Sandia National Laboratories

R-Adaptive Mesh Optimization for Finite Element Basis Compression

Graham Harper, **Denis Ridzal**, Tim Wildey (Sandia Labs)

FEM@LLNL

LDRD
Laboratory Directed Research and Development

October 15, 2024

U.S. DEPARTMENT OF ENERGY | NNSA

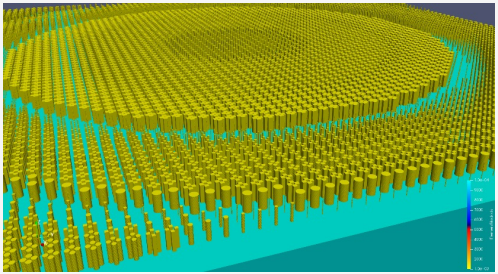
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2024-13700PE

History, circa 2021: Design of electromagnetic systems

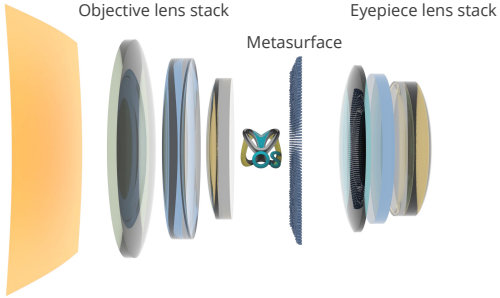
Optimal design

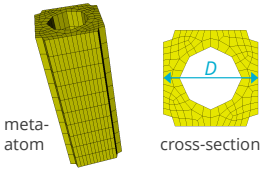
GOAL Electromagnetic devices with nanoscale structures.



EXAMPLE A micron-thin lens with millions of *meta-atoms*.
PURPOSE **Weight and space savings.**
APPLICATION Next-generation night vision systems, etc.

Mirage Project, PI: Ihab El-Kady (Sandia Labs).





For every meta-atom, find optimal "diameter" D of its cross-section, to maximize focusing efficiency of full lens.

1. Posit **spatially dependent** *material design* problem;
2. **Homogenize** material to cross section diameter.

The memory challenge

- Modern computing technology has embraced heterogeneous architectures with complex processing units that enable massive concurrency for scientific applications.
- **Frontier** at Oak Ridge National Lab and **Aurora** at Argonne National Lab are demonstrating **exascale** calculations[1].
- Many scientific applications involving the numerical solution PDEs using FEM are **limited** in their prediction fidelity by **system memory** rather than processing power ...
- ...due to significant **reductions** in the amount of system **memory per compute thread**.
- Efforts to compress data output through dimension reduction and machine learning do not reduce the amount of system memory required to run the simulations.
- New approaches are required to enable higher-fidelity simulations for predictive science.

[1] Top500.org (2024). https://www.top500.org/lists/top500/list/2024/06. (Visited on 06/30/2024).

Opportunities

- **Library-based and modular HPC applications**, through FEM software libraries like **MFEM**[2], **deal.ii**[3] and **Intrepid**[4].
- A far-reaching opportunity for the community to **tackle the memory challenge** at its source, namely, the data structures used for finite element computations.
- Few if any changes to the implementations of the PDEs that underlie HPC applications.
- If we are successful, mesh databases will join.

---

[2] R. Anderson et al. (2021). "MFEM: A modular finite element methods library". In: *Computers & Mathematics with Applications* 81, pp. 42–74. DOI: 10.1016/j.camwa.2020.06.009.

[3] W. Bangerth, R. Hartmann, and G. Kanschat (2007). "deal.II–a general-purpose object-oriented finite element library". In: *ACM Transactions on Mathematical Software (TOMS)* 33.4, 24–es. DOI: 10.1145/1268776.1268779.

[4] P. Bochev et al. (2012). "Solving PDEs with Intrepid". In: *Scientific Programming* 20.2, pp. 151–180. DOI: 10.3233/SPR-2012-0340.

Related work

- Matrix-free approaches, such as **Jacobian-free Newton Krylov** methods[5] and MFEM's **partial assembly**, seek to avoid storage of Jacobians of PDE residuals and utilize matrix-vector products for linear algebra operations.
- Constructing effective **matrix-free preconditioners** is challenging and remains an active area of research[6][7].

[5] D. A. Knoll and D. E. Keyes (2004). "Jacobian-free Newton–Krylov methods: a survey of approaches and applications". In: *Journal of Computational Physics* 193.2, pp. 357–397. DOI: 10.1016/j.jcp.2003.08.010.

[6] D. A. May, J. Brown, and L. Le Pourhiet (2015). "A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow". In: *Computer methods in applied mechanics and engineering* 290, pp. 496–523. DOI: 10.1016/j.cma.2015.03.014.

[7] T. C. Clevenger and T. Heister (2021). "Comparison between algebraic and matrix-free geometric multigrid for a Stokes problem on adaptive meshes with variable viscosity". In: *Numerical Linear Algebra with Applications* 28.5, e2375. DOI: 10.1002/nla.2375.

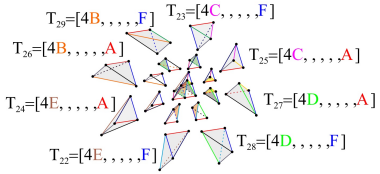
Related work

- Other data reduction techniques such as **sum factorizations**[8][9][10] seek to exploit structure to reduce the cost in storing or computing finite element basis functions.
- Most beneficial to three-dimensional problems with high-order function approximations.
- Elsewhere, data reduction techniques have been used with evolutionary PDEs to enable adjoint-based PDE-constrained optimization through **checkpointing**[11].

