Shape and topology optimisation with GridapTopOpt

Zachary J Wegert, Jordi Manyer, Connor Mallon, Santiago Badia, Vivien J Challis

School of Mathematical Sciences, Queensland University of Technology, Australia

27 March 2026

Queensland University of Technology

What is shape and topology optimisation for computational design?

Domain

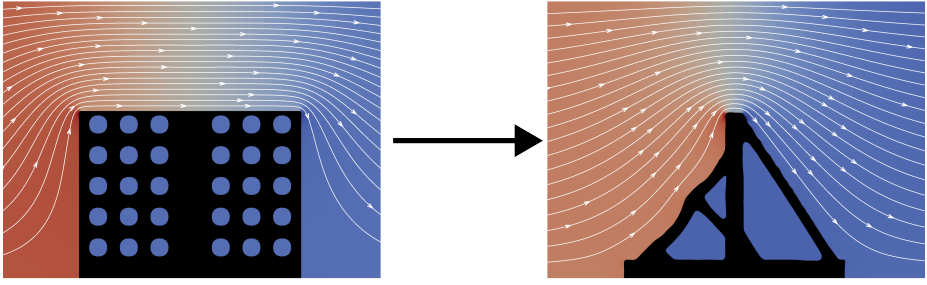
$$\min_{\Omega \in D} J(\Omega, \boldsymbol{u}(\Omega))$$

PDE solution

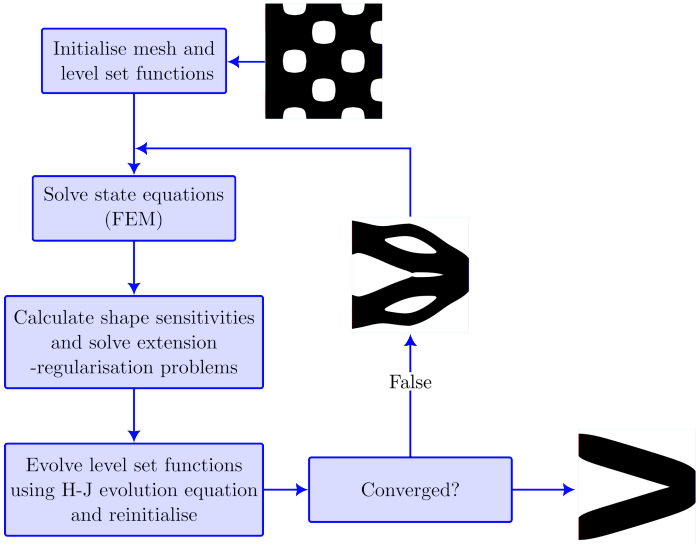
$$\text{s.t. } C_j(\Omega, \boldsymbol{u}(\Omega)) = 0,\ j = 1, \ldots, N,$$

$$\begin{cases} \boldsymbol{u} \in \mathcal{U}(\Omega) \\ a(\boldsymbol{u}, \boldsymbol{v}) = l(\boldsymbol{v}),\ \forall \boldsymbol{v} \in \mathcal{V}(\Omega) \end{cases}$$

PDE weak form



Crash course on level set-based topology optimisation

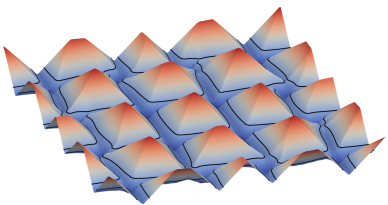


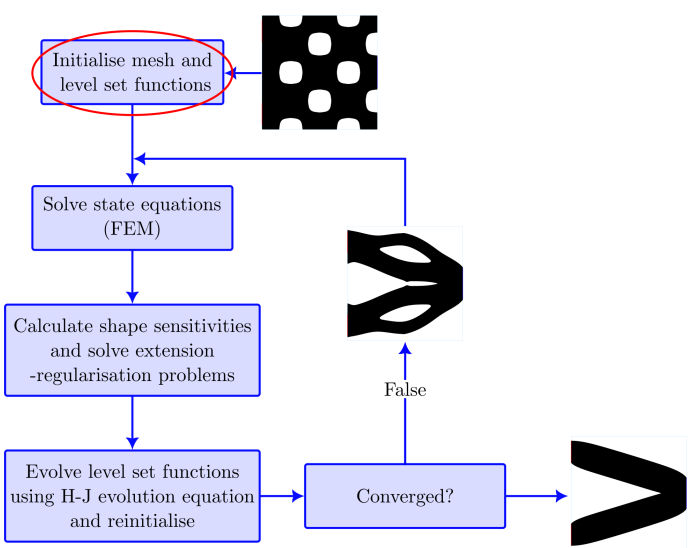
Crash course on level set-based topology optimisation

Initialisation [8]

- Fixed computational domain D
- Implicitly track $\Omega \subset D$ via a level set function ϕ with

$$\begin{cases} \phi(\boldsymbol{x}) < 0 & \text{if } \boldsymbol{x} \in \Omega, \\ \phi(\boldsymbol{x}) = 0 & \text{if } \boldsymbol{x} \in \partial\Omega, \\ \phi(\boldsymbol{x}) > 0 & \text{if } \boldsymbol{x} \in D \backslash \bar{\Omega}. \end{cases}$$





Crash course on level set-based topology optimisation

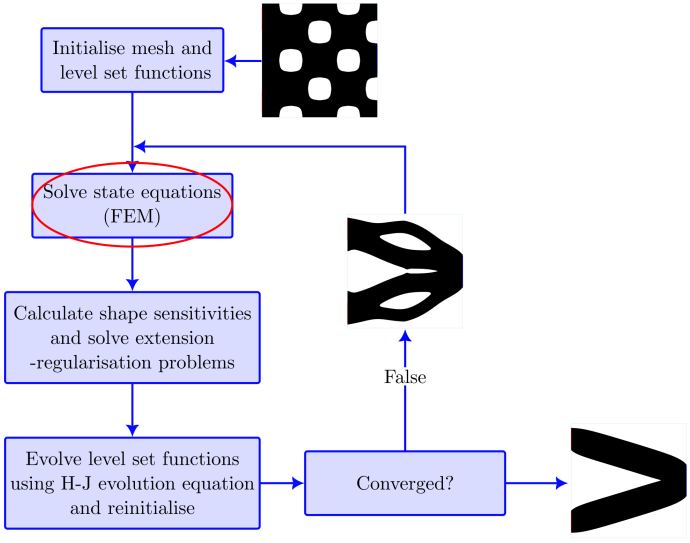
Solve PDEs

- Solve with body-fitted mesh, unfitted methods, or interface smoothing
- Interface smoothing: *Find* $u \in U$ *such that*

$$\int_D H(\phi)\kappa \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_{\Gamma_N} gv \, \mathrm{d}\Omega,$$

for all $v \in V$.

