Implementation aspects of polytoptal methods and handling of polytopal meshes in HArDCore

Jérôme Droniou

NEMESIS
New generation methods
for numerical simulations

Towards polytopal meshes in GMSH
Montpellier, 26–27 January 2026

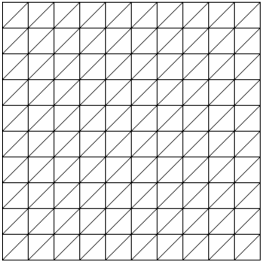
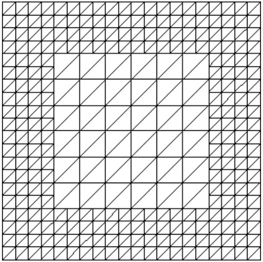
Funded by the European Union
European Research Council

Outline

Funded by the European Union
European Research Council
Established by the European Commission

Reissner–Mindlin plate problem

[Di Pietro and Droniou, 2022]

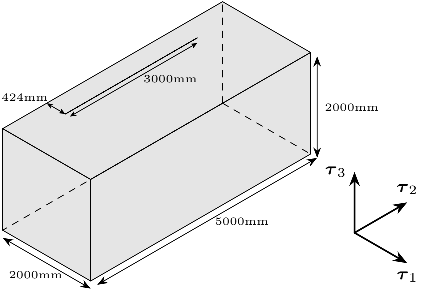
Stabilised $\mathcal{P}^2$-($\mathcal{P}^1 + \mathcal{B}^3$) scheme		DDR scheme	
			
nb. DOFs	Error	nb. DOFs	Error
2403	0.138	550	0.161
9603	6.82e-2	2121	6.77e-2
38402	3.40e-2	8329	3.1e-2

Funded by the European Union
European Research Council

Electromagnetic wave I

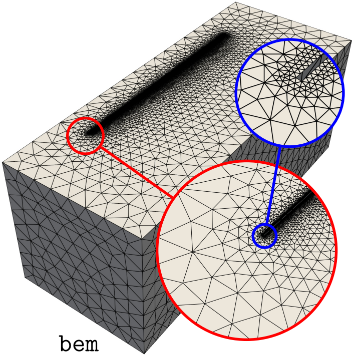
[Touzalin, 2025]

Problem: use a boundary element method to analyse the shielding effectiveness of a perfectly conductive box with a very small slit.

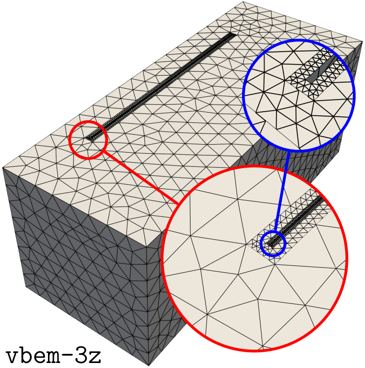


Electromagnetic wave II

Meshes: conforming triangular for finite-element boundary method (bem), non-conforming triangular (polygonal) for virtual element boundary method (vbem-3z).



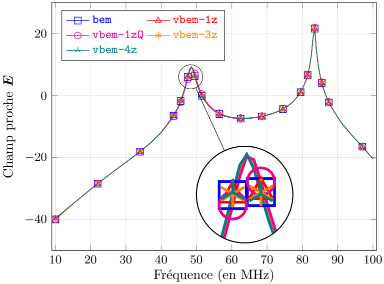
bem

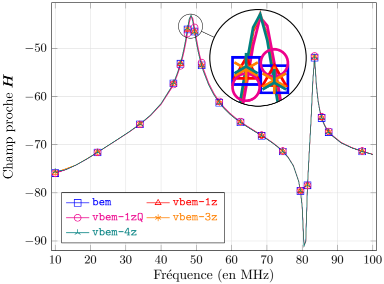


vbem-3z

Electromagnetic wave III

Accuracy: comparison of modulus of reflected near fields at the top.