[8] M. Ainsworth, G. Andriamaro, and O. Davydov (2011). "Bernstein–Bézier finite elements of arbitrary order and optimal assembly procedures". In: *SIAM Journal on Scientific Computing* 33.6, pp. 3087–3109. DOI: 10.1137/11082539X.
[9] A. Bressan and S. Takacs (2019). "Sum factorization techniques in isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 352, pp. 437–460. DOI: 10.1016/j.cma.2019.04.031.
[10] J. Mora and L. Demkowicz (2019). "Fast integration of DPG matrices based on sum factorization for all the energy spaces". In: *Computational Methods in Applied Mathematics* 19.3, pp. 523–555. DOI: 10.1515/cmam-2018-0205.
[11] A. Griewank (1992). "Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation". In: *Optimization Methods and Software* 1.1, pp. 35–54. DOI: 10.1080/10556789208805505.

Our complementary contributions

- We **detect and exploit redundancies** in the **computational mesh**, based on **cell shapes**.
- Additionally, we **enhance mesh redundancies**, by moving mesh nodes without changes to the mesh topology, i.e., through **r-adaptivity**.
- A goal is to enable **clustering** of mesh cells into a relatively small number of classes based on shape identity or, more broadly, **shape similarity**[12][13] …



…important related ideas in mesh generation and refinement: **Finite number of similarity classes, $\leq$ 37, in longest edge bisection of tetrahedra.**

- Our challenge is different: **maintain the topology and quality of a given mesh**.

[12] J. P. Suárez, A. Trujillo, and T. Moreno (2021). "Computing the exact number of similarity classes in the longest edge bisection of tetrahedra". In: *Mathematics* 9.12, p. 1447. DOI: 10.3390/math9121447.

[13] A. Trujillo-Pino, J. P. Suárez, and M. A. Padrón (2024). "Finite number of similarity classes in Longest Edge Bisection of nearly equilateral tetrahedra". In: *Applied Mathematics and Computation* 472, p. 128631. DOI: 10.1016/j.amc.2024.128631.

Contribution No. 1: Detect and exploit mesh redundancy

- Cost of storing finite element quantities, such as cell Jacobians, scales linearly with the number of mesh elements, N.
- We aim to reduce the $\mathcal{O}(N)$ storage cost to $\mathcal{O}(m)$, where $m \ll N$.
- We develop a **dictionary-based data compression** scheme for problems where a given mesh contains this redundant structure.
- The scheme compresses the evaluations of finite element basis functions at quadrature points, which are linked directly to cell geometries.
- The dictionary achieves reductions in storage of more than 99% for meshes with redundant structure, enabling **billion-element simulations** in under 100 gigabytes of memory.

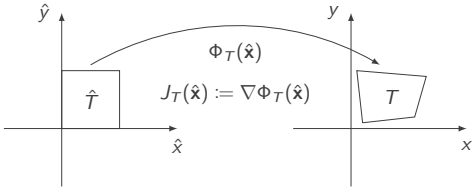
FEM background

- An important computational kernel in FEM:

$$\int_T k(\mathbf{x}) D^{\alpha_1} \phi_i(\mathbf{x}) D^{\alpha_2} \phi_j(\mathbf{x}) \, d\mathbf{x},$$

  where T is a domain that corresponds to a mesh cell in a mesh $\mathcal{T}_h$ of the domain Ω, $k(\mathbf{x})$ is some problem coefficient matrix, α_1 and α_2 are alpha-indices for the derivative operator D, and $\phi_i(\mathbf{x})$ and $\phi_j(\mathbf{x})$ are finite element basis functions.

- Integrals are evaluated using quadrature and basis functions on a reference domain $\hat{T}$.



- $\hat{T}$ is related to T by the **reference to physical map** $\Phi_T : \hat{T} \to T$, where $\mathbf{x}$ refers to physical coordinates on T and $\hat{\mathbf{x}}$ refers to reference coordinates on $\hat{T}$.

FEM background

- For a function $f : T \to \mathbb{R}$, we use $\hat{f} : \hat{T} \to \mathbb{R}$ to define the **pullback** $\hat{f}(\hat{\mathbf{x}}) := f(\Phi_T(\hat{\mathbf{x}}))$.
- Conversely, one defines f as the **pushforward** of $\hat{f}$ by $f(\mathbf{x}) = \hat{f}(\Phi_T^{-1}(\mathbf{x}))$.
- Example: Evaluation of $\nabla f(\mathbf{x})$ involves transforming a vector field to the reference domain, which requires the application of the covariant transformation[14]

$$\nabla f(\mathbf{x}) = J_T^{-\mathsf{T}}(\hat{\mathbf{x}}) \nabla \hat{f}(\hat{\mathbf{x}}).$$

- In $\mathbb{R}^3$, the **de Rham complex** with the associated spaces and transforms is

$$\begin{array}{ccccccc} H^1(\Omega) & \xrightarrow{\nabla} & H(\mathrm{curl};\Omega) & \xrightarrow{\nabla\times} & H(\mathrm{div};\Omega) & \xrightarrow{\nabla\cdot} & L^2(\Omega) \\ I(\hat{\mathbf{x}}) & & J^{-\mathsf{T}}(\hat{\mathbf{x}}) & & \det(J(\hat{\mathbf{x}}))^{-1} J(\hat{\mathbf{x}}) & & \det(J(\hat{\mathbf{x}}))^{-1} \end{array}$$

- Example:

$$\int_T \nabla\phi_i(\mathbf{x}) \cdot \nabla\phi_j(\mathbf{x})\, d\mathbf{x} = \int_{\hat{T}} \left(J_T^{-\mathsf{T}}(\hat{\mathbf{x}}) \nabla\hat{\phi}_i(\hat{\mathbf{x}}) \right) \cdot \left(J_T^{-\mathsf{T}}(\hat{\mathbf{x}}) \nabla\hat{\phi}_j(\hat{\mathbf{x}}) \right) \det J_T(\hat{\mathbf{x}})\, d\hat{\mathbf{x}}.$$

[14] D. Boffi, F. Brezzi, M. Fortin, et al. (2013). *Mixed finite element methods and applications*. Vol. 44. Springer. DOI: 10.1007/978-3-642-36519-5, c.f. §2.1.3, 2.1.4.