- Interpolate between Ω and $D \backslash \Omega$ via smoothed Heaviside function H



Crash course on level set-based topology optimisation

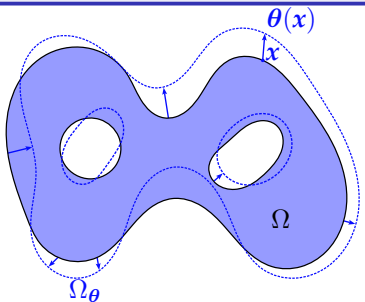
Calculate shape derivatives [11, 1, 2]

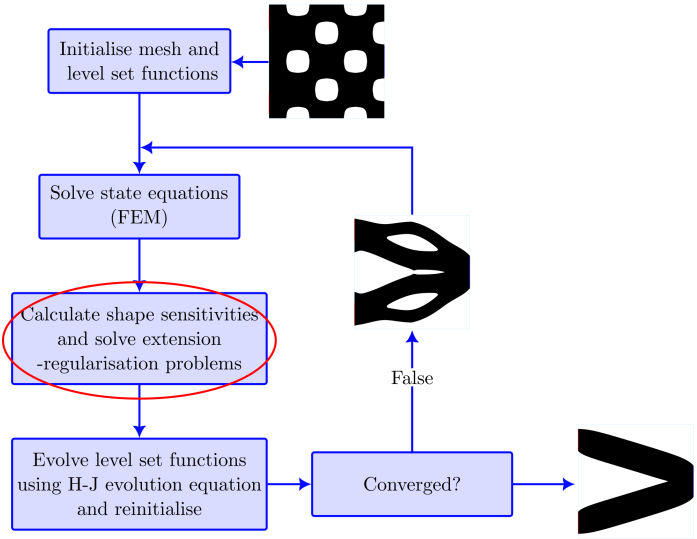
E.g., find Fréchet derivative of

$$J(\Omega) = \int_\Omega \kappa \|\nabla u\|^2 \, \mathrm{d}\Omega$$

under smooth variation

$$\boldsymbol{\theta} \mapsto (\boldsymbol{I} + \boldsymbol{\theta})(\Omega) \; (=: \Omega_{\boldsymbol{\theta}})$$





Crash course on level set-based topology optimisation

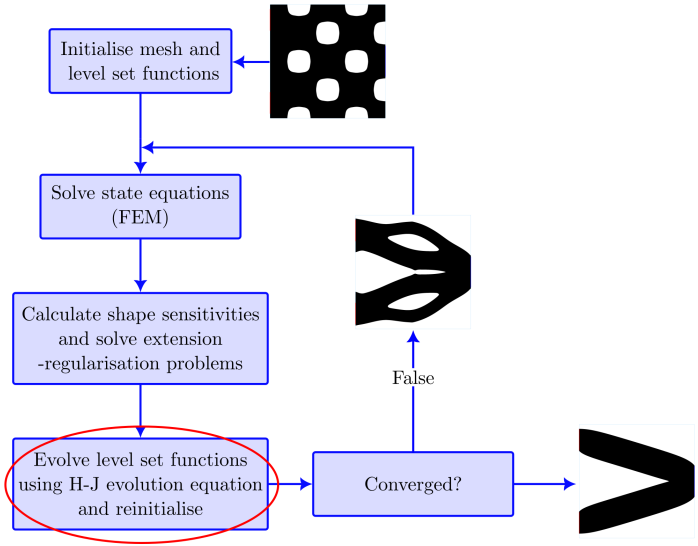
Evolve and reinitialise LSF [8, 9]

Hamilton-Jacobi evolution equation:

$$\begin{cases} \frac{\partial \phi}{\partial t} + V(\boldsymbol{x})\|\boldsymbol{\nabla}\phi\| = 0, \\ \phi(0,\boldsymbol{x}) = \phi_0(\boldsymbol{x}),\ \boldsymbol{x} \in D,\ t \in (0,T). \end{cases}$$

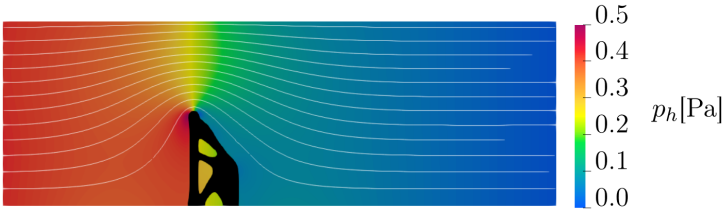
Reinitialisation equation:

$$\begin{cases} \frac{\partial \phi}{\partial t} + \text{sign}(\phi)(\|\boldsymbol{\nabla}\phi\| - 1) = 0, \\ \phi(0,\boldsymbol{x}) = \phi_0(\boldsymbol{x}),\ \boldsymbol{x} \in D,\ t \in (0,T). \end{cases}$$

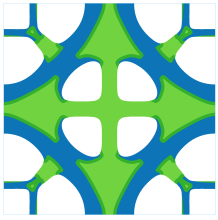


Original motivation for GridapTopOpt.jl

- Interested in developing new methods for multi-physics & multi-material problems $\longrightarrow$ Extensible level-set framework
- Avoid reinventing the wheel for new problems/methods $\longrightarrow$ high-level interface and automatic differentiation
- 2D & 3D $\longrightarrow$ Easily switch between serial & distributed regimes



(a) Mallon et al. [6]



(b) Wegert et al. [12]

Automatic differentiation for PDE-constrained functionals

$$\text{Find } \mathrm{d}J(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p}))(\boldsymbol{q}) \text{ where } \begin{cases} \boldsymbol{u} \in \mathcal{U} \text{ s.t.,} \\ R(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{p}) = 0, \ \forall \boldsymbol{v} \in \mathcal{V}. \end{cases}$$

Automatic differentiation for PDE-constrained functionals

$$\text{Find } \mathrm{d}J(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p}))(\boldsymbol{q}) \text{ where } \begin{cases} \boldsymbol{u} \in \mathcal{U} \text{ s.t.,} \\ R(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{p}) = 0, \ \forall \boldsymbol{v} \in \mathcal{V}. \end{cases}$$

GridapTopOpt uses a mix of forward-mode and reverse-mode automatic differentiation:

1. Partial derivatives computed cell-wise using forward-mode AD (e.g., $\frac{\mathrm{d}}{\mathrm{d}t} J(\boldsymbol{p} + \boldsymbol{q}t, u)|_{t=0}$)

Theorem (Céa's adjoint method)

Suppose $J(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p}))$ is a sufficiently smooth functional and $\boldsymbol{u} \in \mathcal{U}$ depends on $\boldsymbol{p} \in \Phi$ through the residual

$$R(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{p}) = 0, \ \forall \boldsymbol{v} \in \mathcal{V}.$$

The directional derivative of J at $\boldsymbol{p}$ in the direction $\boldsymbol{\psi}$ is given by

$$\mathrm{d}J(\boldsymbol{p}; \boldsymbol{\psi}) = \partial_{\boldsymbol{p}} J(\boldsymbol{p}, \boldsymbol{u})(\psi) - \partial_{\boldsymbol{p}} R(\boldsymbol{u}, \boldsymbol{\lambda}; \boldsymbol{p})(\boldsymbol{\psi})$$

where $\boldsymbol{\lambda} \in V$ satisfies the weak formulation $\partial_{\boldsymbol{u}} R(\boldsymbol{u}, \boldsymbol{\lambda}; \boldsymbol{\phi})(\boldsymbol{w}) = \partial_{\boldsymbol{u}} J(\boldsymbol{p}, \boldsymbol{u})(\boldsymbol{w}), \ \forall \boldsymbol{w} \in U$.