Computational cost

Method	*Assembly*	*Resolution*
bem	813s	125s
vbem-3z	321s	19s

Outline

Algebraic realisation of potential

- Space: $\underline{V}_T^k = \{\underline{v}_T = (v_T, (v_F)_{F\in\mathcal{F}_T}) \,:\, v_T \in \mathcal{P}^{k-1}(T)\,,\ v_F \in \mathcal{P}^k(F)\}$.
- Potential reconstruction ($k \geq 1$):

$$\int_T \nabla p_T^{k+1}\underline{v}_T \cdot \nabla w = -\int_T v_T \Delta w + \sum_{F\in\mathcal{F}_T} \int_F v_F\,(\nabla w \cdot \boldsymbol{n}_{TF}) \quad \forall w \in \mathcal{P}^{k+1}(T),$$

$$\int_T p_T^{k+1}\underline{v}_T = \int_T v_T.$$

- Recast as: for all $w \in \mathcal{P}^{k+1}(T)$,

$$(\nabla p_T^{k+1}\underline{v}_T, \nabla w)_T + (p_T^{k+1}\underline{v}_T, 1)_T (w,1)_T = -(v_T, \Delta w)_T + \sum_{F\in\mathcal{F}_T} (v_F, \nabla w \cdot \boldsymbol{n}_{TF})_F + (v_T, 1)_T (w,1)_T.$$

(Solution of a Riesz representation problem)

Funded by the European Union
European Research Council

Matrix of $p_T^{k+1} : \underline{V}_T^k \to \mathcal{P}^{k+1}(T)$ on selected bases

- Pick bases Φ_F^k of $\mathcal{P}^k(F)$ for $F \in \mathcal{F}_T$ and Φ_T^{k-1} of $\mathcal{P}^{k-1}(T)$.

$$\text{Basis of } \underline{V}_T^k : \Phi_{F_1}^k \times \cdots \times \Phi_{F_{N_{\partial T}}}^k \times \Phi_T^{k-1}.$$

(Increasing dimension of mesh entities)

- Pick basis $\widehat{\Phi}_T^{k+1}$ of $\mathcal{P}^{k+1}(T)$.

Matrix of $p_T^{k+1} : \underline{V}_T^k \to \mathcal{P}^{k+1}(T)$ on selected bases

- Pick bases Φ_F^k of $\mathcal{P}^k(F)$ for $F \in \mathcal{F}_T$ and Φ_T^{k-1} of $\mathcal{P}^{k-1}(T)$.

$$\text{Basis of } \underline{V}_T^k : \Phi_{F_1}^k \times \cdots \times \Phi_{F_{N_{\partial T}}}^k \times \Phi_T^{k-1}.$$

(Increasing dimension of mesh entities)

- Pick basis $\widehat{\Phi}_T^{k+1}$ of $\mathcal{P}^{k+1}(T)$.

- Notations:

$$\begin{aligned}
\Phi_F^k &= \{\varphi_1^F, \ldots, \varphi_{N_{k,F}}^F\},\\
\Phi_T^{k-1} &= \{\varphi_1^T, \ldots, \varphi_{N_{k,T}}^T\},\\
\widehat{\Phi}_T^{k+1} &= \{\widehat{\varphi}_1^T, \ldots, \widehat{\varphi}_{N_{k+1,T}}^T\}.
\end{aligned}$$

Matrix of $p_T^{k+1} : \underline{V}_T^k \to \mathcal{P}^{k+1}(T)$ on selected bases

- Pick bases Φ_F^k of $\mathcal{P}^k(F)$ for $F \in \mathcal{F}_T$ and Φ_T^{k-1} of $\mathcal{P}^{k-1}(T)$.

$$\text{Basis of } \underline{V}_T^k : \Phi_{F_1}^k \times \cdots \times \Phi_{F_{N_{\partial T}}}^k \times \Phi_T^{k-1}.$$

(Increasing dimension of mesh entities)

- Pick basis $\widehat{\Phi}_T^{k+1}$ of $\mathcal{P}^{k+1}(T)$.

- Notations: vector $\underline{\mathsf{V}}_T$ of $\underline{v}_T \in \underline{V}_T^k$ written as:

$$\underline{\mathsf{V}}_T = \begin{bmatrix} \mathsf{V}_{F_1} \\ \vdots \\ \mathsf{V}_{F_{N_{\partial T}}} \\ \mathsf{V}_T \end{bmatrix} \quad \text{with} \quad \begin{array}{l} \mathsf{V}_F = [V_i^F]_{i=1,\ldots,N_{k,F}} \text{ coefficients on } \Phi_F^k, \\ \mathsf{V}_T = [V_i^T]_{i=1,\ldots,N_{k-1,T}} \text{ coefficients on } \Phi_T^{k-1}. \end{array}$$