FEM background

- We can **trade computation for storage** of basis functions and their derivatives.

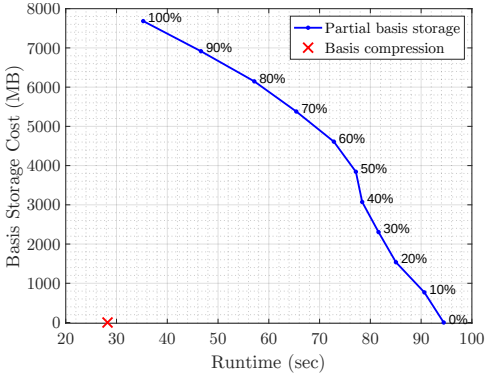


Figure: Tradeoffs between storing and recomputing basis data for a heat transfer simulation, where we show the percentage of basis data retained across the mesh. For the given mesh, containing significant redundancy, the cost of our proposed basis compression scheme is denoted by the × mark.

FEM background

- Two mesh cells T_1 and T_2 with equal Jacobians $J_{T_1} = J_{T_2}$ must have mappings Φ_{T_1} and Φ_{T_2} which differ by only a constant.
- I.e., T_1 is a **translation** of T_2 if and only if $J_{T_1} = J_{T_2}$, i.e., T_1 and T_2 have identical *shapes*.

Lemma

Let $\hat{\phi} \in \mathcal{S}$, with $\mathcal{S} \in \{H^1(\hat{T}), H(\text{curl}; \hat{T}), H(\text{div}; \hat{T}), L^2(\hat{T})\}$, be a finite element basis function on the reference domain. Let $D \in \{1, \text{grad}, \text{curl}, \text{div}\}$ be an operator so that $D\hat{\phi} \in L^2(\hat{T})$. Then if two mesh cells T_1 and T_2 have the same shape, the physical values $D\phi$ are equal and have equal integrals,

$$\int_{T_1} D\phi(\mathbf{x})\, d\mathbf{x} = \int_{T_2} D\phi(\mathbf{x})\, d\mathbf{x}.$$

Mesh examples with redundancy

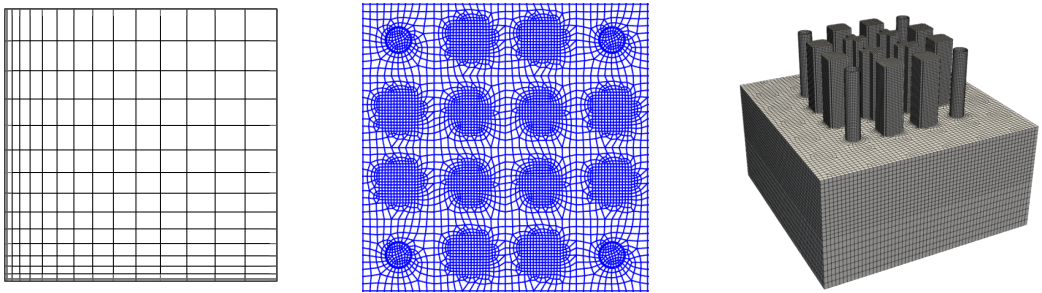


Figure: Three examples of mesh classes with redundancy: (left) structured/graded, (middle) geometrically patterned, and (right) extruded. The latter two meshes comprise large clusters of identical cell shapes. The cells in the graded mesh have shapes that are identical up to a dimensional scaling.

Compression algorithm to detect and exploit redundancy

- A list of "unique" cells is constructed by looping over mesh cells T_i and comparing them to cell T_j in the list of unique cells.
- A distance $d(T_i, T_j)$ is computed using the cell Jacobians,

$$d(T_i, T_j) = \frac{\sqrt{\sum_{k=1}^{q} w_k \|J_{T_i}(\hat{\mathbf{x}}_k) - J_{T_j}(\hat{\mathbf{x}}_k)\|_F^2}}{\sqrt{\sum_{k=1}^{q} w_k \|J_{T_j}(\hat{\mathbf{x}}_k)\|_F^2}},$$

  where $\hat{\mathbf{x}}_k$ denote the quadrature points on the reference domain, with weights w_k, $k = 1, 2, \ldots, q$, and $\|\cdot\|_F$ is the Frobenius norm.
- If $d(T_i, T_j) < \varepsilon$, then a match is found, and basis evaluations on T_i are replaced by the basis evaluations on T_j.
- Let ε_0 be the largest acceptable compression tolerance for a given application without a significant impact on accuracy, cf. Cea's Lemma, Strang's Lemma. We say that the compression is **lossless** when $\varepsilon \leq \varepsilon_0$, and if $\varepsilon > \varepsilon_0$ we refer to the compression as **lossy**.

Compression algorithm to detect and exploit redundancy

Algorithm 1. Construction of the mesh dictionary.

Input: Mesh $\mathcal{T}_h = \{T_1, T_2, \ldots, T_N\}$, tolerance ε.
$\mathcal{B}_\varepsilon \leftarrow \{\}$, $\psi_\varepsilon \leftarrow \vec{0} \in \mathbb{N}^N$
for $i = 1, 2, \ldots, N$ **do**
    match$\leftarrow$`false`
    for $T_j \in \mathcal{B}_\varepsilon$ **do**
        if $d(T_i, T_j) < \varepsilon$ **then**
            match$\leftarrow$`true`; $\psi_\varepsilon(i) \leftarrow j$; **break**
        end if
    end for
    if match=`false` **then**
        append T_i to $\mathcal{B}_\varepsilon$; $\psi_\varepsilon(i) \leftarrow |\mathcal{B}_\varepsilon|$
    end if
end for
Output: Dictionary of cells, $\mathcal{B}_\varepsilon$, with lookup indices $\psi_\varepsilon \in \mathbb{N}^N$.

