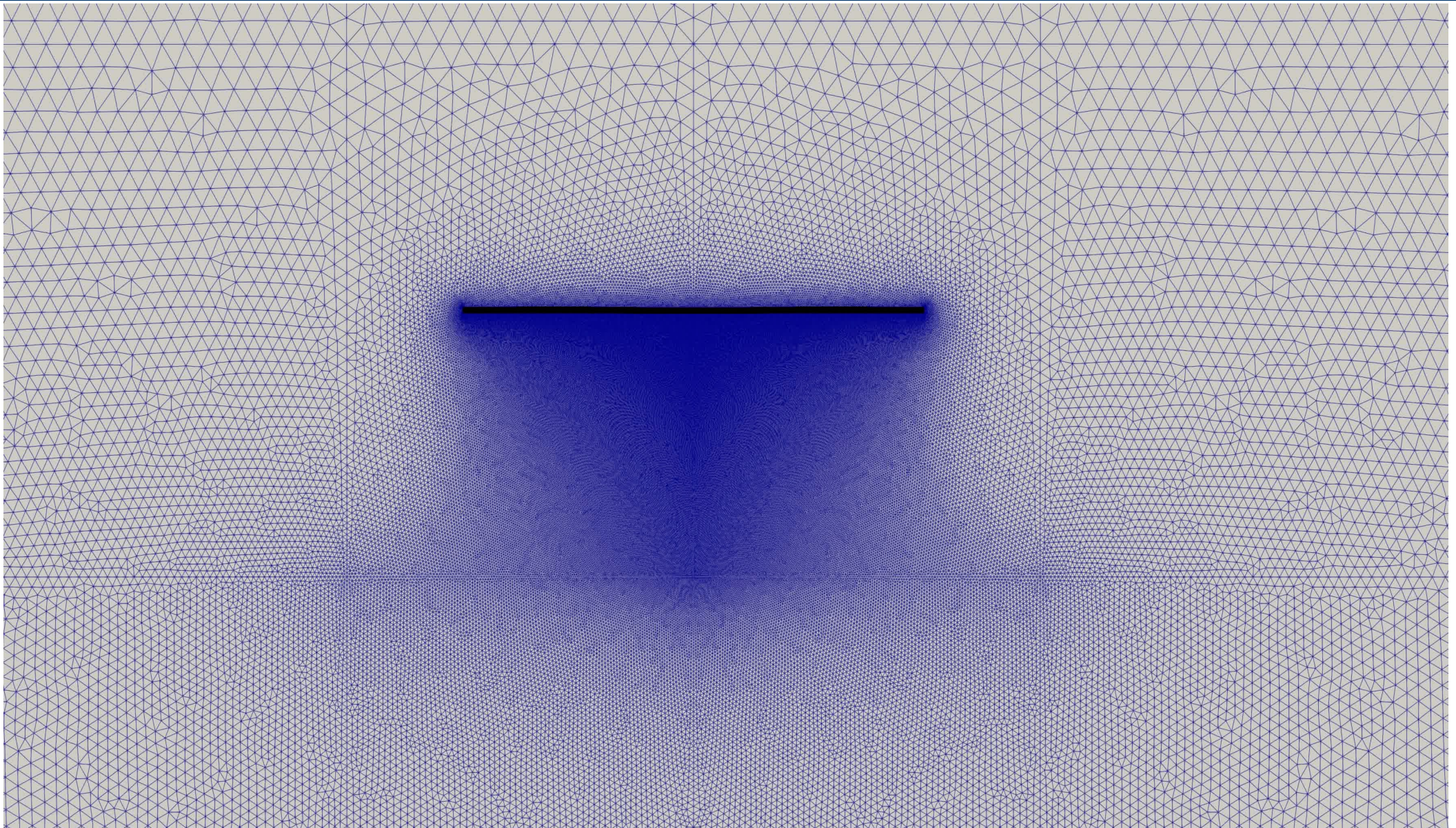
$\{\mu_1, \mu_2, \kappa\}$ represent fictitious elastic properties of Ω_{v0}

$\{p_1, p_2, p_3\}$ penalise in a nonlinear fashion the isochoric invariants and the volumetric invariant J

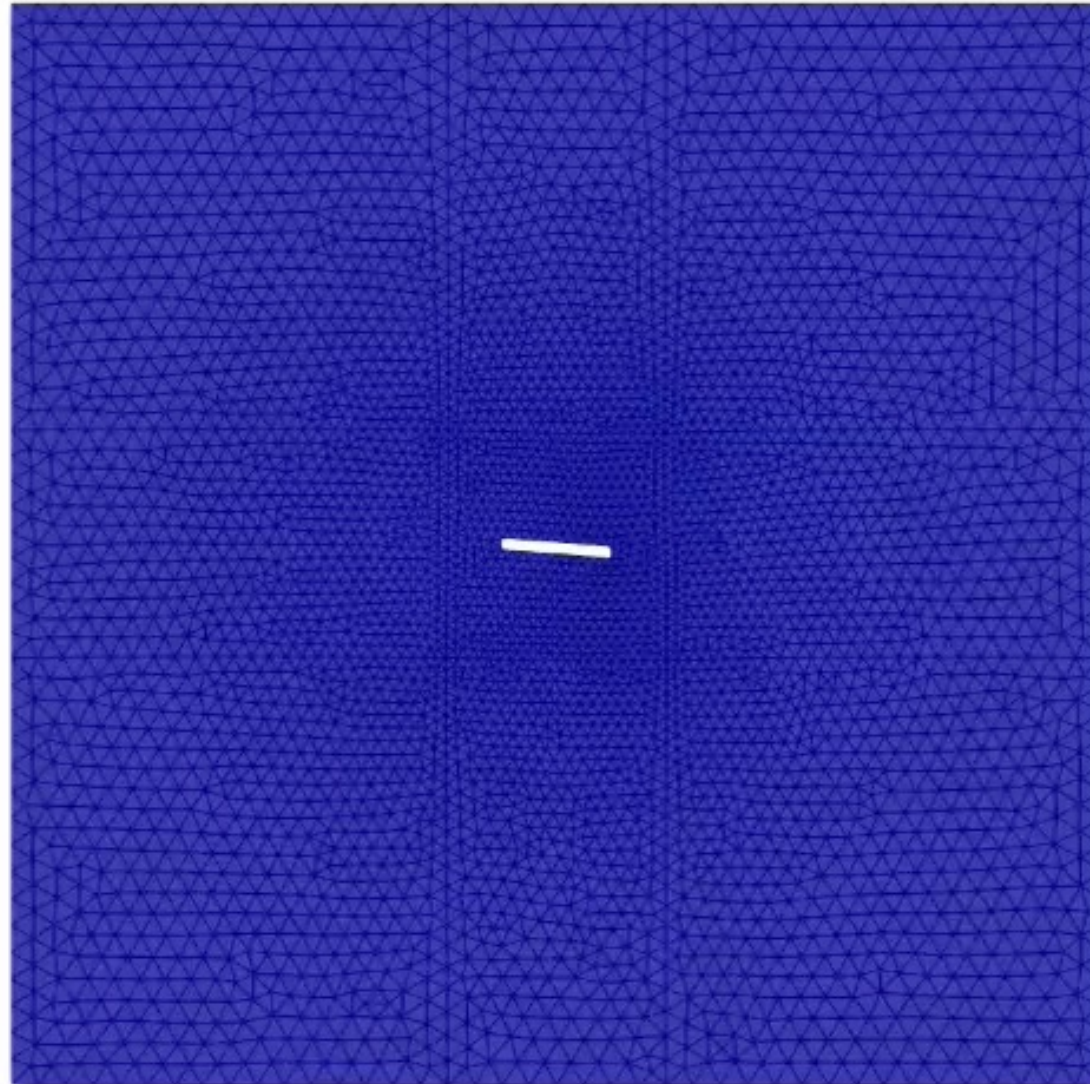
2) Stabilization of the elasticity tensor

$$DW_{\phi_v}(\phi_v|_k)[\Delta\phi_v] = \int_{\Omega_0} \nabla_0 \mathbf{v} : \mathbf{C}_{\text{reg}}(\mathbf{F}) : \nabla_0 \Delta\phi_v dV$$
$$\mathbf{C}_{\text{reg}} = \sum_{i=1}^9 \max(\lambda_i, 0) \mathbf{V}_i \otimes \mathbf{V}_i,$$