Matrix of $p_T^{k+1} : \underline{V}_T^k \to \mathcal{P}^{k+1}(T)$ on selected bases

- Definition:

$$(\nabla p_T^{k+1}\underline{v}_T, \nabla w)_T + (p_T^{k+1}\underline{v}_T, 1)_T(w,1)_T$$
$$= -(v_T, \Delta w)_T + (v_T, 1)_T(w,1)_T + \sum_{F\in\mathcal{F}_T}(v_F, \nabla w\cdot \boldsymbol{n}_{TF})_F.$$

- Algebraic translation: the coefficients P_T of $p_T^{k+1}\underline{v}_T$ on $\widehat{\Phi}_T^{k+1}$ satisfy

$$\left(\mathbf{S}_T + \widehat{\mathsf{L}}_T^{k+1}(\widehat{\mathsf{L}}_T^{k+1})^\intercal\right)\mathsf{P}_T = \left(\mathbf{B}_{P,T} + \widehat{\mathsf{L}}_T^{k+1}(\mathsf{L}_T^{k-1})^\intercal\right)\mathsf{V}_T + \sum_{F\in\mathcal{F}_T}\mathbf{B}_{P,F}\mathsf{V}_F,$$

with

$$\mathbf{S}_T := \left[(\nabla\widehat{\varphi}_i^T, \nabla\widehat{\varphi}_j^T)_T\right]_{1\le i,j\le N_{k+1,T}}, \quad \widehat{\mathsf{L}}_T^{k+1} := [(\widehat{\varphi}_i^T, 1)_T]_{1\le i\le N_{k+1,T}},$$
$$\mathbf{B}_{P,T} := \left[-(\Delta\widehat{\varphi}_i^T, \varphi_j^T)_T\right]_{1\le i\le N_{k+1,T}, 1\le j\le N_{k-1,T}}, \quad \mathsf{L}_T^{k-1} := [(\varphi_i^T, 1)_T]_{1\le i\le N_{k-1,T}}$$
$$\mathbf{B}_{P,F} := \left[(\nabla\widehat{\varphi}_i^T\cdot\boldsymbol{n}_{TF}, \varphi_j^F)_F\right]_{1\le i\le N_{k+1,T}, 1\le j\le N_{k,F}}.$$

Matrix of $p_T^{k+1} : \underline{V}_T^k \to \mathcal{P}^{k+1}(T)$ on selected bases

- The coefficients P_T of $p_T^{k+1}\underline{v}_T$ on $\widehat{\Phi}_T^{k+1}$ satisfy

$$\left(\mathbf{S}_T + \widehat{\mathsf{L}}_T^{k+1}(\widehat{\mathsf{L}}_T^{k+1})^\intercal\right)\mathsf{P}_T = \left(\mathbf{B}_{P,T} + \widehat{\mathsf{L}}_T^{k+1}(\mathsf{L}_T^{k-1})^\intercal\right)\mathsf{V}_T + \sum_{F\in\mathcal{F}_T}\mathbf{B}_{P,F}\mathsf{V}_F,$$

$$\mathbf{S}_T := \left[(\nabla\widehat{\varphi}_i^T,\nabla\widehat{\varphi}_j^T)_T\right]_{1\le i,j\le N_{k+1,T}}, \quad \widehat{\mathsf{L}}_T^{k+1} := [(\widehat{\varphi}_i^T,1)_T]_{1\le i\le N_{k+1,T}},$$

$$\mathbf{B}_{P,T} := \left[-(\Delta\widehat{\varphi}_i^T,\varphi_j^T)_T\right]_{1\le i\le N_{k+1,T},1\le j\le N_{k-1,T}}, \quad \mathsf{L}_T^{k-1} := [(\varphi_i^T,1)_T]_{1\le i\le N_{k-1,T}}$$

$$\mathbf{B}_{P,F} := \left[(\nabla\widehat{\varphi}_i^T\cdot\boldsymbol{n}_{TF},\varphi_j^F)_F\right]_{1\le i\le N_{k+1,T},1\le j\le N_{k,F}}.$$