Theorem (Céa's adjoint method)

Suppose $J(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p}))$ is a sufficiently smooth functional and $\boldsymbol{u} \in \mathcal{U}$ depends on $\boldsymbol{p} \in \Phi$ through the residual

$$R(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{p}) = 0, \ \forall \boldsymbol{v} \in \mathcal{V}.$$

The directional derivative of J at $\boldsymbol{p}$ in the direction $\boldsymbol{\psi}$ is given by

$$\mathrm{d}J(\boldsymbol{p}; \boldsymbol{\psi}) = \partial_{\boldsymbol{p}} J(\boldsymbol{p}, \boldsymbol{u})(\psi) - \partial_{\boldsymbol{p}} R(\boldsymbol{u}, \boldsymbol{\lambda}; \boldsymbol{p})(\boldsymbol{\psi})$$

where $\boldsymbol{\lambda} \in V$ satisfies the weak formulation $\partial_{\boldsymbol{u}} R(\boldsymbol{u}, \boldsymbol{\lambda}; \boldsymbol{\phi})(\boldsymbol{w}) = \partial_{\boldsymbol{u}} J(\boldsymbol{p}, \boldsymbol{u})(\boldsymbol{w}), \ \forall \boldsymbol{w} \in U.$

- Partial derivatives computed with ❶

Theorem (Céa's adjoint method)

Suppose $J(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p}))$ is a sufficiently smooth functional and $\boldsymbol{u} \in \mathcal{U}$ depends on $\boldsymbol{p} \in \Phi$ through the residual

$$R(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{p}) = 0, \ \forall \boldsymbol{v} \in \mathcal{V}.$$

The directional derivative of J at $\boldsymbol{p}$ in the direction $\boldsymbol{\psi}$ is given by

$$\mathrm{d}J(\boldsymbol{p}; \boldsymbol{\psi}) = \partial_{\boldsymbol{p}} J(\boldsymbol{p}, \boldsymbol{u})(\psi) - \partial_{\boldsymbol{p}} R(\boldsymbol{u}, \boldsymbol{\lambda}; \boldsymbol{p})(\boldsymbol{\psi})$$

where $\boldsymbol{\lambda} \in V$ satisfies the weak formulation $\partial_{\boldsymbol{u}} R(\boldsymbol{u}, \boldsymbol{\lambda}; \boldsymbol{\phi})(\boldsymbol{w}) = \partial_{\boldsymbol{u}} J(\boldsymbol{p}, \boldsymbol{u})(\boldsymbol{w}), \ \forall \boldsymbol{w} \in U.$

- Partial derivatives computed with [1]
- New FE operators built into GridapTopOpt:
  - Construct forward & adjoint systems
  - Cache & solve appropriately in serial/distributed
  - Implements reverse-mode rules that can be called by reverse AD package (e.g., Zygote.jl)

Automatic differentiation for PDE-constrained functionals

$$\text{Find } \mathrm{d}J(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p}))(\boldsymbol{q}) \text{ where } \begin{cases} \boldsymbol{u} \in \mathcal{U} \text{ s.t.,} \\ R(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{p}) = 0, \ \forall \boldsymbol{v} \in \mathcal{V}. \end{cases}$$

GridapTopOpt uses a mix of forward-mode and reverse-mode automatic differentiation:

1. Partial derivatives computed cell-wise using forward-mode AD (e.g., $\frac{\mathrm{d}}{\mathrm{d}t} J(\boldsymbol{p} + \boldsymbol{q}t, u)|_{t=0}$)
2. Adjoint methods implemented for PDE-constraints

Automatic differentiation for PDE-constrained functionals

$$\text{Find } \mathrm{d}J(\boldsymbol{p},\boldsymbol{u}(\boldsymbol{p}))(\boldsymbol{q}) \text{ where } \begin{cases} \boldsymbol{u} \in \mathcal{U} \text{ s.t.,} \\ R(\boldsymbol{u},\boldsymbol{v};\boldsymbol{p}) = 0, \ \forall \boldsymbol{v} \in \mathcal{V}. \end{cases}$$

GridapTopOpt uses a mix of forward-mode and reverse-mode automatic differentiation:

1. Partial derivatives computed cell-wise using forward-mode AD (e.g., $\frac{\mathrm{d}}{\mathrm{d}t} J(\boldsymbol{p} + \boldsymbol{q}t, u)|_{t=0}$)
2. Adjoint methods implemented for PDE-constraints
3. (If needed) Reverse-mode AD to differentiate through complicated mappings — hooks into appropriate reverse-rule where needed

Simple example: `https://zjwegert.github.io/GridapTopOpt.jl/dev/examples/AD-PDE-constrained-functions/`

Core design

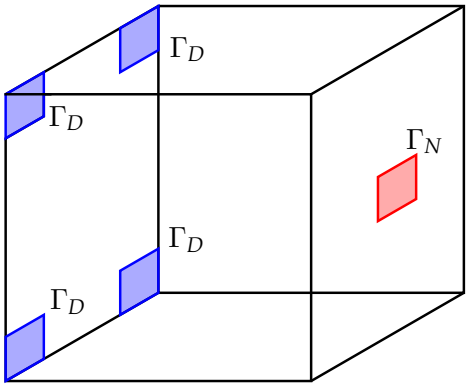
Original version (Wegert et al. [13]): level-set method with interface smoothing [2]

Source code breakdown (latest version):

- `src/Optimisers/`: Optimisers for level-set methods (augmented Lagrangian & projection method)
- `src/LevelSetEvolution/`: FD and FE solvers for Hamilton-Jacobi and reinitialisation equations
- `src/StateMaps/`: Adjoint methods & reverse-mode rules
- ...

Dependencies: PartitionedArrays.jl [2], GridapDistributed.jl [2], GridapPETSc.jl [10], GridapSolvers.jl [7], ...