Compression ratio

- The **compression ratio** of a mesh is the ratio of the number of saved Jacobian evaluations and the total number of mesh cells,

$$\varrho(\mathcal{T}_h; \varepsilon) := \frac{N - |\mathcal{B}_\varepsilon|}{N}.$$

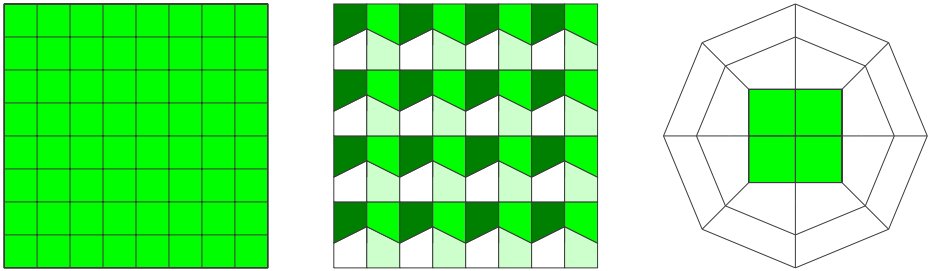
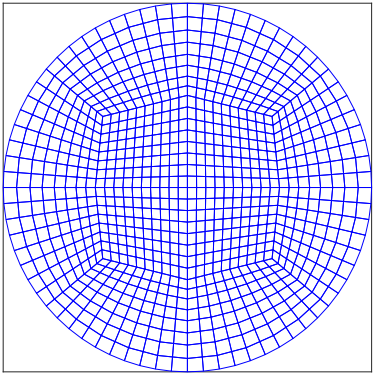


Figure: Illustration of (left) a structured square mesh with one unique shape and a lossless compression ratio ($\varepsilon = \varepsilon_0$) of 98.4% (middle) a trapezoidal mesh with only four unique shapes and a lossless compression ratio of 93.75%, and (right) a circle mesh with a lossless compression ratio of 15%. Cells with the same shape within a single mesh are highlighted with the same color.

Compression curve

- We plot the compression ratio $\varrho(\mathcal{T}_h; \varepsilon)$, as a function of ε.



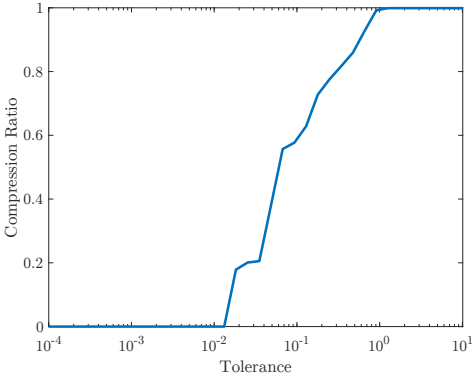
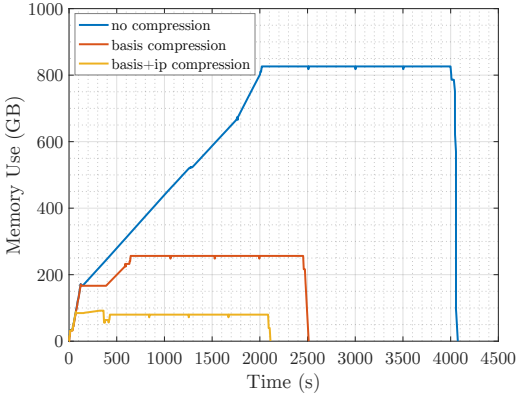


Figure: Compression curve for a circle mesh; no compression is observed for $\varepsilon < 10^{-2}$.

Memory savings for a heat transfer example

- Memory profiles and timing data for a heat transfer simulation in **MrHyDE**[15] are shown.
- Uniform $32,000 \times 32,000$ mesh of square cells and a compression tolerance of $\varepsilon = 10^{-10}$.
- Basis compression reduces the memory cost of a billion sets of basis function data to **one**.



[15] *MrHyDE library* (n.d.). Accessed: 09/25/2024. URL: https://github.com/sandialabs/MrHyDE.

Contribution No. 2: Enhance mesh redundancy

- We develop a new **r-adaptive mesh optimization** scheme to enhance the redundancy.
- Our scheme combines a novel adaptation of ***k*-medoids** clustering[16] with a sparsity-promoting optimization formulation.
- The latter is inspired by the **constrained optimization alternative**[17] to r-adaptive schemes based on solving the **Monge-Ampère equation**[18][19].
- Inequality constraints on the cell volumes, to prevent tangling and ensure minor deviations from the original mesh, are handled using a **scalable augmented Lagrangian method**[20].

---

[16] H.-S. Park and C.-H. Jun (2009). "A simple and fast algorithm for K-medoids clustering". In: *Expert systems with applications* 36.2, pp. 3336–3341. DOI: 10.1016/j.eswa.2008.01.039.

[17] M. D'Elia et al. (2016). "Optimization-based mesh correction with volume and convexity constraints". In: *Journal of Computational Physics* 313, pp. 455–477. DOI: 10.1016/j.jcp.2016.02.050.

[18] J.-F. Cossette, P. K. Smolarkiewicz, and P. Charbonneau (2014). "The Monge–Ampère trajectory correction for semi-Lagrangian schemes". In: *Journal of Computational Physics* 274, pp. 208–229. DOI: 10.1016/j.jcp.2014.05.016.