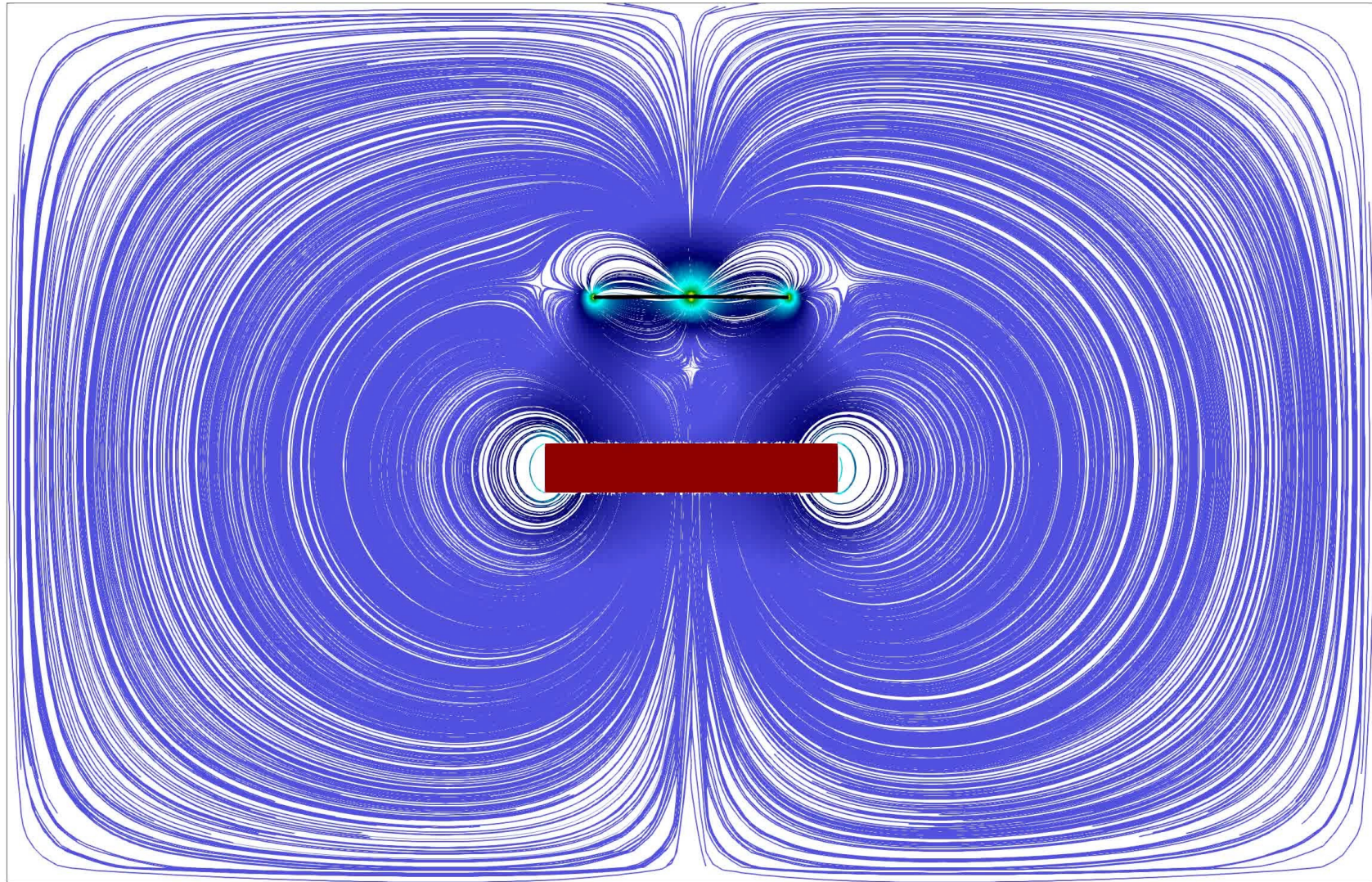
Mesh movement stabilization



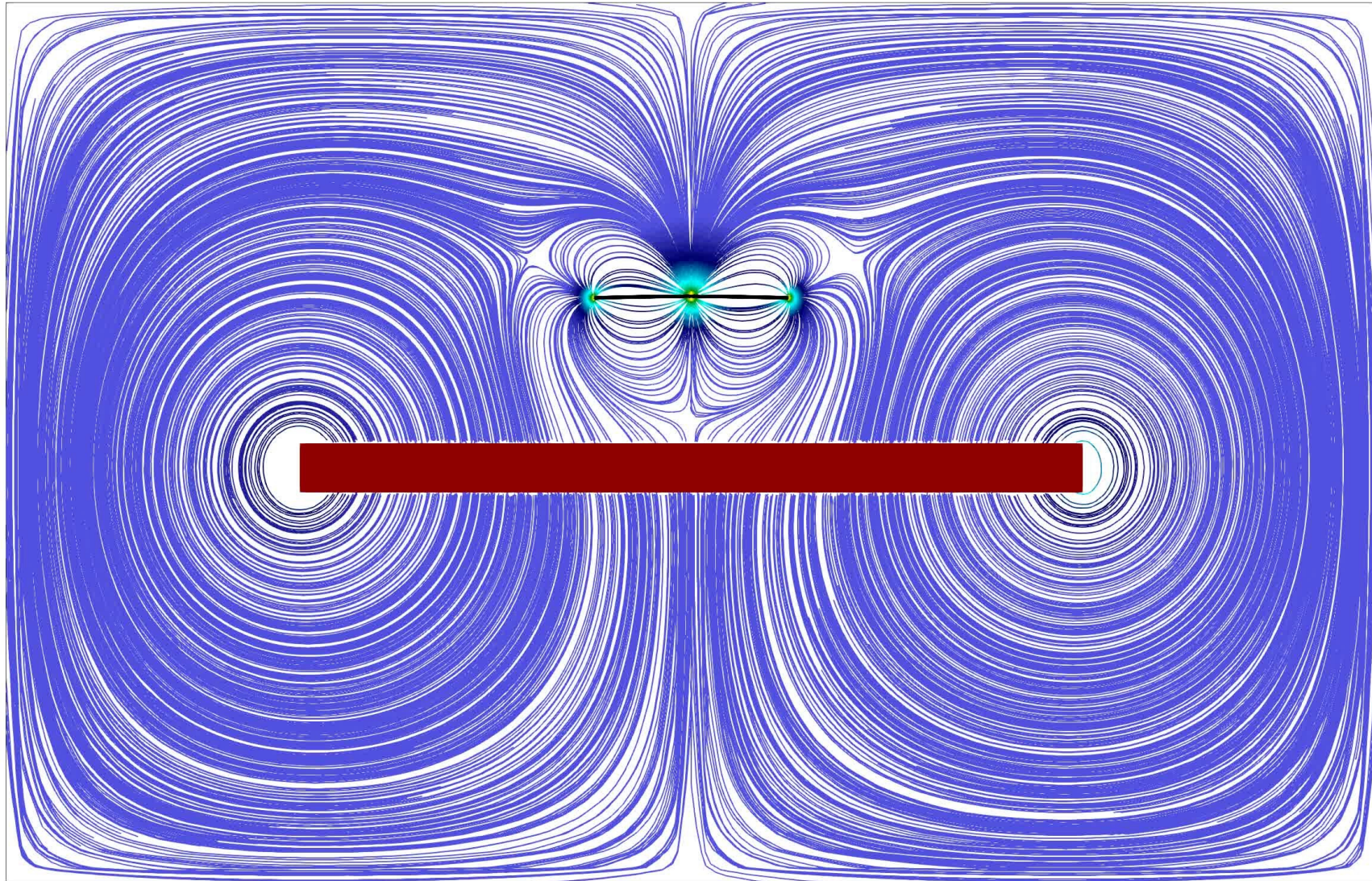
Mesh movement stabilization



Staggered solver for magneto-mechanics in vacuum

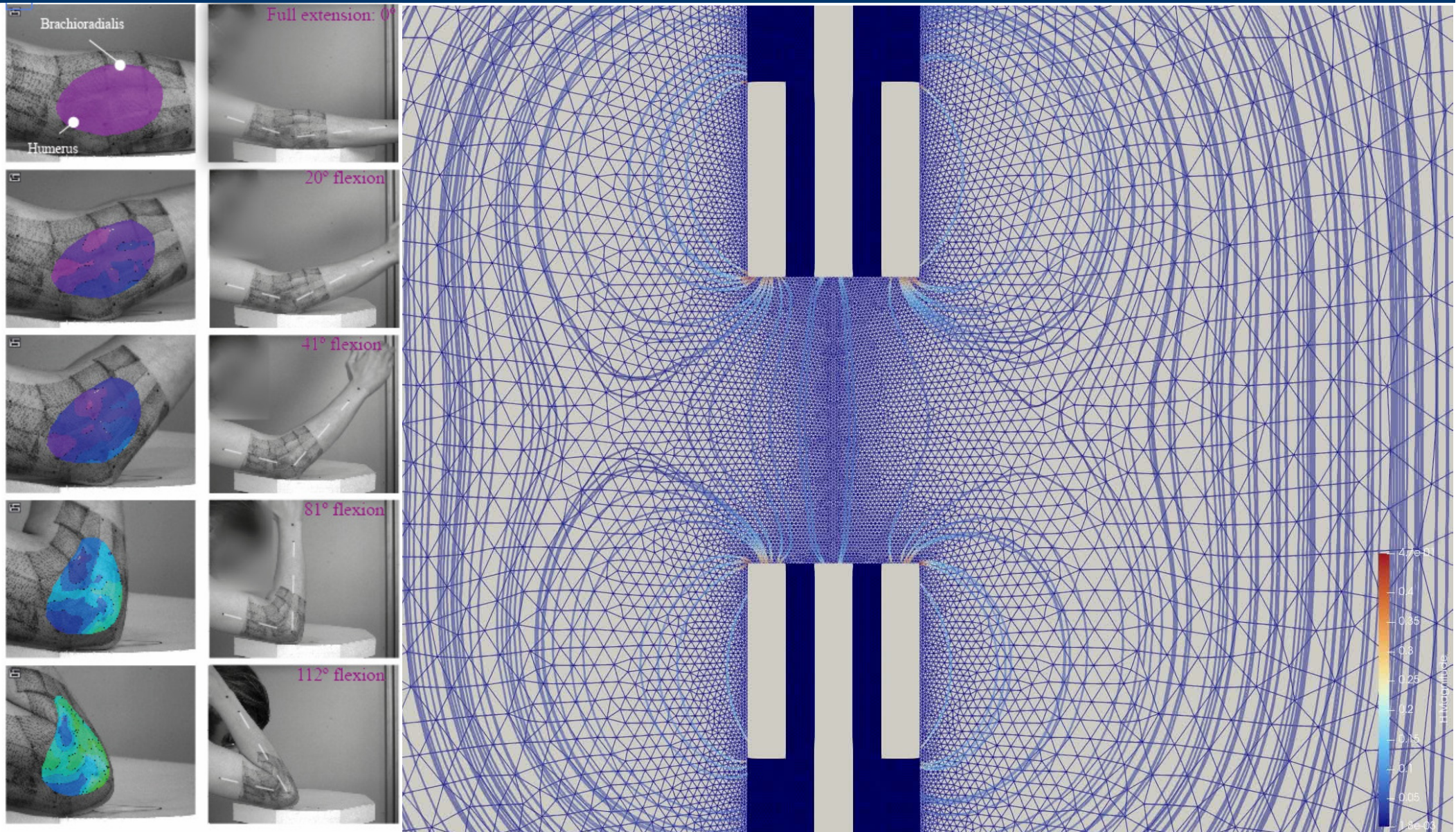


Examples



Sensing application hMREs

f SéNeCa⁽⁺⁾
Agencia de Ciencia y Tecnología
Región de Murcia



OUTLINE

Contents



01 Our initial contact on TO in hyperelasticity

02 Previous work on Topology Optimization of active materials

03 Residual Mechanical Anisotropy in ultra-soft hMREs

04 Computational platform to Program Magneto-Mechanical Actuation

05 Multimaterial and Topology optimization algorithm

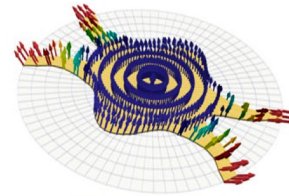
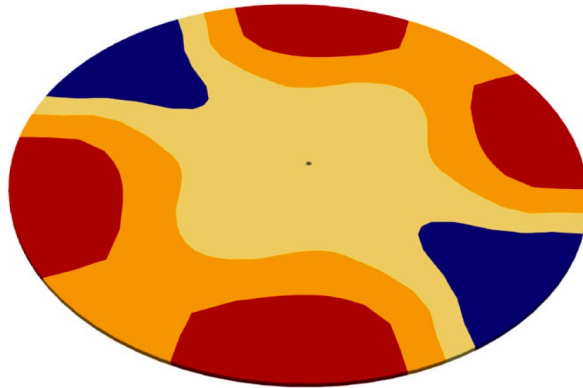
06 Julia implementation - HyperFEM



HyperTop

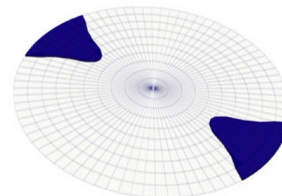
Optimization framework

A



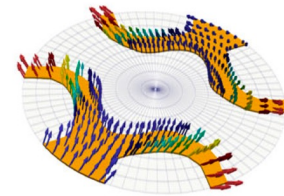
Material 1

⊕



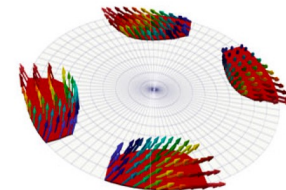
Material 2

⊕



Material 3

⊕



Material 4

Design variables

- Geometry
- Materials dataset
- Magnetic stimuli
- Target actuation

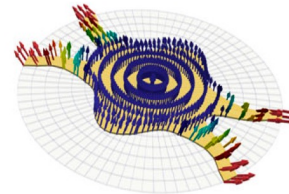
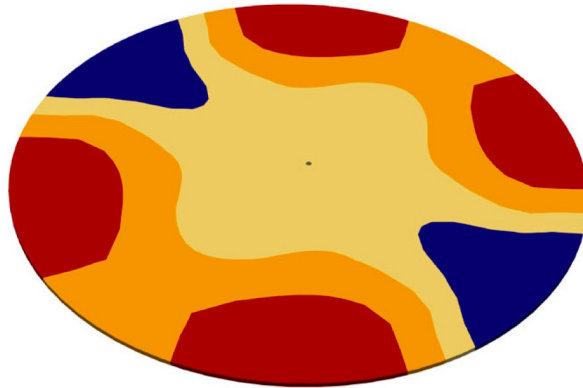
$$\mathbf{H}_r = H_r \mathbf{N}$$

$$\Psi(\xi, c, \mathbf{N}, H_{app}, \mathbf{F}) = (1 - \xi^{p_\xi}) \Psi_p(\mathbf{F}) + \xi^{p_\xi} \Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F})$$

$$\Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F}) = (1 - c^{p_c}) \Psi|_{c=0.2}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app}) + c^{p_c} \Psi|_{c=0.3}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app})$$

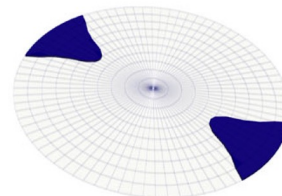
Optimization framework