Example – Minimum thermal compliance



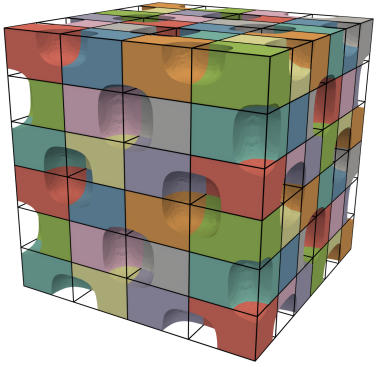
$$\min_{\phi} \int_D H_\eta(\phi) \|\boldsymbol{\nabla} u\|^2 \, \mathrm{d}\boldsymbol{x}$$

$$\text{s.t. } \mathrm{Vol}(\Omega) = V_f,$$

$$\begin{cases} u \in H^1_{\Gamma_D}(D) \\ \displaystyle\int_D H_\eta(\phi) \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \, \mathrm{d}\boldsymbol{x} = \int_{\Gamma_N} v \, \mathrm{d}s, \ \forall v \in H^1_{\Gamma_D}(D). \end{cases}$$

Example code

Minimum thermal compliance



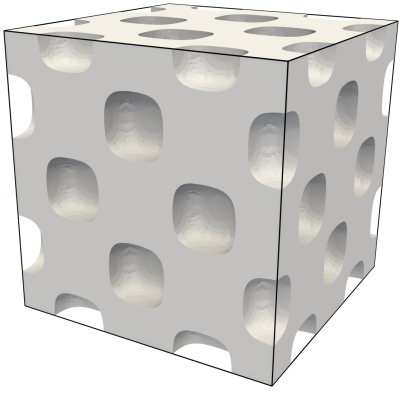
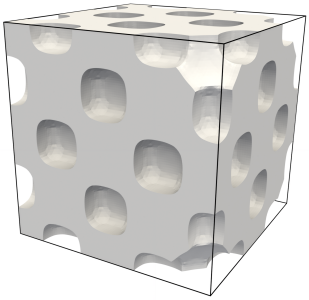
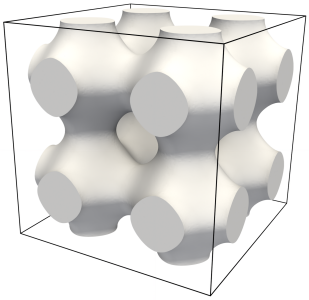


Figure: Initial structure with 150^3 elements & (4,6,6)-partitioning

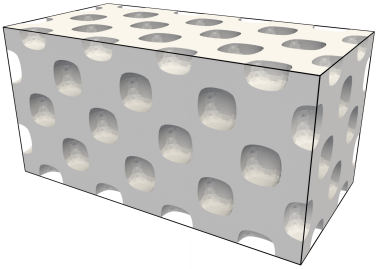
Many topology optimisation problems tractable



(a)

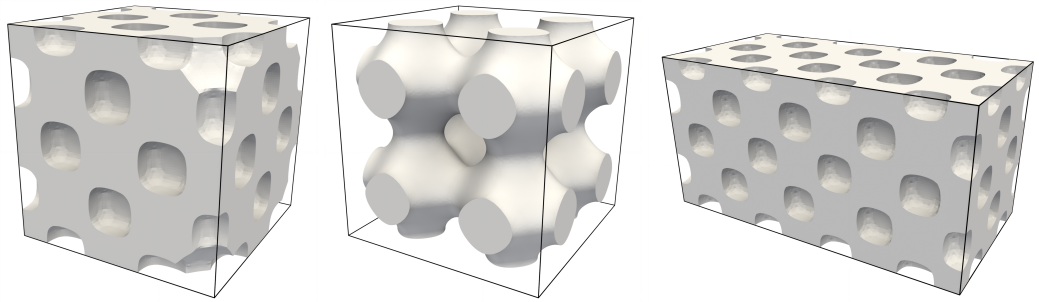


(b)



(c)

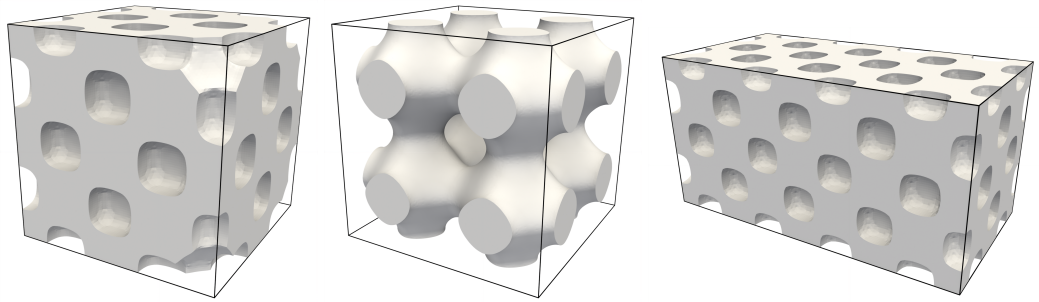
Many topology optimisation problems tractable



(a) (b) (c)

Limitations due to interface ‘smoothing’:

Many topology optimisation problems tractable

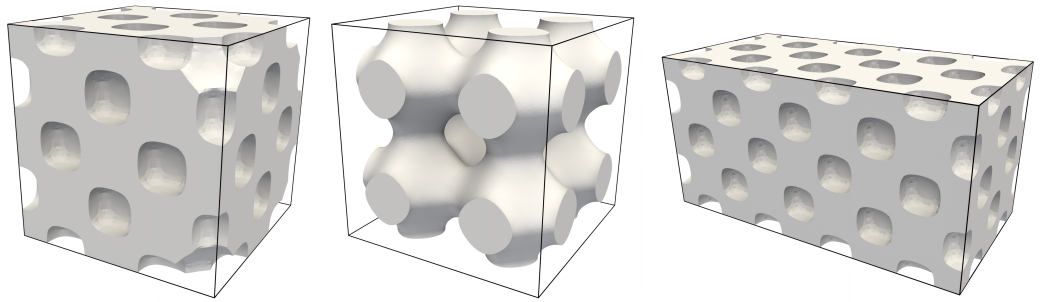


(a) (b) (c)

Limitations due to interface 'smoothing':

- Automatic differentiation only approximates shape derivatives

Many topology optimisation problems tractable



(a) (b) (c)

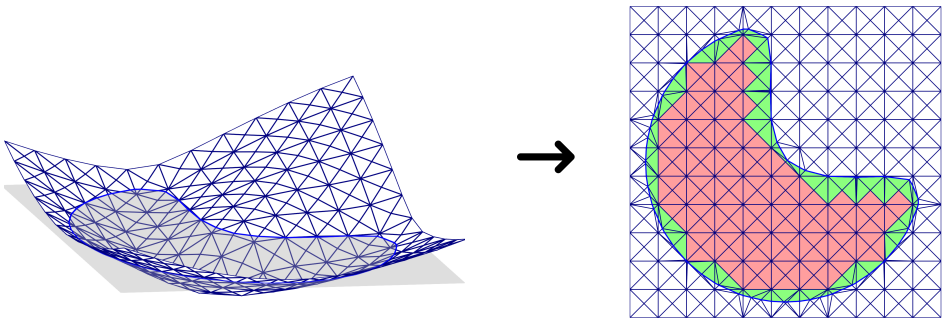
Limitations due to interface 'smoothing':