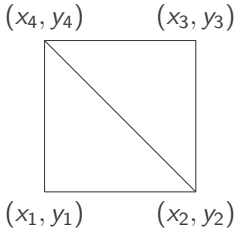
[19] P. A. Browne et al. (2014). "Fast three dimensional r-adaptive mesh redistribution". In: *Journal of Computational Physics* 275, pp. 174–196. DOI: 10.1016/j.jcp.2014.06.009.

[20] H. Antil, D. P. Kouri, and D. Ridzal (2023). "ALESQP: An augmented Lagrangian equality-constrained SQP method for optimization with general constraints". In: *SIAM Journal on Optimization* 33.1, pp. 237–266. DOI: 10.1137/20M1378399.

Problem formulation: Quantities of interest

- We focus on the **quadrilateral** case with straight line geometry.
- Mesh $\mathcal{T}_h = \{T_1, T_2, \ldots, T_N\}$ of a domain $\Omega \subset \mathbb{R}^2$, with n_p vertices $\hat{\mathbf{p}} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_{n_p}\}$.
- Dictionary $\mathcal{B}_\varepsilon$, obtained by applying Algorithm 1 to $\mathcal{T}_h$ for a tolerance ε.
- **Goal:** Produce a mesh $\mathcal{T}_{h,r}$ such that the resulting $\mathcal{B}_{\varepsilon,r}$ is smaller than $\mathcal{B}_\varepsilon$.
- Requirement: Maintain topology, with relatively small mesh motion.
- Optimization variables: Vertices $\mathbf{p} = \{p_1, p_2, \ldots, p_{n_p}\}$, where $p_i = (x_i, y_i)$, $i = 1, 2, \ldots, n_p$.
- Let the double-indexing $p_{i,j} = (x_{i,j}, y_{i,j})$ denote vertex j of mesh cell i, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, n_{cv}$, with n_{cv} denoting the number of vertices in each cell.

Problem formulation: Quantities of interest



- Split the quadrilateral into two triangles using the vertex index triples $(1, 2, 4)$ and $(3, 4, 2)$.
- On a general mesh cell T_i, $1 \leq i \leq N$, we define

$$c_{i,x2} = x_{i,2} - x_{i,1}, \qquad d_{i,x4} = x_{i,4} - x_{i,3},$$
$$c_{i,x4} = x_{i,4} - x_{i,1}, \qquad d_{i,x2} = x_{i,2} - x_{i,3},$$
$$c_{i,y2} = y_{i,2} - y_{i,1}, \qquad d_{i,y4} = y_{i,4} - y_{i,3},$$
$$c_{i,y4} = y_{i,4} - y_{i,1}, \qquad d_{i,y2} = y_{i,2} - y_{i,3}.$$

- The two triangle Jacobians composed of these scalars define a computationally efficient proxy for the Jacobian of the quadrilateral.
- Row vector $K_i(\mathbf{p})$, $i = 1, 2, \ldots, N$, consisting of the Jacobian entries,

$$K_i(\mathbf{p}) = [c_{i,x2}, c_{i,x4}, d_{i,x4}, d_{i,x2}, c_{i,y2}, c_{i,y4}, d_{i,y4}, d_{i,y2}].$$

- For each cell T_i in the mesh, we also associate a "target" shape μ_i, $i = 1, 2, \ldots, N$, which is a row vector of eight entries as well, analogous to $K_i(\mathbf{p})$.
- Our goal is to reduce the misfit between $K(\mathbf{p})$ and μ, where $K(\mathbf{p}), \mu \in \mathbb{R}^{N \times 8}$.

Problem formulation: Objective function

- We focus on lossless finite element basis compression.
- Ideally, this is a sparse matching problem between the rows of $K(\mathbf{p})$ and μ, which would **maximize the number of zero rows** in $K(\mathbf{p}) - \mu$.
- Approaches enforcing the so-called **group sparsity**, such as those using the mixed ℓ_1/ℓ_q norm, $q > 1$, have been suggested[21], with **non-smooth quantities of interest** of the form

$$\sum_{i=1}^{N} \|K_i(\mathbf{p}) - \mu_i\|_q.$$

- The non-smoothness, induced by the norm, is difficult to handle algorithmically.
- In contrast, the functional $\sum_{i=1}^{N} \|K_i(\mathbf{p}) - \mu_i\|_2^2$, is smooth and relatively easy to minimize using derivative-based optimization methods, but any notion of group sparsity may be lost.
- Minimizing it typically produces no zero-misfit terms whatsoever, i.e., no lossless compression, unless $K_i(\mathbf{p}) - \mu_i = 0$ is achievable at the optimum for all $i = 1, 2, \ldots, N$.

[21] F. Bach et al. (2012). "Structured sparsity through convex optimization". In: *Statistical Science* 27.4, pp. 450–468. DOI: 10.1214/12-STS394.

Problem formulation: Objective function

- This challenge also motivates a key idea:

$$\text{identify a subset of cell indices, } \mathcal{Z} \subseteq \{1, 2, \ldots, N\},$$
$$\text{such that } K_i(\mathbf{p}) - \mu_i = 0 \text{ is achievable for all } i \in \mathcal{Z}.$$

- Given a set $\mathcal{Z}$, the functional to minimize is

$$\sum_{i \in \mathcal{Z}} \|K_i(\mathbf{p}) - \mu_i\|_2^2,$$

leading to our primary "lossless" objective functional

$$L(\mathbf{p}) = \sum_{i=1}^{N} \alpha_i \|K_i(\mathbf{p}) - \mu_i\|_2^2, \quad \alpha_i \in \{0, 1\}, \quad \text{for } i = 1, 2, \ldots, N,$$

where $\alpha_i = 1$ if $i \in \mathcal{Z}$ and $\alpha_i = 0$ otherwise.