- The matrix $\mathbf{P}_T$ of p_T^{k+1} is thus obtained by solving

$$\left(\mathbf{S}_T + \widehat{\mathsf{L}}_T^{k+1}(\widehat{\mathsf{L}}_T^{k+1})^\intercal\right)\mathbf{P}_T = \left[\mathbf{B}_{P,F_1} \quad \cdots \quad \mathbf{B}_{P,F_{N_{\partial T}}} \quad \mathbf{B}_{P,T} + \widehat{\mathsf{L}}_T^{k+1}(\mathsf{L}_T^{k-1})^\intercal\right].$$

Matrix of $p_T^{k+1} : \underline{V}_T^k \to \mathcal{P}^{k+1}(T)$ on selected bases

- The coefficients P_T of $p_T^{k+1}\underline{v}_T$ on $\widehat{\Phi}_T^{k+1}$ satisfy

$$\left(\mathbf{S}_T + \widehat{\mathsf{L}}_T^{k+1}(\widehat{\mathsf{L}}_T^{k+1})^\intercal\right)\mathsf{P}_T = \left(\mathbf{B}_{P,T} + \widehat{\mathsf{L}}_T^{k+1}(\mathsf{L}_T^{k-1})^\intercal\right)\mathsf{V}_T + \sum_{F\in\mathcal{F}_T}\mathbf{B}_{P,F}\mathsf{V}_F,$$

$$\mathbf{S}_T := \left[(\nabla\widehat{\varphi}_i^T, \nabla\widehat{\varphi}_j^T)_T\right]_{1\le i,j\le N_{k+1,T}}, \quad \widehat{\mathsf{L}}_T^{k+1} := [(\widehat{\varphi}_i^T, 1)_T]_{1\le i\le N_{k+1,T}},$$

$$\mathbf{B}_{P,T} := \left[-(\Delta\widehat{\varphi}_i^T, \varphi_j^T)_T\right]_{1\le i\le N_{k+1,T}, 1\le j\le N_{k-1,T}}, \quad \mathsf{L}_T^{k-1} := [(\varphi_i^T, 1)_T]_{1\le i\le N_{k-1,T}}$$

$$\mathbf{B}_{P,F} := \left[(\nabla\widehat{\varphi}_i^T\cdot\boldsymbol{n}_{TF}, \varphi_j^F)_F\right]_{1\le i\le N_{k+1,T}, 1\le j\le N_{k,F}}.$$

- The matrix $\mathbf{P}_T$ of p_T^{k+1} is thus obtained by solving

$$\left(\mathbf{S}_T + \widehat{\mathsf{L}}_T^{k+1}(\widehat{\mathsf{L}}_T^{k+1})^\intercal\right)\mathbf{P}_T = \left[\mathbf{B}_{P,F_1} \quad \cdots \quad \mathbf{B}_{P,F_{N_{\partial T}}} \quad \mathbf{B}_{P,T} + \widehat{\mathsf{L}}_T^{k+1}(\mathsf{L}_T^{k-1})^\intercal\right].$$

- A few requirements for the library:
  - Gram matrices of families of polynomials (integrate products).
  - Differential operators between families of polynomials.
  - Direct access to faces of an elements, and integrate on them.

HArDCore library (`https://github.com/jdroniou/HArDCore`)

- Linear algebra: via Eigen (`https://gitlab.com/libeigen/eigen`).
- In-house Mesh classes and Polynomial family classes.
- Fast integration rules [Chin et al., 2015]: essential for polytopal methods, requires full connectivity from element to vertices, as well as geometric information (outer normal, etc.).

  (In some instances, quadrature-based integration is still required.)
- Various interfaces:
  - ⋆ PaSTiX (`https://solverstack.gitlabpages.inria.fr/pastix/`),
  - ⋆ PETSc (`https://petsc.org/release/`),
  - ⋆ etc.
- Many polytopal methods and schemes : HHO, DDR, VEM, elasticity with Tresca contact, plates, Yang–Mills, etc.