- Automatic differentiation only approximates shape derivatives
- Causes computational issues for multi-physics problems (e.g., fluid-structure interaction & piezoelectricity)

New features: Unfitted FEs and automatic shape differentiation

As part of a recent publication (Wegert et al. [14]), we:

- Extended GridapTopOpt to unfitted discretisations
- Implemented automatic shape differentiation for unfitted discretisations

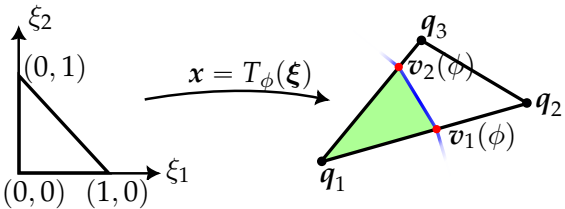


Automatic shape differentiation on unfitted discretisations

For

$$F(\phi) = \int_{\Omega(\phi)} f(\phi)\, \mathrm{d}\boldsymbol{x}$$

Need to compute directional derivative $F'(\phi)(w)$.

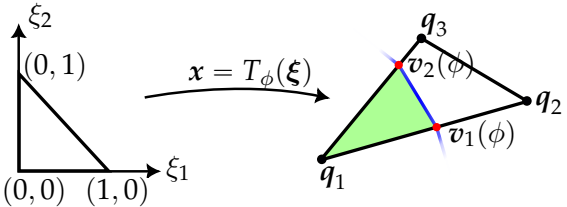


Automatic shape differentiation on unfitted discretisations

For

$$F(\phi) = \int_{\Omega(\phi)} f(\phi)\, \mathrm{d}\boldsymbol{x}$$

Need to compute directional derivative $F'(\phi)(w)$.



---

for *each* ***cut cell*** **do**

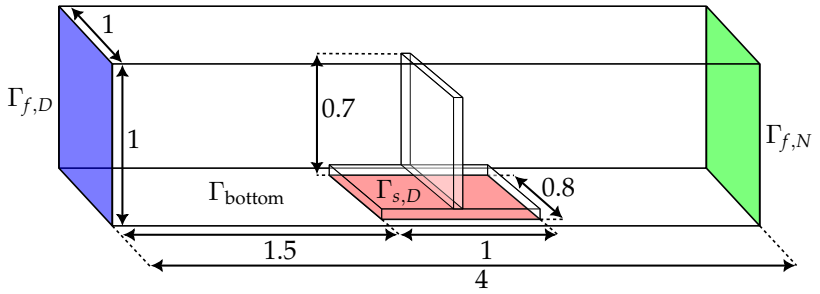
- Propagate dual numbers through the cell-wise degrees of freedom ($u_i \to u_i + \varepsilon_i$);
- Compute the nodal coordinates of the cuts $\boldsymbol{v}_1(\phi_i + \varepsilon_i)$ and $\boldsymbol{v}_2(\phi_i + \varepsilon_i)$;
- Construct the map $T_{\phi_i+\varepsilon_i}$ and the corresponding Jacobian $J_{K(\phi_i+\varepsilon_i)}$;
- Evaluate the functional $F(\phi_i + \varepsilon_i)$ and extract dual components;

end

Assemble these local vectors into the global gradient.

(Verified against existing analytic formulas and finite differences)

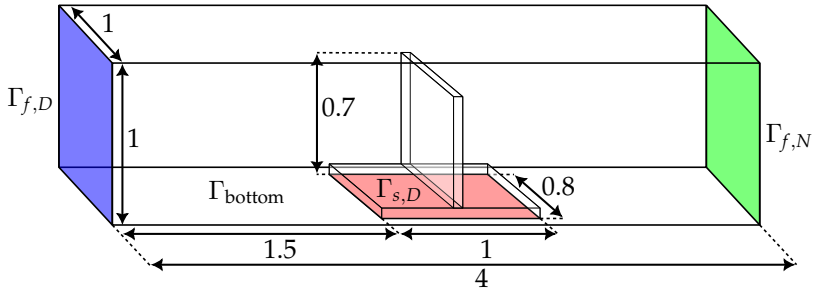
Example: Fluid-structure interaction



Similar setup to Feppon et al. [5]. Staggered scheme, i.e **one-way coupling**, between

- **Stokes**: stabilised $P_1 - P_0$ with Nitsche and face ghost-penalty stabilisation [3].
- **Elasticity**: face ghost-penalty method with linear Lagrangian elements [4].

Example: Fluid-structure interaction



$$
\begin{cases}
-\nabla \cdot \boldsymbol{\sigma}_f(\boldsymbol{u}, p) = \mathbf{0} & \text{in } \Omega_{\text{out}}(\phi), \\
\nabla \cdot \boldsymbol{u} = \mathbf{0} & \text{in } \Omega_{\text{out}}(\phi), \\
\boldsymbol{u} = \boldsymbol{u}_d & \text{on } \Gamma_{f,D}, \\
\boldsymbol{\sigma}_f(\boldsymbol{u}, p) \cdot \boldsymbol{n} = \mathbf{0} & \text{on } \Gamma_{f,N}, \\
\boldsymbol{u} = \mathbf{0} & \text{on } \Gamma(\phi) \cup \Gamma_{\text{bottom}}, \\
\boldsymbol{u} \cdot \boldsymbol{n} = 0 & \text{on } \Gamma_{\text{sides}},
\end{cases}
\longrightarrow
\begin{cases}
-\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{d}) = \mathbf{0} & \text{in } \Omega(\phi), \\
\boldsymbol{\sigma}(\boldsymbol{d}) \cdot \boldsymbol{n} = \boldsymbol{\sigma}_f(\boldsymbol{u}, p) \cdot \boldsymbol{n} & \text{in } \Gamma(\phi), \\
\boldsymbol{d} = \mathbf{0} & \text{on } \Gamma_{s,D}.
\end{cases}
$$

Example: Fluid-structure interaction

$$
\begin{aligned}
\min_{\phi \in W^h} J(\phi) &:= \int_{\Omega(\phi)} \boldsymbol{\sigma}(\boldsymbol{d}) : \boldsymbol{\varepsilon}(\boldsymbol{d}) \, \mathrm{d}\boldsymbol{x} \\
\text{s.t.} \quad & \mathrm{Vol}(\Omega(\phi)) = 0.06 \, \mathrm{Vol}(D), \\
& r_{\text{fluid}}((\boldsymbol{u}, p), (\boldsymbol{v}, q)) = 0, \ \forall (\boldsymbol{v}, q) \in V \times Q, \\
& r_{\text{elast}}((\boldsymbol{u}, p), \boldsymbol{d}, \boldsymbol{s}) = 0, \ \forall \boldsymbol{s} \in T.
\end{aligned}
$$