- To identify the set $\mathcal{Z}$ we will also consider a relaxed version, namely

$$L_r(\mathbf{p}) = \sum_{i=1}^{N} \alpha_i \|K_i(\mathbf{p}) - \mu_i\|_2^2, \quad \alpha_i \in \mathbb{R}_0^+, \quad \text{for } i = 1, 2, \ldots, N.$$

Problem formulation: Constraints

- We require that the mesh modifications, due to minimizing our primary objective function $L(\mathbf{p})$, maintain the quality of the **finite element approximation** given by the original mesh.
- Among other requirements, mesh cells should **not become tangled or inverted**.
- We **restrict the volume change** for each mesh cell. Optionally: maintain **cell convexity**.
- We eliminate nodes defining important geometric features (e.g., boundaries).
- Let $v_i(\mathbf{p})$ be the volume of cell i, $i = 1, 2, \ldots N$, given by the shoelace formula

$$v_i(\mathbf{p}) = \frac{1}{2} \sum_{j=1}^{n_{cv}} x_{i,j} y_{i,j+1} - x_{i,j+1} y_{i,j}, \quad i = 1, 2, \ldots, N,$$

where $n_{cv} = 4$ for quadrilaterals, and index $n_{cv} + 1 = 5$ is associated with $5 \pmod{n_{cv}}$.

Problem formulation: Constraints

- Let $v_{\min} = \min_{i\in\{1,2,\dots N\}} v_i(\hat{\mathbf{p}})$ and $v_{\max} = \max_{i\in\{1,2,\dots N\}} v_i(\hat{\mathbf{p}})$.
- We define the lower and upper volume bounds, v_i^{lo} and v_i^{up}, $i = 1, 2, \dots, N$, respectively, in two ways: using **global** quantities, where for $0 < \gamma < 1$,

$$v_i^{\text{lo}} = (1-\gamma)v_{\min} \quad \text{and} \quad v_i^{\text{up}} = (1+\gamma)v_{\max}, \quad i = 1, 2, \dots N,$$

  or using **local** quantities, where

$$v_i^{\text{lo}} = (1-\gamma)v_i(\hat{\mathbf{p}}) \quad \text{and} \quad v_i^{\text{up}} = (1+\gamma)v_i(\hat{\mathbf{p}}), \quad i = 1, 2, \dots N.$$

- We seek $\mathbf{p} \in \mathbb{R}^{n_p}$ that solves

$$\underset{\mathbf{p}}{\text{minimize}} \quad L(\mathbf{p}) \tag{1a}$$

$$\text{subject to} \quad v^{\text{lo}} \le v(\mathbf{p}) \le v^{\text{up}}. \tag{1b}$$

Lemma

The feasible set defined in (1b) *is nonempty for both definitions of the bounds, global and local.*

Algorithms

$$\begin{aligned} \underset{\mathbf{p}}{\text{minimize}} \quad & L(\mathbf{p}) = \sum_{i=1}^{N} \alpha_i \|K_i(\mathbf{p}) - \mu_i\|_2^2 \\ \text{subject to} \quad & v^{\text{lo}} \leq v(\mathbf{p}) \leq v^{\text{up}}, \text{ and} \\ & \alpha_i \in \{0, 1\}, \ i = 1, 2, \ldots, N. \end{aligned}$$

- **Optimization algorithm** to move the mesh nodes $\mathbf{p}$.
- **Clustering algorithm** to define the cell targets μ.
- **Bracketing algorithm** to define the weights α.

Optimization algorithm: ALESQP

- ALESQP uses an **augmented Lagrangian** penalty to handle inequality constraints.
- General inequality constraints are transformed to equality constraints and bounds using **slack variables**, $s \in \mathbb{R}^N$, $s_i = v_i(\mathbf{p})$, $i = 1, 2, \ldots, N$:

$$\begin{aligned} \underset{\mathbf{p},s}{\text{minimize}} \quad & L(\mathbf{p}) \\ \text{subject to} \quad & v(\mathbf{p}) - s = 0, \\ & v^{\text{lo}} \leq s \leq v^{\text{up}}. \end{aligned}$$

- The subproblems in ALESQP penalize inequality constraints, and maintain equalities:

$$\underset{\mathbf{p},s}{\text{minimize}} \quad L(\mathbf{p}) + \frac{1}{2r} \left\| r \left(\frac{\lambda}{r} + s - \max\left(\min\left(\frac{\lambda}{r} + s, v^{\text{lo}} \right), v^{\text{up}} \right) \right) \right\|_2^2 \tag{2a}$$

$$\text{subject to} \quad v(\mathbf{p}) - s = 0, \tag{2b}$$

  where $r > 0$ is a scalar computed by ALESQP, $\lambda \in \mathbb{R}^N$ is a Lagrange multiplier corresponding to the slack variables s, and min and max are applied elementwise.

- Objective function in (2a) is differentiable, despite the non-smooth min and max functions.

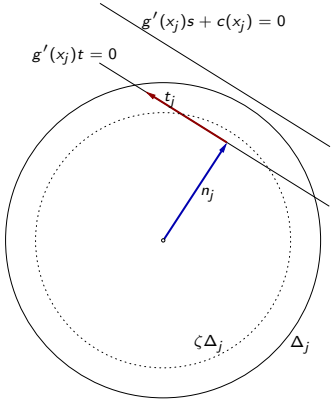
Subproblem solver: Composite-step trust-region SQP[22]

- **Trust-region step**: $s_j = n_j + t_j$

- **Quasi-normal step** n_j:

  reduces linear infeasibility

$$\begin{aligned} \min_{n \in X} \quad & \|g'(x_j)n + g(x_j)\|^2 \\ \text{s.t.} \quad & \|n\| \leq \zeta \Delta_j \end{aligned}$$



- **Tangential step** t_j:

  improves optimality while staying in the null space of the linearized constraints

$$\begin{aligned} \min_{t \in X} \quad & \frac{1}{2}\langle \nabla_{xx}\mathscr{L}(x_j, \zeta_j)(t + n_j), t + n_j\rangle + \langle \nabla_x \mathscr{L}(x_j, \zeta_j), t + n_j\rangle + \mathscr{L}(x_j, \zeta_j) \\ \text{s.t.} \quad & g'(x_j)t = 0\,, \quad \|t + n_j\| \leq \Delta_j \end{aligned}$$

[22] M. Heinkenschloss and D. Ridzal (2014). “A matrix-free trust-region SQP method for equality-constrained optimization”. In: *SIAM Journal on Optimization* 24.3, pp. 1507–1541. DOI: 10.1137/130921738.