Outline

Construction of polynomial bases

- Store vectors of powers up to degree ℓ

$$\{\boldsymbol{\alpha} = (\omega_1, \ldots, \omega_d) \in \mathbb{N}^d \;:\; \sum_i \omega_i = \ell\} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_{N_\ell}\},$$

which define the monomials ($\boldsymbol{x}_T$ = center of mass of T, h_T = diameter)

$$m_j(\boldsymbol{x}) = \left(\frac{x_1 - x_{T,1}}{h_T}\right)^{\alpha_{j,1}} \cdots \left(\frac{x_1 - x_{T,d}}{h_T}\right)^{\alpha_{j,d}}.$$

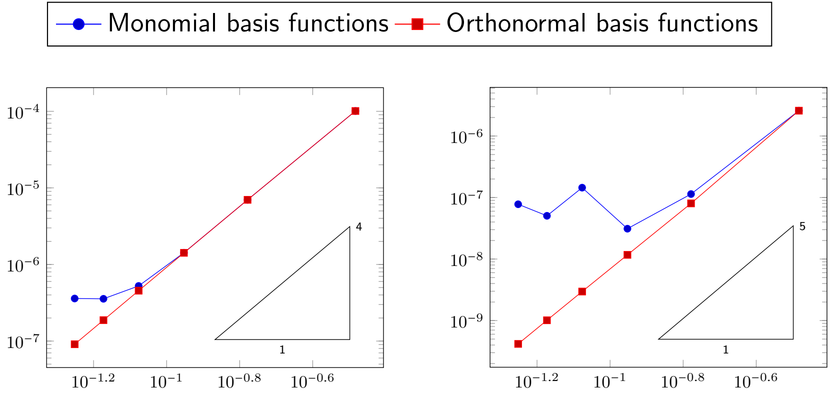
- Basis $\Phi_T^\ell = \{\phi_1, \ldots, \phi_{N_\ell}\}$ of $\mathcal{P}^\ell(T)$: linear combination of monomials

$$\phi_r = \sum_{j=1}^{N_\ell} M_{rj} m_j.$$

- Extensions: vector- or matrix-valued polynomials by tensorisation.
- **`PolynomialFamily`**: defined by degree ℓ, mesh entity (T, F, etc.), generators (list of scalars, vectors, matrices) and matrix M.

Construction of polynomial bases

Local polynomial bases on $\mathcal{P}^k(T)$, $\mathcal{P}^k(F)$ are orthonormalised!



(a) Energy norm vs. h, Kershaw meshes, $k = 3$.

(b) L^2-norm vs. h, Kershaw meshes, $k = 3$.

Unified class for all kinds of polynomial spaces

- **`PolynomialFamily`**: defined by degree ℓ, mesh entity (T, F, etc.), generators (list of scalars, vectors, matrices) and matrix M.
- Suitable choices of generators and matrix describe **`PolynomialFamilies`** that are not $\mathcal{P}^{\ell}(X)$.
  For example, for PF *polynomial family:*
  - ⋆ **`gradient`**(PF), **`divergence`**(PF), **`curl`**(PF), *etc.*
  - ⋆ L(PF) *if L linear map acting on generators.*
  - ⋆ $\mathrm{PF}_1 + \mathrm{PF}_2$ *(sum of spanned vector spaces).*

Unified class for all kinds of polynomial spaces

- **`PolynomialFamily`**: defined by degree ℓ, mesh entity (T, F, etc.), generators (list of scalars, vectors, matrices) and matrix M.
- Suitable choices of generators and matrix describe **`PolynomialFamilies`** that are not $\mathcal{P}^\ell(X)$.
  For example, for PF *polynomial family:*
  - ⋆ **`gradient`**(PF), **`divergence`**(PF), **`curl`**(PF), *etc.*
  - ⋆ L(PF) *if* L *linear map acting on generators.*
  - ⋆ $\text{PF}_1 + \text{PF}_2$ *(sum of spanned vector spaces).*
- Koszul complements are also described as **`PolynomialFamilies`**, e.g.:

$$\mathcal{R}^{c,\ell}(T) = (\boldsymbol{x} - \boldsymbol{x}_T)\mathcal{P}^{\ell-1}(T)\,, \quad \mathcal{G}^{c,\ell}(T) = (\boldsymbol{x} - \boldsymbol{x}_T) \times \mathcal{P}^{\ell-1}(T)^3.$$

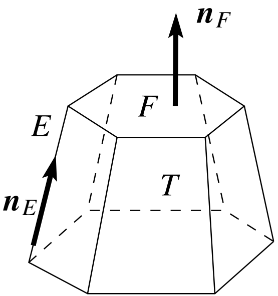
  (leads to straightforward description of arbitrary-order Nédélec and Raviart–Thomas elements)
- Unified class of polynomials means unified (fast) integration rules.