- Non-bodyfitted methods (e.g., smoothed interface method) struggle to resolve boundary conditions
- Analytic calculation of shape derivatives is difficult

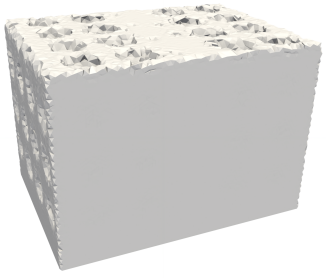
Example: Fluid-structure interaction

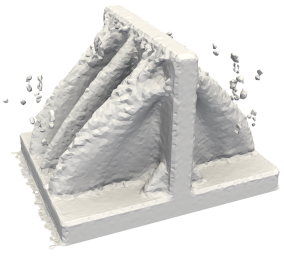
$$\min_{\phi \in W^h} J(\phi) := \int_{\Omega(\phi)} \boldsymbol{\sigma}(\boldsymbol{d}) : \boldsymbol{\varepsilon}(\boldsymbol{d}) \, \mathrm{d}\boldsymbol{x}$$

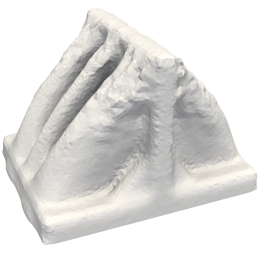
$$\text{s.t.} \quad \mathrm{Vol}(\Omega(\phi)) = 0.06 \, \mathrm{Vol}(D),$$

$$r_{\text{fluid}}((\boldsymbol{u}, p), (\boldsymbol{v}, q)) = 0, \ \forall (\boldsymbol{v}, q) \in V \times Q,$$

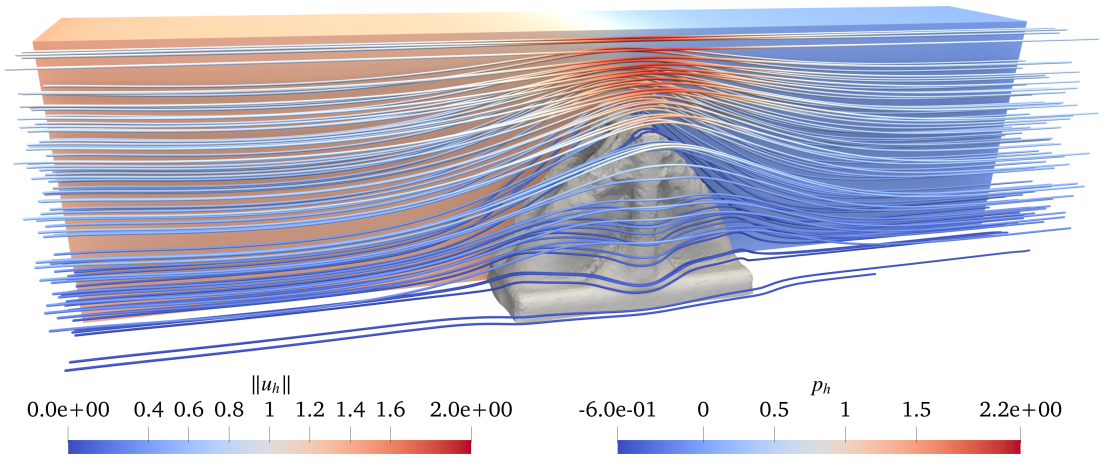
$$r_{\text{elast}}((\boldsymbol{u}, p), \boldsymbol{d}, \boldsymbol{s}) = 0, \ \forall \boldsymbol{s} \in T.$$







Example: Fluid-structure interaction

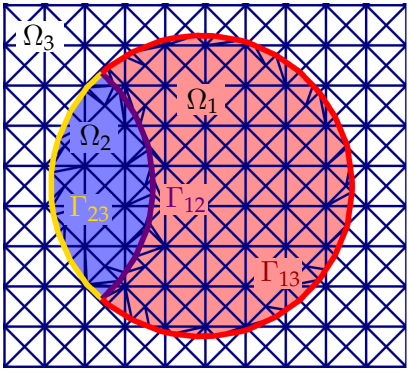


Final remarks and future work

- GridapTopOpt.jl: framework for level-set topology optimisation in the Gridap ecosystem
- Automatic differentiation tools for generic PDE-constrained functionals

Future work/features

- Multi-phase unfitted methods
- Second order AD methods
- Geometric constraints
- Support for alternate topology optimisation methods (e.g., density methods)



Acknowledgments

Australian Research Council Discovery Project (DP220102759 & DP240102104). Computational resources provided by QUT, UQ, NCI, and Pawsey.

References I

[1] Grégoire Allaire, François Jouve, and Anca-Maria Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 194(1):363–393, 2 2004. ISSN 00219991. doi: 10.1016/j.jcp.2003.09.032.

[2] Grégoire Allaire, Charles Dapogny, and François Jouve. *Shape and topology optimization*, volume 22, page 1–132. Elsevier, 2021. ISBN 978-0-444-64305-6. doi: 10.1016/bs.hna.2020.10.004.

[3] 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*, 104(7):472–501, 11 2015. ISSN 0029-5981, 1097-0207. doi: 10.1002/nme.4823.

[4] Erik Burman, Daniel Elfverson, Peter Hansbo, Mats G. Larson, and Karl Larsson. Shape optimization using the cut finite element method. *Computer Methods in Applied Mechanics and Engineering*, 328:242–261, 1 2018. ISSN 00457825. doi: 10.1016/j.cma.2017.09.005.

References II

[5] F. Feppon, G. Allaire, C. Dapogny, and P. Jolivet. Topology optimization of thermal fluid–structure systems using body-fitted meshes and parallel computing. *Journal of Computational Physics*, 417:109574, 9 2020. ISSN 00219991. doi: 10.1016/j.jcp.2020.109574.

[6] Connor N. Mallon, Aaron W. Thornton, Matthew R. Hill, and Santiago Badia. Neural Level Set Topology Optimization Using Unfitted Finite Elements. *International Journal for Numerical Methods in Engineering*, 126(6):e70004, 2025. doi: 10.1002/nme.70004.

[7] Jordi Manyer, Alberto F. Martín, and Santiago Badia. GridapSolvers.jl: Scalable multiphysics finite element solvers in Julia, journal = Journal of Open Source Software. 9(102):7162, 2024. doi: 10.21105/joss.07162. URL `https://doi.org/10.21105/joss.07162`.

References III

[8] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. Springer Science & Business Media, 1 edition, 4 2006. ISBN 978-0-387-22746-7.

[9] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-Based Fast Local Level Set Method. *Journal of Computational Physics*, 155(2): 410–438, 11 1999. ISSN 00219991. doi: 10.1006/jcph.1999.6345.