A



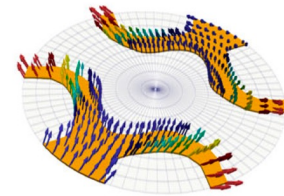
Material 1

⊕



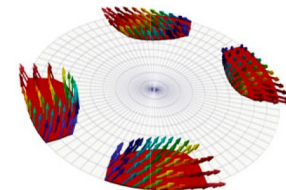
Material 2

⊕



Material 3

⊕



Material 4

Design variables

- Geometry
- Materials dataset
- Magnetic stimuli
- Target actuation

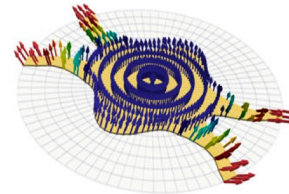
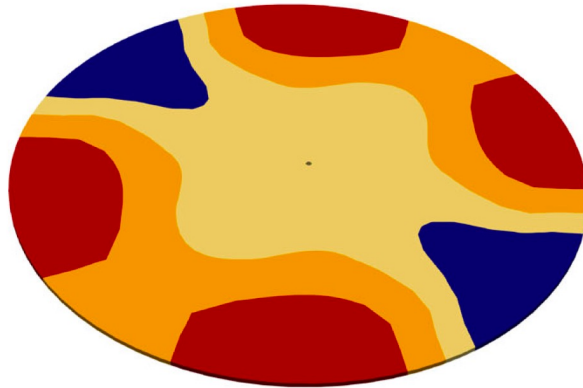
$$\mathbf{H}_r = H_r \mathbf{N}$$

$$\Psi(\xi, c, \mathbf{N}, H_{app}, \mathbf{F}) = (1 - \xi^{p_\xi}) \Psi_p(\mathbf{F}) + \xi^{p_\xi} \Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F})$$

$$\Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F}) = (1 - c^{p_c}) \Psi|_{c=0.2}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app}) + c^{p_c} \Psi|_{c=0.3}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app})$$

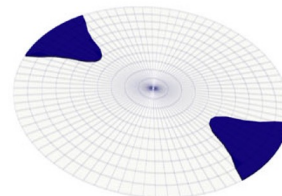
Optimization framework

A



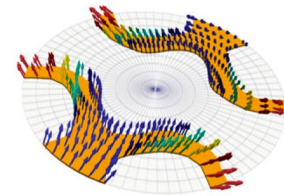
Material 1

⊕



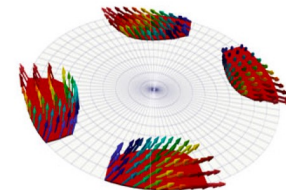
Material 2

⊕



Material 3

⊕



Material 4

Design variables

- Geometry
- Materials dataset
- Magnetic stimuli
- Target actuation

$$\mathbf{H}_r = H_r \mathbf{N}$$

$$\Psi(\xi, c, \mathbf{N}, H_{app}, \mathbf{F}) = (1 - \xi^{p_\xi}) \Psi_p(\mathbf{F}) + \xi^{p_\xi} \Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F})$$

$$\Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F}) = (1 - c^{p_c}) \Psi|_{c=0.2}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app}) + c^{p_c} \Psi|_{c=0.3}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app})$$

Optimization formulation

Minimize in $(\xi, c, \mathbf{N}, H_r) : \mathcal{J}(\xi, c, \mathbf{N}, H_{app}) := \int_{\Omega_0} \|\phi(\xi, c, \mathbf{N}, H_{app}) - \phi_d\|^2 dV$