The key computational kernel: Augmented system

1. Compute **quasi-normal step** n_j using **Powell dogleg**, where for the Newton step we solve an augmented system inexactly.
2. Solve **tangential subproblem** for $\widetilde{t_j}$ via **projected Steihaug-Toint CG**, where the projections are computed by solving augmented systems inexactly.
3. Restore linearized feasibility, for **tangential step** t_j, via another inexact projection.
4. Update **Lagrange multipliers** ζ_{j+1} by solving an augmented system inexactly.
5. Evaluate progress.

Augmented System

$$\begin{pmatrix} I & g'(x_j)^T \\ g'(x_j) & 0 \end{pmatrix} \begin{pmatrix} y^1 \\ y^2 \end{pmatrix} = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix} + \begin{pmatrix} e^1 \\ e^2 \end{pmatrix}$$

- The size of $(e^1 \ e^2)$ is governed by the progress of the optimization algorithm:

$$\|e^1\| + \|e^2\| \leq \mathscr{T}\left(\|b^1\|, \|b^2\|, \|y^1\|, \Delta_j\right)$$

- In this application, with variables $(\mathbf{p}, s)$, augmented systems are:

$$\begin{pmatrix} I & g'^T \\ g' & 0 \end{pmatrix} \rightarrow \begin{pmatrix} I & & g_{\mathbf{p}}'^T \\ & I & g_s'^T \\ g_{\mathbf{p}}' & g_s' & 0 \end{pmatrix}.$$

The key computational kernel: Augmented system

- Here $g'_{\mathbf{p}}$ corresponds to the Jacobian $v'_{\mathbf{p}}(\mathbf{p})$ and g'_s is $-I$.
- As $v(\mathbf{p})$ is defined by the shoelace formula, the entries of $g'_{\mathbf{p}}$ take the form

$$y_{i,k+1} - y_{i,k-1} \text{ and } x_{i,k-1} - x_{i,k+1}.$$

- The structure of $g'_{\mathbf{p}}$ appears similar to a **discrete divergence** on the mesh.
- Similarly, ${g'_{\mathbf{p}}}^{\mathsf{T}}$ resembles a **discrete gradient**.
- So, the Schur complement of the augmented system matrix is

$$S = g'_{\mathbf{p}} {g'_{\mathbf{p}}}^{\mathsf{T}} + I,$$

  where the $g'_{\mathbf{p}} {g'_{\mathbf{p}}}^{\mathsf{T}}$ matrix resembles a **discrete Laplacian**.
- The augmented system is easily solved using MINRES or GMRES, with no preconditioning.
- For problems with large domain lengthscales, where the term $g'_{\mathbf{p}} {g'_{\mathbf{p}}}^{\mathsf{T}}$ may dominate, multigrid methods are very effective in approximating the application of S^{-1}.

Clustering algorithm: k-medoids to compute cell targets

- k-medoids algorithm is used to obtain lossless compression.
- Recall: medoids are **representatives** of a data cluster, which minimize the distance to all other data points in the cluster.
- We initialize our algorithm by drawing k samples randomly from N data points, without replacement, the indices of which are denoted by the permutation σ_k^N.
- After performing clustering on $K(\mathbf{p})$ and obtaining the cluster assignments ψ and indices of cluster medoids C, the shape targets are defined by the corresponding medoid for each data point, $\mu_i = K_{C_{\psi_i}}(\mathbf{p})$, $i = 1, 2, \ldots, N$.

Algorithm 2. Shape clustering.

Input: Jacobians $J \in \mathbb{R}^{N\times 8}$, number of clusters k.
Initialization: $\psi \leftarrow \vec{0} \in \mathbb{N}^N$, conv $\leftarrow$ `false`, draw permutation $C \leftarrow \sigma_k^N \in \mathbb{N}^k$.
while not conv **do**
  for $i = 1, 2, \ldots, N$ **do** // Assign ψ_i to closest cluster index based on $\|\cdot\|_{\ell_2}^2$.
    $d \leftarrow \vec{0} \in \mathbb{R}^k$
    for $j = 1, 2, \ldots, k$ **do**
      $d_j \leftarrow \|J(i,:) - J(C_j,:)\|_{\ell_2}^2$
    end for
    $\psi_i \leftarrow s : d_s = \min(d)$
  end for
  $C_{\text{old}} \leftarrow C$
  for $j = 1, 2, \ldots, k$ **do** // Assign C_j to global index for cluster medoid.
    $\mathcal{I} = \{l : \psi_l = j\}$; $m = |\mathcal{I}|$
    $v = \vec{0} \in \mathbb{N}^m$; $l = 1$
    for $i = 1, 2, \ldots, N$ **do** // Construct cluster-to-data lookup $\vec{v}$.
      if $\psi_i = j$ **then**
        $v_l \leftarrow i$; $l \leftarrow l + 1$
      end if
    end for
    $J_{\text{med}} \leftarrow \text{median}(J(v,:))$ // Column-wise median over m cluster members.
    $d \leftarrow \vec{0} \in \mathbb{R}^m$
    for $l = 1, 2, \ldots, m$ **do**
      $d_l \leftarrow \|J(v_l,:) - J_{\text{med}}\|_{\ell_2}^2$
    end for
    $C_j \leftarrow v_s : d_s = \min(d)$ // Cluster medoid is closest member to J_{med}.
  end for
  if $C_{\text{old}} = C$ **then** // Converged if assignments are unchanged.
    conv $\leftarrow$ `true`
  end if
end while
Output: Indices of cluster medoids, $C \in \mathbb{N}^k$, and cluster assignments, $\psi \in \mathbb{N}^N$.