[10] Francesc Verdugo, Victor Sande, and Alberto F. Martin. GridapPETSc. `https://github.com/gridap/GridapPETSc.jl`, 2021.

[11] Michael Yu Wang, Xiaoming Wang, and Dongming Guo. A level set method for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 192(1–2):227–246, 1 2003. ISSN 00457825. doi: 10.1016/S0045-7825(02)00559-5.

References IV

[12] Zachary J. Wegert, Anthony P. Roberts, and Vivien J. Challis. A Hilbertian projection method for constrained level set-based topology optimisation. *Structural and Multidisciplinary Optimization*, 66(9):204, 9 2023. ISSN 1615-1488. doi: 10.1007/s00158-023-03663-0.

[13] Zachary J. Wegert, Jordi Manyer, Connor N. Mallon, Santiago Badia, and Vivien J. Challis. GridapTopOpt.jl: a scalable Julia toolbox for level set-based topology optimisation. *Structural and Multidisciplinary Optimization*, 68(1):22, 1 2025. ISSN 1615-1488. doi: 10.1007/s00158-024-03927-3.

[14] Zachary J. Wegert, Jordi Manyer, Connor N. Mallon, Santiago Badia, and Vivien J. Challis. Level-set topology optimisation with unfitted finite elements and automatic shape differentiation. *Computer Methods in Applied Mechanics and Engineering*, 445: 118203, 2025. ISSN 0045-7825. doi: 10.1016/j.cma.2025.118203.

Appendix: Analytic results (volume integrals)

Theorem (Berggren, 2023)

Under perturbations $\phi_t(\boldsymbol{x}) = \phi(\boldsymbol{x}) + tw(\boldsymbol{x})$*, the directional derivative of the volume integral*

$$J_1(\phi) = \int_{\Omega(\phi)} f \, \mathrm{d}\boldsymbol{x},$$

for $f \in C^0(\overline{\mathcal{T}_h})$*, satisfies*

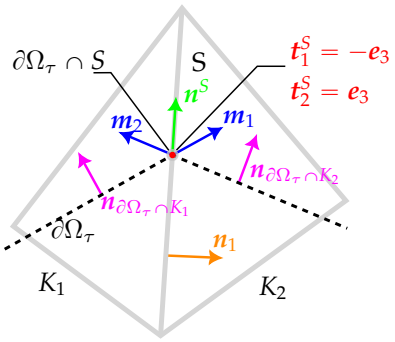
$$\mathrm{d}J_1(\phi)(w) = -\int_{\partial\Omega} f \frac{w}{|\partial_{\boldsymbol{n}}\phi|} \, \mathrm{d}s,$$

where $\partial_{\boldsymbol{n}}\phi = \boldsymbol{n} \cdot \boldsymbol{\nabla}\phi$*.*

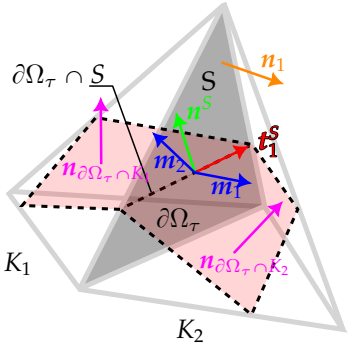
Appendix: Analytic results (Surface integrals (I))

Given the interface $S \in \mathcal{F}_h$ between two cut cells $K_1, K_2 \in \mathcal{T}_h$, we define:

- $\boldsymbol{m}_k$ the co-normal vectors perpendicular to $\boldsymbol{t}_k^S$ and $\boldsymbol{n}_{\partial\Omega_\tau \cap K_k}$
- $\boldsymbol{n}^S$ the interface normal: outward-directed from Ω, and orthogonal to $\partial\Omega \cap S$ for $d = 3$.



(a) Case $D = 2$



(b) Case $D = 3$

Appendix: Analytic results (Surface integrals (II))

Theorem (Berggren, 2023)

Under perturbations $\phi_t(\boldsymbol{x}) = \phi(\boldsymbol{x}) + tw(\boldsymbol{x})$, *if* $\partial\Omega(\phi) \cap \mathcal{V}_h = \varnothing$ *and* $\partial\Omega(\phi) \cap \partial D = \varnothing$, *the directional derivative of the surface integral*

$$J_2(\phi) = \int_{\partial\Omega(\phi)} f \, \mathrm{d}s$$

for $f \in C^1(\overline{\mathcal{T}_h})$, *satisfies*

$$\mathrm{d}J_2(\phi, w) = -\int_{\partial\Omega} \frac{\partial f}{\partial \boldsymbol{n}} \frac{w}{|\partial_{\boldsymbol{n}}\phi|} \, \mathrm{d}s - \sum_{S \in \mathscr{F}_h \setminus \partial D} \int_{\partial\Omega \cap S} \boldsymbol{n}^S \cdot [\![f\boldsymbol{m}]\!] \frac{w}{|\partial_{\boldsymbol{n}}\phi|} \, \mathrm{d}\gamma,$$

where $[\![f\boldsymbol{m}]\!] = f_1\boldsymbol{m}_1 + f_2\boldsymbol{m}_2$. *Here* f_k *is the limit* f *on* S *defined by* $f_k(\boldsymbol{x}) = \lim_{\epsilon \to 0^+} f(\boldsymbol{x} - \epsilon \boldsymbol{m}_k)$ *for* $\boldsymbol{x} \in \partial\Omega \cap S$.

Notation: $\mathcal{V}_h$ is the set of vertices in the mesh $\mathcal{T}_h$, ∂D is the boundary of the background domain, and $\mathscr{F}_h$ is the set of facets in the mesh $\mathcal{T}_h$.

Appendix: Analytic results (Surface integrals (III))

Theorem (ZJW et al., 2025)

Under perturbations $\phi_t(\boldsymbol{x}) = \phi(\boldsymbol{x}) + tw(\boldsymbol{x})$, *if* $\partial\Omega(\phi) \cap \mathcal{V}_h = \varnothing$, *the directional derivative of the surface integral*

$$J_2(\phi) = \int_{\partial\Omega(\phi)} f \, \mathrm{d}s$$

for $f \in C^1(\overline{\mathcal{T}_h})$, *satisfies*

$$\mathrm{d}J_2(\phi, w) = -\int_{\partial\Omega} \frac{\partial f}{\partial \boldsymbol{n}} \frac{w}{|\partial_{\boldsymbol{n}}\phi|} \, \mathrm{d}s - \sum_{S \in \mathscr{F}_h \setminus \partial D} \int_{\partial\Omega \cap S} \boldsymbol{n}^S \cdot [\![f\boldsymbol{m}]\!] \frac{w}{|\partial_{\boldsymbol{n}}\phi|} \, \mathrm{d}\gamma - \sum_{S \in \mathscr{F}_h \cap \partial D} \int_{\partial\Omega \cap S} \boldsymbol{n}^S \cdot (f\boldsymbol{m}) \frac{w}{|\partial_{\boldsymbol{n}}\phi|} \, \mathrm{d}\gamma.$$