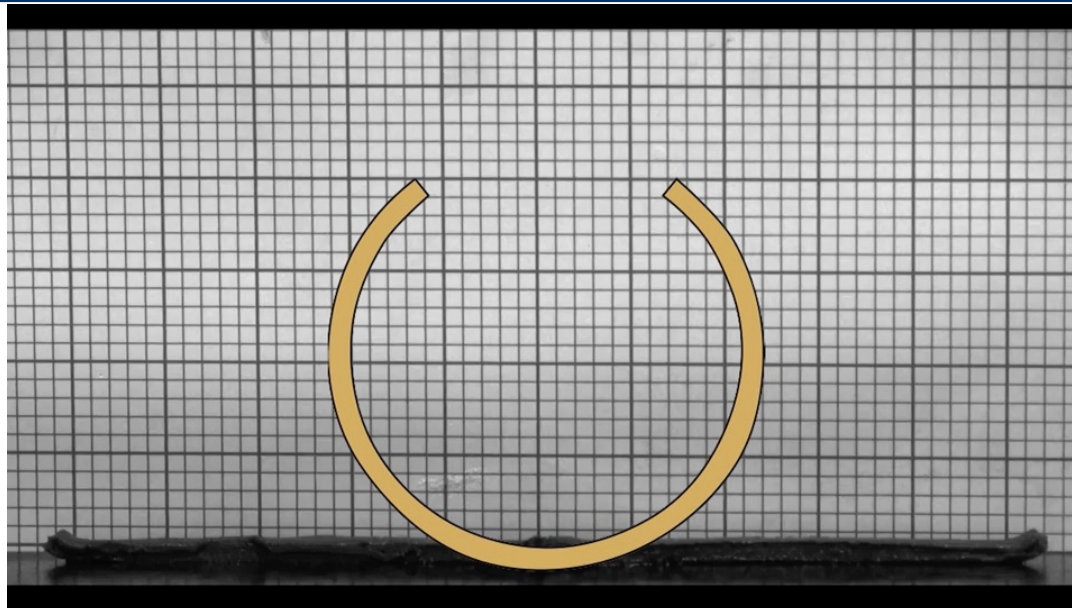
$$\text{subject to } \left\{ \begin{array}{l} \left\{ \begin{array}{ll} \text{DIV } \mathbf{P}(\xi, c, \mathbf{N}, H_{app}, \mathbf{F}) = \mathbf{F}_0; & \text{in } \Omega_0; \\ \mathbf{F} = \nabla_0 \phi \Rightarrow \text{CURL } \mathbf{F} = \mathbf{0}; & \text{in } \Omega_0 \\ \phi = \phi^*; & \text{on } \partial\Omega_0 \phi \end{array} \right. \\ \left\{ \begin{array}{l} \Psi(\xi, c, \mathbf{N}, H_{app}, \mathbf{F}) = (1 - \xi^{p_\xi}) \Psi_p(\mathbf{F}) + \xi^{p_\xi} \Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F}) \\ \Psi_a(c, \mathbf{N}, H_{app}, \mathbf{F}) = \left\{ \begin{array}{l} (1 - c^{p_c}) \Psi|_{c=0.2}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app}) \\ + c^{p_c} \Psi|_{c=0.3}(\mathbf{F}, \mathbf{H}_r(\mathbf{N}), H_{app}) \end{array} \right. \\ \mathbf{P}(\xi, c, \mathbf{N}, H_{app}, \mathbf{F}) = \frac{\partial \Psi(\xi, c, \mathbf{N}, \mathbf{F}, H_{app})}{\partial \mathbf{F}} \end{array} \right. \\ \xi \in [0, 1] \\ c \in [0, 1] \\ \mathbf{N} \in \mathcal{S}^2 := \{\mathbf{N} \in \mathbb{R}^3 : \|\mathbf{N}\| = 1\} \\ H_{app} \in [0, H_{app, max}] \end{array} \right.$$

Optimized Dowsil C-shape

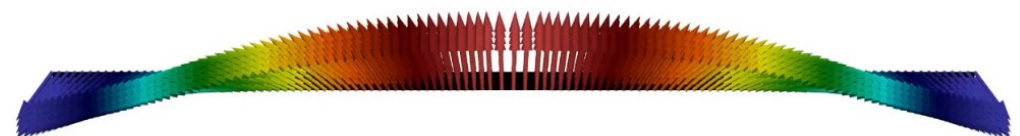
f SéNeCa⁽⁺⁾
Agencia de Ciencia y Tecnología
Región de Murcia



Optimized Dowsil C-shape



But does not work for ultra-soft hMREs

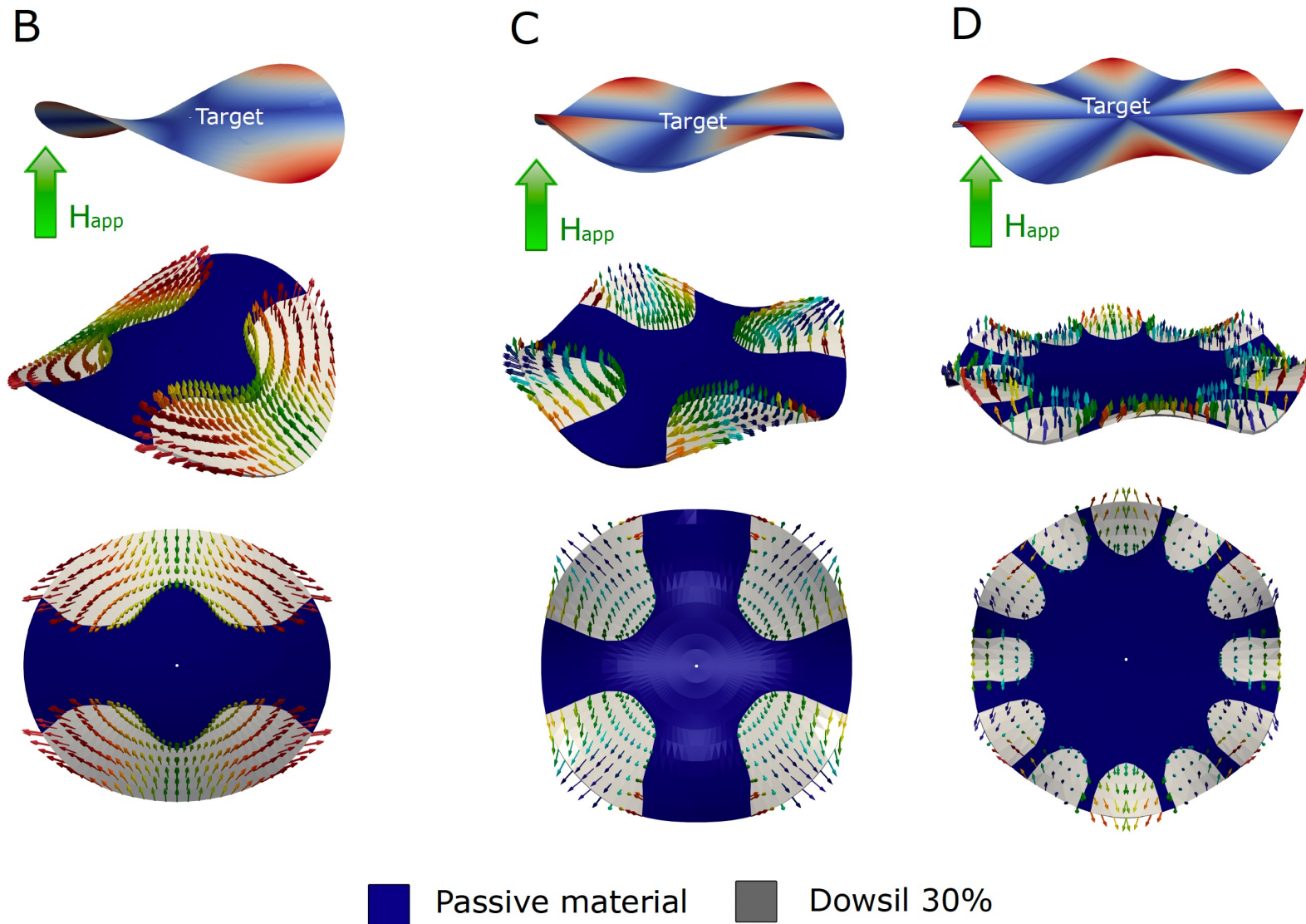


Topology and material optimization

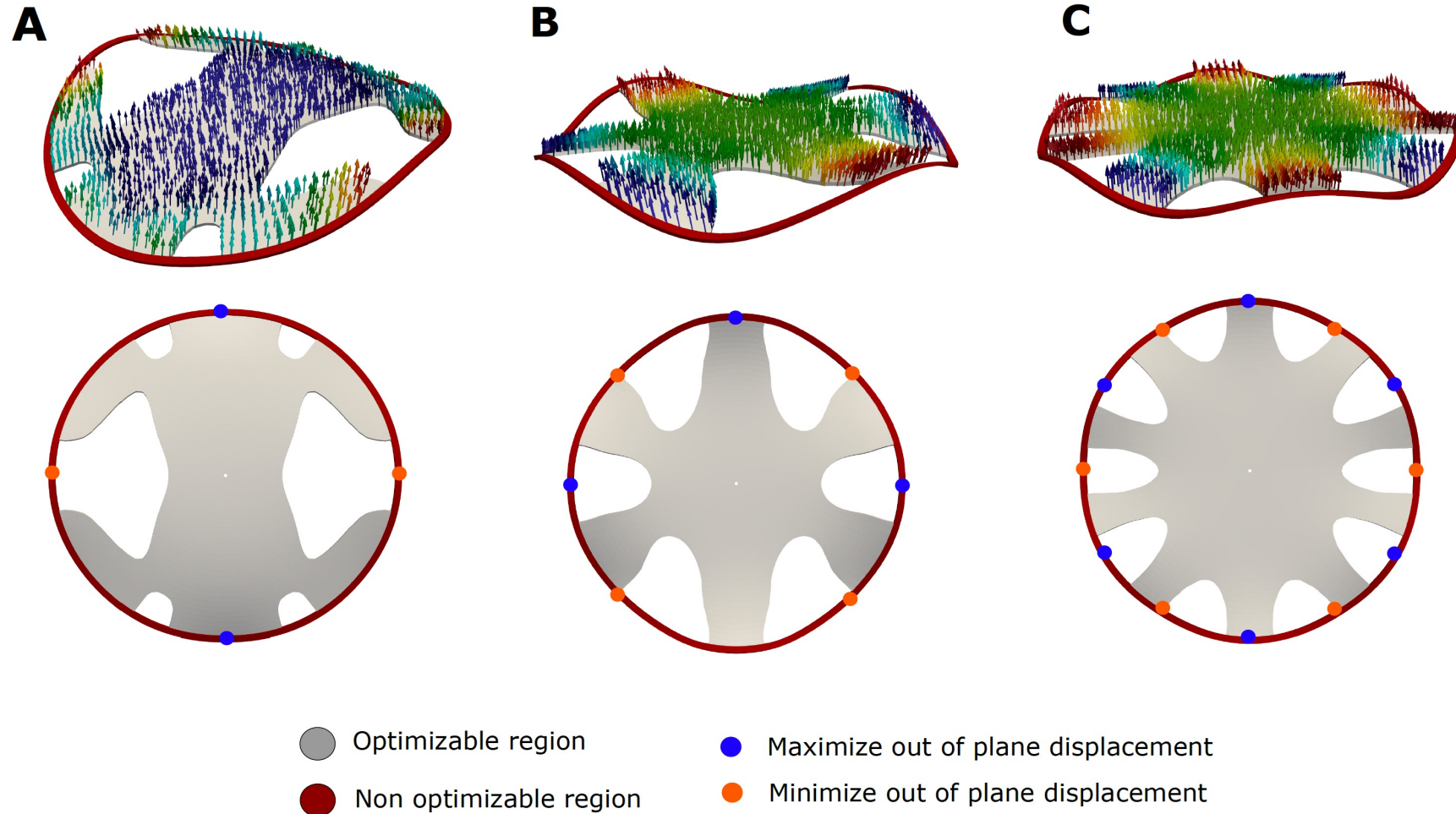
f SéNeCa⁽⁺⁾
Agencia de Ciencia y Tecnología
Región de Murcia