Outline

Funded by the European Union
European Research Council
Established by the European Commission

Classes for mesh entities

- Classes for **`Cell`**, **`Face`**, **`Edge`**, **`Vertex`**.
  All templated versions of the same **`MeshObject`** *class.*
- Full connectivity, for example each **`Face`** embeds (pointers to) its neighbouring **`Cells`**, and the **`Edges`**, **`Vertex`** it contains.
  Useful for local constructions of operators, and fast integration rule.
- Lots of additional embedded information (where relevant): global index; diameter, volume/area/length; center of mass; outer normal, tangent vector; relative orientations of sub-mesh entities; etc.



Classes for mesh entities

- Classes for **`Cell`**, **`Face`**, **`Edge`**, **`Vertex`**.

  All templated versions of the same **`MeshObject`** *class.*

- Full connectivity, for example each **`Face`** embeds (pointers to) its neighbouring **`Cells`**, and the **`Edges`**, **`Vertex`** it contains.

  Useful for local constructions of operators, and fast integration rule.

- Lots of additional embedded information (where relevant): global index; diameter, volume/area/length; center of mass; outer normal, tangent vector; relative orientations of sub-mesh entities; etc.

- Other features:
  - ★ Each cell/face has a simplicial subdivision.
  - ★ 2-level meshes can be handled (coarsened mesh from fine mesh).
  - ★ Mesh transformation/handlers: move vertices; split non-planar faces; etc.

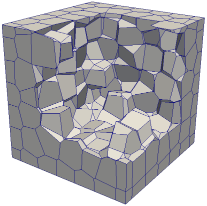
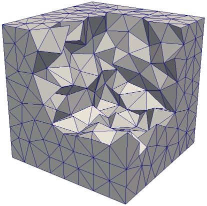
Mesh construction

- Full mesh described in (large) "RF" mesh files with minimal (non-redundant) information.
- **`MeshBuilder`** takes care of creating all the connectivity.

Since last October, the development version of `HArDCore` also has an interface with `GMesh` (`https://gmsh.info/`).

RF mesh file data structure

- Two files for each 3D mesh: `M.node` for vertices, `M.ele` for cells.
- In `M.node`, one line per vertex:
  <vertex id> <x coordinate> <y coordinate> <z coordinate>
- In `M.ele`, each cell is given by:
  - ⋆ Header line: <cell id> <number of faces>
  - ⋆ For each face in the cell:
    <local face id> <number of vertices> <id of first vertex> <id of second vertex> ...
    (vertices listed in order around the face boundary)

Mesh builder

- From RF file or GMesh mesh, creates **`std::vector`** to list vertices and cells (as in RF file format).
- To create a mesh entity in **`Mesh`**, it is split into simplices:
  - ⋆ ensures that orientation will be correct,
  - ⋆ allows for calculation of center of mass, measure, etc.

  (Assumes that faces and cells are star-shaped w.r.t. average of their vertices).
- Entities (partially) created and added by increasing dimension:
  - ⋆ Vertices (no connectivity at this stage).
  - ⋆ Edges (connected to vertices and gives connexion between vertices).
  - ⋆ etc.

Funded by the European Union
European Research Council

Conclusion and transition

- Building polytopal schemes requires:
  - ⋆ Selecting degrees of freedom *(polynomials on vertices, edges, faces, elements)*.
  - ⋆ Implementing reconstruction operators as the solutions to local problems *(Riez representation problems)*.

  No need for explicit shape functions.

- Same approach can be applied to (complex) finite elements, without having to know their shape functions *(typically challenging in some FE for elasticity)*.

- Coding polytopal methods is facilitated by a rich mesh structure, well-suite polynomial bases and efficient libraries for polynomial integration.

Funded by the European Union (ERC Synergy, NEMESIS, project number 101115663). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Thank you for your attention!

References

Chin, E. B., Lasserre, J. B., and Sukumar, N. (2015).
Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra.
Comput. Mech., 56(6):967–981.

Di Pietro, D. A. and Droniou, J. (2022).
A discrete de Rham method for the Reissner–Mindlin plate bending problem on polygonal meshes.
Comput. Math. Appl., 125:136–149.

Touzalin, A. (2025).
Méthode des éléments virtuels pour la discrétisation des équations intégrales de frontière en électromagnétisme dans le domaine fréquentiel.
PhD Thesis, CEA-CESTA.

Funded by the European Union
European Research Council