Appendix: Analytic results (Flux integrals)

Lemma (ZJW et al., 2025)

Under perturbations $\phi_t(\boldsymbol{x}) = \phi(\boldsymbol{x}) + tw(\boldsymbol{x})$, *if* $\partial\Omega(\phi) \cap \mathcal{V}_h = \varnothing$ *and* $\partial\Omega(\phi) \cap \partial D = \varnothing$, *the directional derivative of the surface integral*

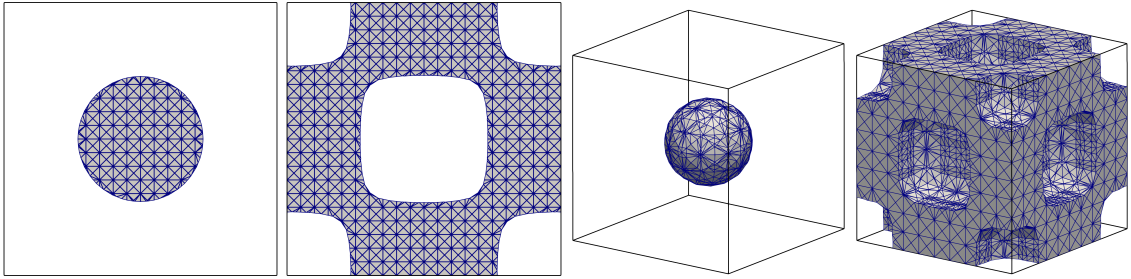
$$J_3(\phi) = \int_{\partial\Omega(\phi)} \boldsymbol{f} \cdot \boldsymbol{n} \, \mathrm{d}\boldsymbol{x},$$

for $\boldsymbol{f} \in [C^1(\overline{\mathcal{T}_h})]^d$, *satisfies*

$$\mathrm{d}J_3(\phi)(w) = -\int_{\partial\Omega} \nabla \cdot \boldsymbol{f} \frac{w}{|\partial_{\boldsymbol{n}}\phi|} \, \mathrm{ds}.$$

Appendix: Correctness results (I)

Comparison of exact shape differentiation, automatic shape differentiation, and finite difference-based derivative for:



(a) Disk (b) 2D bdry intersect (c) Sphere (d) 3D bdry intersect

We consider

$$f(x,y) = \cos(x+y)\,, \quad g(\boldsymbol{n}) = \left\| \boldsymbol{n} - \frac{\boldsymbol{\nabla} k}{\|\boldsymbol{\nabla} k\|} \right\|^2, \quad k(x,y) = x - \frac{1}{10}\sin\left(\frac{\pi y}{3}\right)$$

Appendix: Correctness results (II)

F	Level-set function	$\|\mathrm{d}F_{\mathrm{AD}} - \mathrm{d}F_{\mathrm{exact}}\|_\infty$	$\|\mathrm{d}F_{\mathrm{AD}} - \mathrm{d}F_{\mathrm{FDM}}\|_\infty$
$\int_{\Omega(\phi)} f \, \mathrm{d}\boldsymbol{x}$	Disk	3.47×10^{-17}	4.44×10^{-10}
	2D bdry intersect	7.81×10^{-18}	2.09×10^{-11}
	Sphere	3.90×10^{-18}	4.22×10^{-11}
	3D bdry intersect	1.73×10^{-18}	7.04×10^{-12}
$\int_{\partial\Omega(\phi)} f \, \mathrm{d}s$	Disk	3.47×10^{-16}	3.59×10^{-8}
	2D bdry intersect	3.05×10^{-16}	1.03×10^{-9}
	Sphere	2.64×10^{-16}	9.39×10^{-10}
	3D bdry intersect	2.92×10^{-16}	1.62×10^{-10}
$\int_{\partial\Omega(\phi)} \boldsymbol{f}\cdot\boldsymbol{n} \, \mathrm{d}s$	Disk	4.30×10^{-16}	1.52×10^{-9}
	2D bdry intersect	N/A	6.78×10^{-11}
	Sphere	3.12×10^{-17}	1.44×10^{-10}
	3D bdry intersect	N/A	1.63×10^{-11}
$\int_{\partial\Omega(\phi)} g(\boldsymbol{n}) \, \mathrm{d}s$	Disk	N/A	9.35×10^{-8}
	2D bdry intersect	N/A	2.35×10^{-9}
	Sphere	N/A	2.50×10^{-9}
	3D bdry intersect	N/A	6.09×10^{-10}

Appendix: GridapTopOpt benchmarking

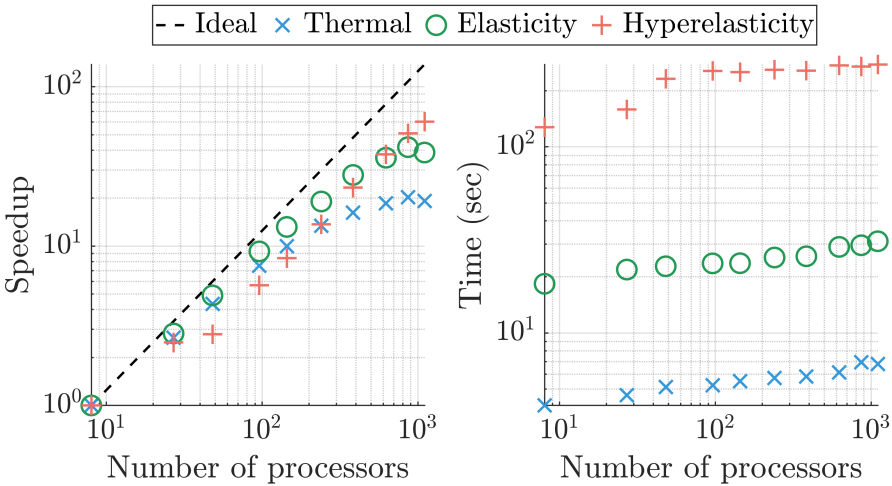


Figure: Figure (a): strong scaling benchmark. Figure (b): weak scaling benchmark.

Appendix: Some numerical example specifics

- Solved using Intel® Xeon® Platinum 8274 Processors with 4GB per core on the Gadi@NCI Australian supercomputer
- $\sim$ 4 million degrees of freedom distributed over 144 partitions
- Number of iteration and total walltime:
  - Thermal compliance problem: 167 iterations, 15 minutes
  - Inverter problem: 523 iterations, 5.5 hours
  - Elastic compliance problem: 108 iterations, 38 minutes
  - Hyperelastic compliance problem: 139 iterations, 8.8 hours
  - Inverse homogenisation: 356 iterations, 4 hours