Staggered Approach



Staggered Approach



C. Pérez-García, R. Ortigosa, J. Martínez-Frutos, and D. García-González, *Topology and material optimization in ultra-soft magnetoactive structures: making advantage of residual anisotropies*. Adv. Mater. (2025)

OUTLINE

Contents



01 Our initial contact on TO in hyperelasticity

02 Previous work on Topology Optimization of active materials

03 Residual Mechanical Anisotropy in ultra-soft hMREs

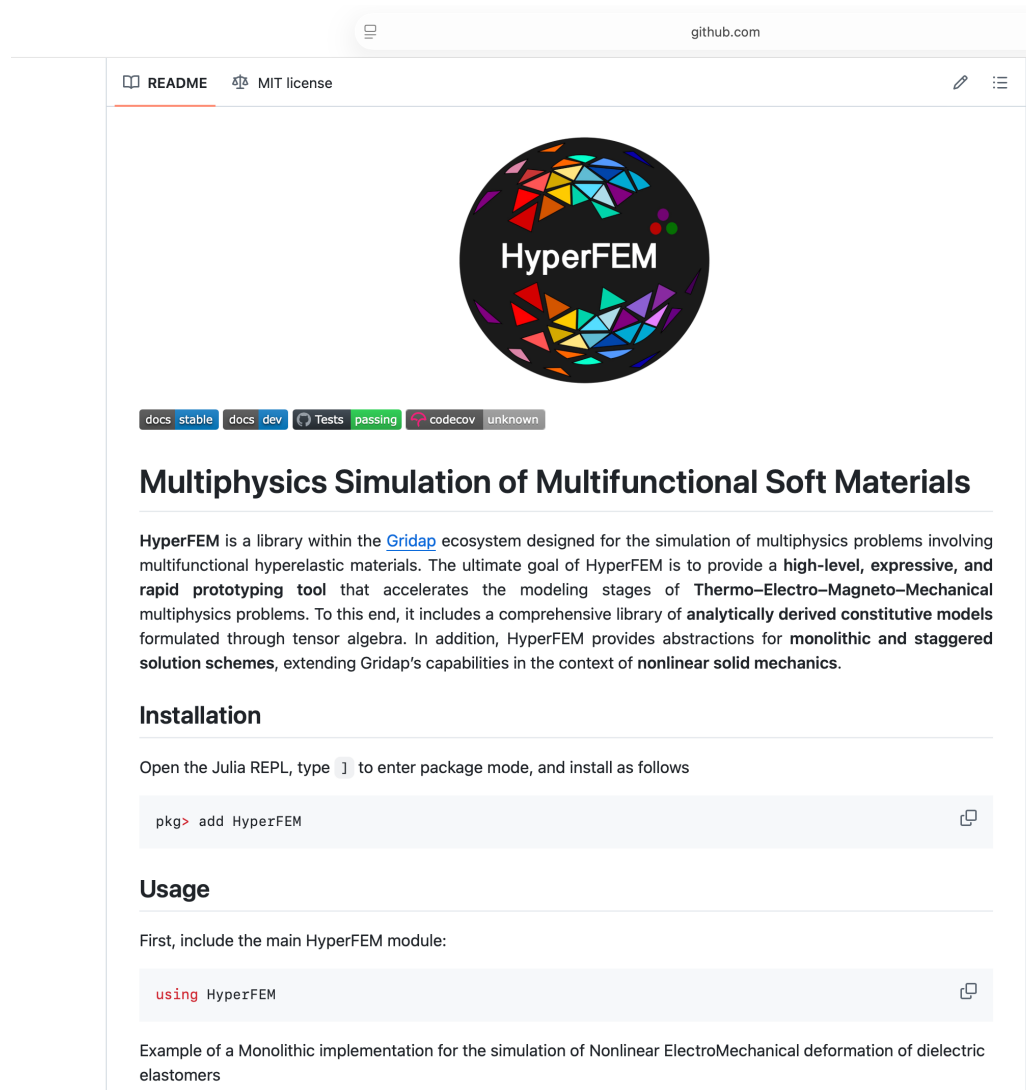
04 Computational platform to Program Magneto-Mechanical Actuation

05 Multimaterial and Topology optimization algorithm

06 Julia implementation - HyperFEM



HyperFEM (julia)



The screenshot shows the GitHub repository for HyperFEM. At the top, there's a navigation bar with 'README' and 'MIT license'. The main content area features the HyperFEM logo, a circular emblem with a colorful geometric pattern. Below the logo, there are badges for 'docs stable', 'docs dev', 'Tests passing', 'codecov', and 'unknown'. The title 'Multiphysics Simulation of Multifunctional Soft Materials' is prominently displayed. The description states that HyperFEM is a library within the Gridap ecosystem for simulating multiphysics problems involving hyperelastic materials. It highlights its goal to provide a high-level, expressive, and rapid prototyping tool. The 'Installation' section instructs users to open the Julia REPL, enter package mode, and install the package. The 'Usage' section shows how to include the main HyperFEM module in a Julia script. At the bottom, an example of a Monolithic implementation for simulating Nonlinear ElectroMechanical deformation of dielectric elastomers is mentioned.

HyperFEM.jl · official Julia package

Registered in the Julia General registry

<https://github.com/MultiSimOLab/HyperFEM.jl>

Install from the Julia REPL (press `]` to enter package mode):

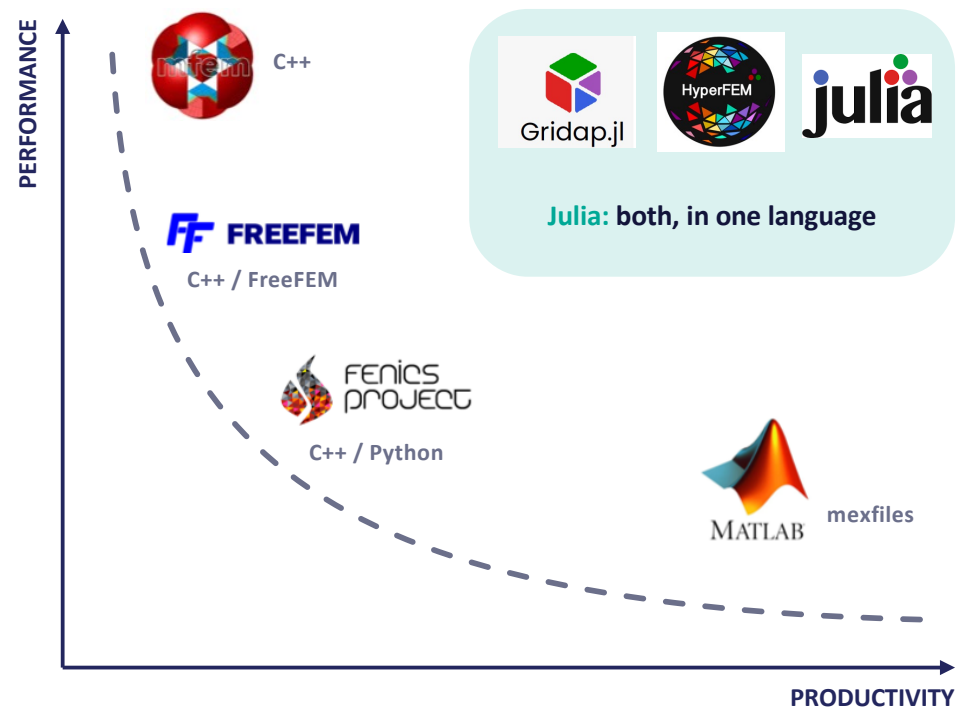
```
pkg> add HyperFEM
```



QR: HyperFEM.jl · Code of this work, Adv. Mater. (2025):
https://github.com/MultiSimOLab/Adv_Mater_Ultra_Soft_Magnet

WHY JULIA

A balance between productivity and performance



● **Productivity** · dynamic languages

Python, MATLAB... expressive, interactive, no compilation step, no environment to set up. Best for chasing math bugs.

● **Performance** · static languages

C++, Fortran... compilers generate highly optimised code.

Two-language barrier

A Python front-end over a C++ core (FEniCS, FreeFEM) works — until a change has to reach the static library.

Julia removes it: prototype and production are the same code.

WORKFLOW

Design

a new method

Analyse

the method

Implement

rapid prototyping

Exploit

in applications

JULIA LANG

“As fast as C, as expressive as Python”



High-level, high-performance language designed for numerical and scientific computing · MIT, 2011–

Productive

Dynamic language, interactive like Python or MATLAB

Performant

Advanced type inference + just-in-time (JIT) compilation via LLVM

The Julia universe

Develop

- Interactive REPL and built-in package manager
- Unit testing and profiling tools
- GitHub-native: package registration, automatic docs

Scale

- LLVM JIT for fast numerical kernels
- Distributed memory and multithreading, natively
- First-class GPU support, one programming model

Abstract

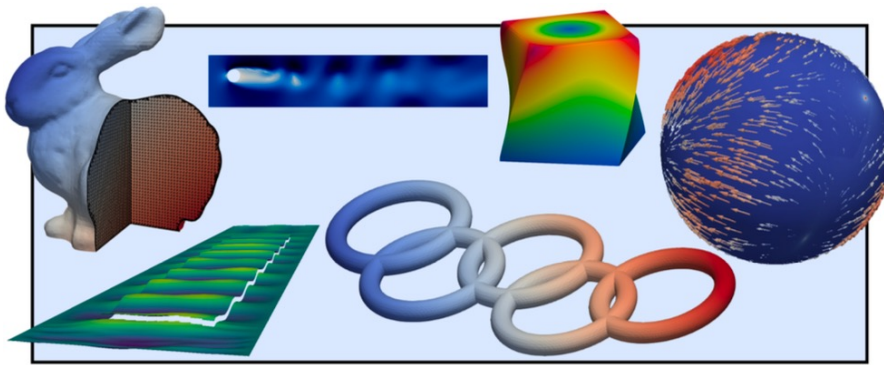
- Multiple dispatch + strong type system
- Expressive, extensible numerical abstractions
- Reproducible environments (Project / Manifest)

Connect

- Rich ecosystem: linear algebra, sparse solvers
- Calls C, C++, Python and Fortran at no penalty

THE FOUNDATION

Gridap: a general-purpose FE library in Julia



S. Badia, F. Verdugo. Gridap: an extensible finite element toolbox in Julia. *J. Open Source Softw.* 5(52), 2520 (2020). [doi:10.21105/joss.02520](https://doi.org/10.21105/joss.02520)

github.com/gridap/Gridap.jl

$$\int_{\Omega} \nabla u \cdot \nabla v \, d\Omega = \int_{\Omega} f v \, d\Omega$$

```
# Poisson problem, complete script
using Gridap
u(x) = x[1] + x[2]; f(x) = 0
model = CartesianDiscreteModel((0,1,0,1), (4,4))
Ω = Interior(model); dΩ = Measure(Ω, 2)
refe = ReferenceFE(lagrangian, Float64, 1)
V = TestFESpace(Ω, refe); U = TrialFESpace(V, u)
a(u, v) = ∫( ∇(u) · ∇(v) ) dΩ
l(v) = ∫( v * f ) dΩ
uh = solve(AffineFEOperator(a, l, U, V))
writevtk(Ω, "results", cellfields=["uh" => uh])
```

FEniCS · Python front-end over C++

```
a = dot(grad(u), grad(v)) * dx
L = f * v * dx
```

Gridap · pure Julia, reads like the math

```
a(u, v) = ∫( ∇(u) · ∇(v) ) dΩ
l(v) = ∫( v * f ) dΩ
```

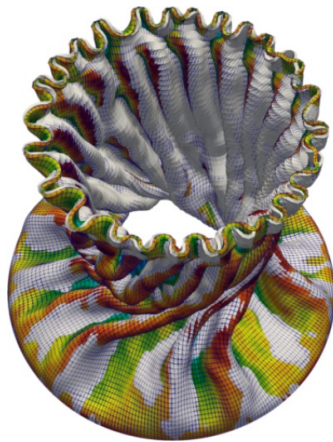
OPEN-SOURCE ECOSYSTEM

MultiSim0 Lab codes

github.com/MultiSimOLab



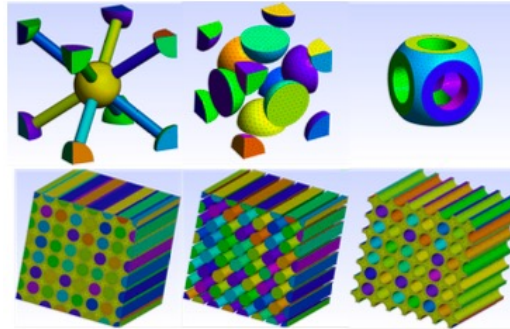
THIS TALK



HyperFEM

Multiphysics finite elements for nonlinear hyperelastic materials: electro, magneto, thermo and dissipative couplings.

Registered Julia package



RVEs

Representative volume elements for microstructured materials: lattices, particle inclusions and fibre-reinforced architectures.

UNDER CONSTRUCTION



HyperShape

Shape and topology optimization problems.

A thin, expressive layer on top of Gridap

L^AT_EX paper

$$\text{res}((\mathbf{u}, \varphi), (\mathbf{v}, \delta\varphi)) = \int_{\Omega} \nabla_0 \mathbf{v} : \frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{E}) \, d\Omega - \int_{\Omega} \nabla_0 \delta\varphi \cdot \frac{\partial \Psi}{\partial \mathbf{E}}(\mathbf{F}, \mathbf{E}) \, d\Omega$$

$$\begin{aligned} \text{jac}((\mathbf{u}, \varphi), (\Delta \mathbf{u}, \Delta \varphi), (\mathbf{v}, \delta\varphi)) &= \int_{\Omega} \nabla_0 \mathbf{v} : \frac{\partial^2 \Psi}{\partial \mathbf{F} \partial \mathbf{F}} : \nabla_0 \Delta \mathbf{u} \, d\Omega + \int_{\Omega} \nabla_0 \delta\varphi \cdot \frac{\partial^2 \Psi}{\partial \mathbf{E} \partial \mathbf{E}} \nabla_0 \Delta \varphi \, d\Omega \\ &\quad - \int_{\Omega} \nabla_0 \Delta \varphi \cdot \frac{\partial^2 \Psi}{\partial \mathbf{E} \partial \mathbf{F}} : \nabla_0 \mathbf{v} \, d\Omega - \int_{\Omega} \nabla_0 \delta\varphi \cdot \frac{\partial^2 \Psi}{\partial \mathbf{E} \partial \mathbf{F}} : \nabla_0 \Delta \mathbf{u} \, d\Omega \end{aligned}$$

with $\mathbf{F} = \mathbf{I} + \nabla_0 \mathbf{u}$, $\mathbf{E} = -\nabla_0 \varphi$

$$\begin{aligned} \text{res}((\mathbf{u}, \varphi), (\mathbf{v}, \mathbf{v}\varphi)) &= \int (\nabla(\mathbf{v})' \odot (\partial\Psi_{\mathbf{F}} \circ (\mathbf{F} \circ \nabla(\mathbf{u})', \mathbf{E} \circ \nabla(\varphi)))) \, d\Omega - \\ &\quad \int (\nabla(\mathbf{v}\varphi)' \cdot (\partial\Psi_{\mathbf{E}} \circ (\mathbf{F} \circ \nabla(\mathbf{u})', \mathbf{E} \circ \nabla(\varphi)))) \, d\Omega \end{aligned}$$

$$\begin{aligned} \text{jac}((\mathbf{u}, \varphi), (\mathbf{du}, \mathbf{d}\varphi), (\mathbf{v}, \mathbf{v}\varphi)) &= \int (\nabla(\mathbf{v})' \odot ((\partial\Psi_{\mathbf{F}\mathbf{F}} \circ (\mathbf{F} \circ \nabla(\mathbf{u})', \mathbf{E} \circ \nabla(\varphi))) \odot \nabla(\mathbf{du})')) \, d\Omega + \\ &\quad \int (\nabla(\mathbf{v}\varphi)' \cdot ((\partial\Psi_{\mathbf{E}\mathbf{E}} \circ (\mathbf{F} \circ \nabla(\mathbf{u})', \mathbf{E} \circ \nabla(\varphi))) \cdot \nabla(\mathbf{d}\varphi))) \, d\Omega - \\ &\quad \int (\nabla(\mathbf{d}\varphi)' \cdot ((\partial\Psi_{\mathbf{E}\mathbf{F}} \circ (\mathbf{F} \circ \nabla(\mathbf{u})', \mathbf{E} \circ \nabla(\varphi))) \odot \nabla(\mathbf{v}))) \, d\Omega - \\ &\quad \int (\nabla(\mathbf{v}\varphi)' \cdot ((\partial\Psi_{\mathbf{F}\mathbf{E}} \circ (\mathbf{F} \circ \nabla(\mathbf{u})', \mathbf{E} \circ \nabla(\varphi))) \odot \nabla(\mathbf{du}))) \, d\Omega \end{aligned}$$

</> Code HyperFEM

WHAT IT IS

A thin, expressive layer on top of Gridap

Computational models & solvers

Static, dynamic and linear drivers · monolithic and staggered · Newton with line search

Weak forms

Residuals and Jacobians assembled from the energy derivatives — per physics or fully coupled

Physical models

≈40 constitutive models: mechano, electro, magneto, thermo, visco — and their couplings

Tensor algebra

Unicode operator set for 2nd, 3rd and 4th-order tensors; exact analytical derivatives

Gridap.jl — meshes, FE spaces, assembly, VTK

Design rules

- **Energy in, physics out**
- One model = one struct
- Models compose with +
- Closed-form derivatives, no AD in the hot loop
- Weak form reads like the paper
- MPI / PETSc via package extension

CORE IDEA 1

Tensor algebra you can read off the page

Coupled tangents live in third- and fourth-order space. HyperFEM ships an operator set whose indices are written in the symbol itself, so the code line and the hand-derived expression are the same object.

$\otimes_1^2$

$a \otimes b \rightarrow 2\text{nd order}$

$\otimes_{12}^{34}$

$A \otimes B \rightarrow 4\text{th order}$

$\otimes_{13}^{24}$

index-shuffled outer

$\times_i^4$

tensor cross product

cof, H, J

kinematic invariants

$\odot$

double contraction

```
# ideal dielectric: coupled tangent  $\partial^2 \Psi / \partial E \partial F$ 
 $\partial \Psi_{EH}(F, E) = (-\varepsilon / J(F)) * ((I3 \otimes_{13}^2 H(F) * E) + (H(F)' \otimes_{12}^3 E))$ 
 $\partial \Psi_{EF}(F, E) = (\partial \Psi_{EH}(F, E) \times F) + (\partial \Psi_{EJ}(F, E) \otimes_1^2 H(F))$ 
```

CORE IDEA 2

Energy and derivatives, written once

$\Psi, \partial\Psi, \partial^2\Psi$ in closed form

Each constitutive model implements its energy and every first and second derivative explicitly, with the tensor-algebra operators. The derivation is done once, by hand, and lives inside the model.

$\partial\Psi/\partial\mathbf{F}$	first Piola stress
$\partial\Psi/\partial\mathbf{E}$	electric displacement
$\partial^2\Psi/\partial\mathbf{F}\partial\mathbf{F}$	mechanical tangent
$\partial^2\Psi/\partial\mathbf{E}\partial\mathbf{F}$	coupling tangent
$\partial^2\Psi/\partial\mathbf{E}\partial\mathbf{E}$	dielectric tangent

```
function (obj::NeoHookean3D) ()
    J(F) = det(F); H(F) = J(F)*inv(F) '
     $\Psi(F) = \mu/2*\text{tr}(F'*F) - \mu*\log(J(F)) + \lambda/2*(J(F)-1)^2$ 
     $\partial\Psi\partial J(F) = -\mu/J(F) + \lambda*(J(F)-1)$ 
     $\partial^2\Psi\partial J^2(F) = \mu/J(F)^2 + \lambda$ 
     $\partial\Psi\partial F(F) = \mu*F + \partial\Psi\partial J(F)*H(F)$ 
     $\partial\Psi\partial F\partial F(F) = \mu*I9 + \partial^2\Psi\partial J^2(F)*(H(F)\otimes H(F)) + \partial\Psi\partial J(F)*x_i^4(F)$ 
    return ( $\Psi$ ,  $\partial\Psi\partial F$ ,  $\partial\Psi\partial F\partial F$ )
end
```

Calling the model only returns those closures — nothing is differentiated at run time. Coupled models add up what their parts provide: ElectroMechModel returns (Ψ , $\partial\Psi F$, $\partial\Psi E$, $\partial\Psi FF$, $\partial\Psi EF$, $\partial\Psi EE$) without a new derivation.

CORE IDEA 2 · IN PRACTICE

Expressive construction of multiphysics models

```
# 1 · building blocks: one struct per physics
modeliso      = MooneyRivlin3D(λ=1e5, μ1=1e3, μ2=1e3)
modelfib1     = HGO_1Fiber(c1=c1[1], c2=c2[1])
modelfib2     = HGO_1Fiber(c1=c1[2], c2=c2[2])
modelfib3     = HGO_1Fiber(c1=c1[3], c2=c2[3])
modelfib4     = HGO_1Fiber(c1=c1[4], c2=c2[4])
modelelec     = IdealDielectric(ε=4.0)

# 2 · compose with + → coupled model inferred
constit_model = modeliso +
                [modelfib1 modelfib2 modelfib3 modelfib4] +
                modelelec

# 3 · kinematics per field
F, J, H = get_Kinematics(Kinematics(Mechano, Solid))
E       = get_Kinematics(Kinematics(Electro, Solid))

# 4 · energy + analytical derivatives
Ψ, ∂ΨF, ∂ΨFF = constitutive_model()
Ψ(F(∇u), E(∇φ), (M1,M2,M3,M4))
∂ΨF(F(∇u), E(∇φ), (M1,M2,M3,M4))
∂ΨFF(F(∇u), E(∇φ), (M1,M2,M3,M4))
```

 **MooneyRivlin3D**
Mechano · isotropic matrix

+

 **[HGO_1Fiber × 4]**
Mechano · anisotropic fibre families

+

 **IdealDielectric**
Electro · $\epsilon = 4.0$

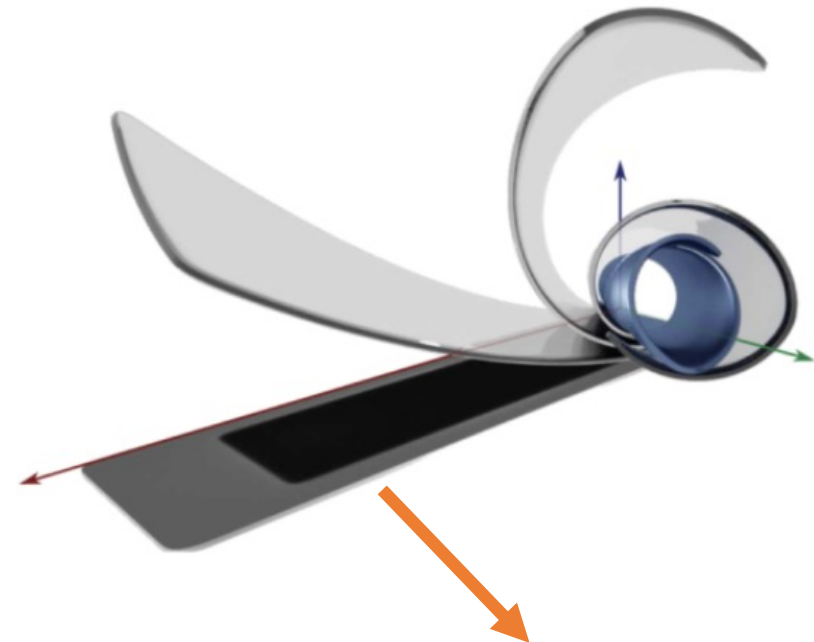
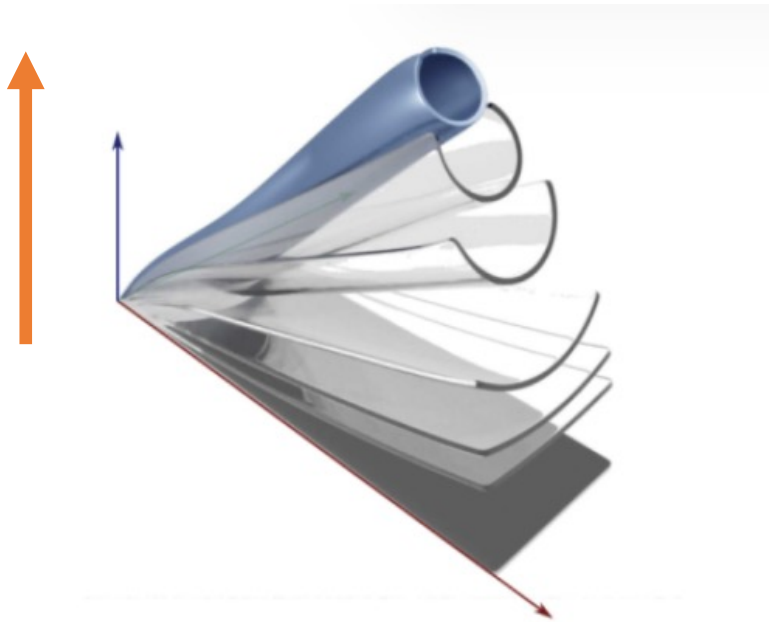
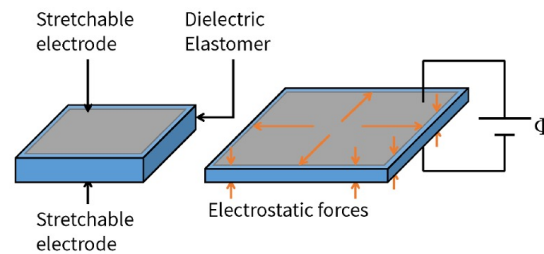
 *multiple dispatch*

ElectroMechModel

returns $(\Psi, \partial\Psi F, \partial\Psi FF)$ as closures:
evaluated at $F(\nabla u)$, $E(\nabla \phi)$, $M_1 \dots M_4$

CORE IDEA 2 · IN PRACTICE

Expressive construction of multiphysics models



WHAT IS IN THE BOX

A library of constitutive models

● Mechanical

Neo-Hookean · Mooney–Rivlin · Yeoh · Gent · Eight-chain · ARAP · incompressible and coercive-volumetric variants · Hessian regularisation

● Anisotropic

Transverse isotropy 2D/3D · HGO with 1 and 4 fibre families · multi-fibre composition

● Electro & flexo

Ideal dielectric · ElectroMechModel over any mechanical branch · FlexoElectroModel

● Magneto

Ideal magnetic 2D/3D · hard-magnetic with residual magnetisation and susceptibility · MagnetoMechModel

● Thermo

Thermal laws: constant-Cv, entropic elasticity, nonlinear melting and softening · ThermoElectroMechano

● Dissipative

Viscous incompressible · N-branch generalised Maxwell · polyconvex viscosity with internal state

One interface for all of them — `model()` → $(\psi, \text{first and second derivatives})$ — so they are interchangeable inside any driver.

Hard-magnetic soft actuators (no vacuum)

1 · Constitutive model

```
mech = NonlinearMooneyRivlin2D(λ=λ, μ1=μ1, μ2=μ2, α1=α1, α2=α2) +  
      TransverseIsotropy2D(μ=μf, α1=αf1, α2=αf2)
```

magnetisation and applied field as FE fields

```
Nh = interpolate_everywhere(x -> VectorValue(1.0, 0.0), V_N)  
Bah = interpolate_everywhere(VectorValue(0.0, -Ba), V_N)  
∂Ψmag∂F(F, N, Ba) = (-αr/μ0) * (Ba ⊗ ((F' * F) * N))
```

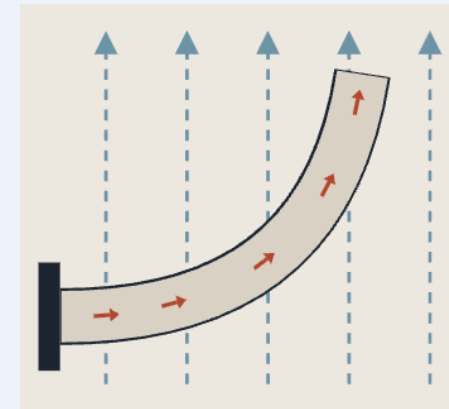
```
Ψ, ∂Ψs∂F, ∂Ψs∂FF = mech()
```

2 · Weak forms

```
res(Λ) = (u, v) -> ∫( ∇(v)' ⊙ (∂Ψs∂F ∘ (F ∘ ∇(u)', Nh)) ) dΩ + Λ * ∫( (∂Ψmag∂F ∘ (F ∘ ∇(uh⁻)', Nh, Bah)) ⊙ ∇(v)' ) dΩ  
jac(Λ) = (u, du, v) -> ∫( ∇(v)' ⊙ ((∂Ψs∂FF ∘ (F ∘ ∇(u)', Nh)) ⊙ ∇(du)') ) dΩ
```

3 · Computational model

```
nls = NewtonSolver(LUSolver(); maxiter=20, atol=1e-8, rtol=1e-2)  
model = StaticNonlinearModel(res, jac, U, V, D_bc; nls=nls)  
  
solve!(model; stepping=(nsteps=20, maxbisec=1))
```



Drivers built for nonlinear solids

1

Monolithic or staggered

The same physical model exposes per-field residuals and Jacobians, so a coupled problem can be solved fully implicitly or field-by-field without rewriting it.

2

Static, dynamic, linear

StaticNonlinearModel, DynamicNonlinearModel, StaticLinearModel and StaggeredModel share one solve! interface with a post-processing hook.

3

Robust continuation

Newton with line search — including injectivity-preserving searches that keep $\det F > 0$ — plus automatic bisection of the load factor.

4

Scales out

Distributed assembly and PETSc solvers arrive through a package extension: no change to the model code.

Topology optimisation, contact with a third medium and incompressible flow tutorials sit on the same drivers.

LEARN BY EXAMPLE

HyperFEM tutorials

github.com/MultiSimOLab/HyperFEM_tutorials

E1 Poisson

Fundamental FEM concepts on a simple Poisson problem.

BASICS

E2 Hyperelastic beam stretching

Large deformations and material nonlinearity.

MECHANO

E3 Fibre-reinforced cylinder

Anisotropic 4-fibre model under internal pressure.

MECHANO · ANISOTROPIC

E4 Electromechanical beam

Coupled electrical and mechanical fields.

ELECTRO-MECHANICS

E5 Anisotropic electromechanical beam

Anisotropy in coupled electromechanics.

ELECTRO-MECHANICS

E6 Contact with a third medium

Hyperelastic contact through a compliant third medium.

CONTACT

E7 Topology optimisation

Phase-field optimisation of a hyperelastic cantilever.

OPTIMISATION

E8 Magnetomechanical beam

Hard-magnetic actuation — the setting of this work.

MAGNETO-MECHANICS

```
# get started
$ git clone .../MultiSimOLab/HyperFEM_tutorials
pkg> activate . ; pkg> instantiate
```

Also: incompressible flow

Stokes · Stokes–SUPG · Navier–Stokes SUPG with backward Euler

HyperTop

Workshop on Advances in Shape and Topology Optimization for Hyperelastic Materials

Barcelona, 23 – 25 September 2026

Programming shape-morphing ultra-soft magneto-active structures via topology optimization

ACKNOWLEDGEMENTS

Project **POTENTIAL** Grant PID2022-141957OA-C22 funded by MCIN/AEI/10.13039/501100011033 and FEDER, UE

Project: **PRIORITY** — [EDF-2025-LS-RA-SMERO]

MULTISIMO-60Nd partnership



MULTISIMO LAB

Multiphysics Simulation and Optimisation Lab



Funded by
the European Union



UNIÓN EUROPEA



FONDO EUROPEO DE
DESARROLLO REGIONAL
"Una manera de hacer Europa"



ComputationalModels: an abstraction of numerical problem classes (e.g., static linear, static nonlinear, dynamic nonlinear, staggered schemes) designed to encapsulate PDE solvers and enable their integration within outer-loop workflows such as control, optimization, and machine learning

Problem 1: residual and jacobian

```
res_mag(Λ) = (φ, vφ) -> -1.0 * ∫((∇(vφ) · (∂Ψs∂H0 · (F · (∇(uhsolid(Λ))'), H0 · (∇(φ)), Nh))))dΩsolid -  
    ∫((∇(vφ) · (∂Ψv∂H0 · (F · (∇(uhvacuum(Λ))'), H0 · (∇(φ))))))dΩvacuum_mag
```

```
jac_mag(Λ) = (φ, dφ, vφ) -> ∫(∇(vφ)' · ((∂Ψs∂H0H0 · (F · (∇(uhsolid(Λ))'), H0 · (∇(φ)), Nh)) · ∇(dφ)))dΩsolid +  
    ∫(∇(vφ)' · ((∂Ψv∂H0H0 · (F · (∇(uhvacuum(Λ))'), H0 · (∇(φ)))) · ∇(dφ)))dΩvacuum_mag
```

Problem 2: residual and jacobian

```
res_mech(Λ) = (u, v) -> ∫((∇(v)' ⊙ (∂Ψs∂F · (F · (∇(u))'), H0 · (∇(φh(Λ))), Nh))))dΩsolid -  
    ∫((v.' · ((∂Ψv∂F · (F · (∇(uhvacuum(Λ))'), H0 · (∇(φh(Λ))))).- · nΓsf.+)))dΓsf
```

```
jac_mech(Λ) = (u, du, v) -> ∫(∇(v)' ⊙ ((∂Ψs∂FF · (F · (∇(u))'), H0 · (∇(φh(Λ))), Nh)) ⊙ (∇(du))))dΩsolid
```

Problem 3: residual and jacobian

```
res_vacmech(Λ) = (u, v) -> ∫((∇(v)' ⊙ (∂Ψvm∂F · (F · (∇(u))'))))dΩvacuum_mec
```

```
jac_vacmech(Λ) = (u, du, v) -> ∫(∇(v)' ⊙ ((∂Ψvm∂FF · (F · (∇(u))')) ⊙ (∇(du))))dΩvacuum_mec
```

ComputationalModels: an abstraction of numerical problem classes (e.g., static linear, static nonlinear, dynamic nonlinear, staggered schemes) designed to encapsulate PDE solvers and enable their integration within outer-loop workflows such as control, optimization, and machine learning

```
# *****  
#           Computational problems  
# *****  
  
# Problem 1  
nls_mag = NewtonSolver(LUSolver(); maxiter=20, atol=1.e-9, rtol=1.e-8, verbose=true)  
comp_model_mag = StaticNonlinearModel(res_mag, jac_mag, Uφ+, Vφ, Dφ; nls=nls_mag, xh=φh+)  
  
# Problem 2  
nls_mech = NewtonSolver(LUSolver(); maxiter=20, atol=1.e-6, rtol=1.e-2, verbose=true)  
comp_model_mech = StaticNonlinearModel(res_mech, jac_mech, Uu_solid+, Vu_solid, Du_solid; nls=nls_mech, xh=uh_solid+)  
  
# Problem 3  
nls_vacmech = NewtonSolver(LUSolver(); maxiter=10, rtol=2, verbose=true)  
comp_model_vacmech = StaticNonlinearModel(res_vacmech, jac_vacmech, Uu_vacuum+, Vu_vacuum, Du_vacuum; nls=nls_vacmech, xh=uh_vacuum+)  
  
# Staggered resolution: Problem1-> Problem2-> Problem3  
comp_model = StaggeredModel((comp_model_mag, comp_model_mech, comp_model_vacmech), (φh+, uh_solid+, uh_vacuum+), (φh-, uh_solid-, uh_vacuum-))
```