Bracketing algorithm: Computing binary weights

- Two phases: a **"ranking" phase** to determine the best **candidate** cells for lossless compression, and a **"bracketing" phase** to determine **losslessly compressible** cells.
- In the ranking phase, where we use the relaxed objective function L_r, we compute cell-wise weights $\alpha \in \mathbb{R}^N$ based on the normalized squares of cluster sizes. This weighting scheme prioritizes large clusters while sacrificing smaller clusters.
- In the bracketing phase, we employ a **bisection-like procedure** based on the percentage of the cells for which we conjecture near-zero misfit.
- We compute the weights $\alpha \in \{0, 1\}^N$ by sorting the cell misfits $\|K_i(\mathbf{p}) - \mu_i\|_2^2$, $i = 1, 2, \ldots N$, in ascending order.
- We begin with a guess of 50%, and if the obtained objective function is small, after several rounds of clustering, we increase the percentage to 75%; if it is large, we decrease it to 25%.
- The process is then repeated.
- We zoom in, within 1% accuracy, on the number of losslessly compressible cells in only a handful of iterations, due to the scheme's **logarithmic complexity**.

Bracketing algorithm: Computing binary weights

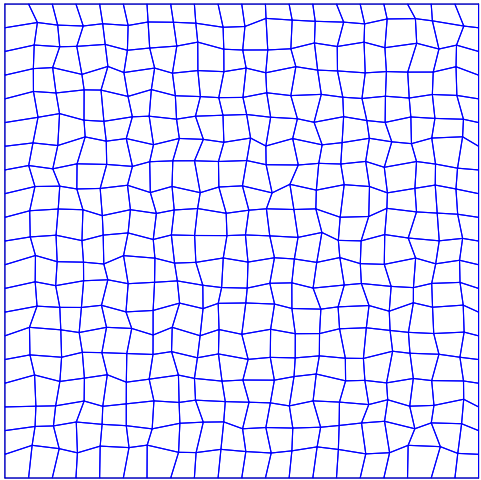
Algorithm 3. r-adaptive optimization with bracketing.

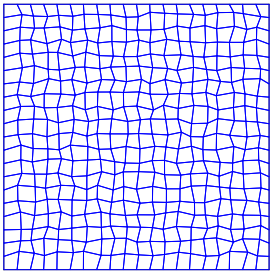
Input: Mesh $\mathcal{T}_h$ with points $\hat{\mathbf{p}}$; compression tolerance ε; ALESQP tolerance tol.
Initialization: Set ALESQP parameters as in [6, p. 258] and set ALESQP stopping tolerance to tol. Set bracket $(b_{\text{bot}}, b_{\text{mid}}, b_{\text{top}}) \leftarrow (0, 50, 100)$. Set $b_{\text{pick}} \leftarrow b_{\text{mid}}$. Choose numbers of clusters, k, ranking iterations, n_{rank}, bracketing iterations, n_{bracket}, and clustering iterations, n_{clust}. Set $\mathbf{p} \leftarrow \hat{\mathbf{p}}$, $L_{\text{prev}} \leftarrow \infty$ and $\tau \leftarrow 10^{-8}$.
for $i = 1, 2, \ldots, n_{\text{rank}}$ **do** // Ranking phase to find candidates.
    Compute targets $\mu \in \mathbb{R}^{N\times 8}$ using Algorithm 5.1.
    Compute weights $\alpha \in \mathbb{R}^N$ using formula (5.4).
    Solve problem (5.1) with the relaxed objective function L_r using ALESQP.
    if $L_r(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$ **then** // Objective tolerance met.
        break
    end if
    if $|L_r(\mathbf{p}) - L_{\text{prev}}|/|L_{\text{prev}}| \leq 10^{-3}$ **then** // Stagnation detected.
        break
    end if
    Set $L_{\text{prev}} \leftarrow L_r(\mathbf{p})$.
end for
if $L_r(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$ **then** // Ranking phase sufficient.
    Set $b_{\text{pick}} \leftarrow 99$ and skip bracketing phase.
end if
Set $\mathbf{p}_{\text{init}} = \mathbf{p}$, $L_{\text{rank}} = L_r(\mathbf{p})$.
for $i = 1, 2, \ldots, n_{\text{bracket}}$ **do** // Bracketing phase.
    Set $\mathbf{p} \leftarrow \mathbf{p}_{\text{init}}$, $L_{\text{prev}} = L_{\text{rank}}$.
    for $i = 1, 2, \ldots, n_{\text{clust}}$ **do** // Clustering.
        Compute targets $\mu \in \mathbb{R}^{N\times 8}$ using Algorithm 5.1.
        Compute weights $\alpha \in \{0, 1\}^N$ using formula (5.5) with $\eta = \lfloor b_{\text{mid}}/100 \cdot N \rfloor$.
        Solve problem (5.1) using ALESQP.
        if $L(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$ **then** // Objective tolerance met.
            break
        end if
        if $|L(\mathbf{p}) - L_{\text{prev}}|/|L_{\text{prev}}| \leq 10^{-3}$ **then** // Stagnation detected.
            break
        end if
        Set $L_{\text{prev}} \leftarrow L(\mathbf{p})$.
    end for
    if $L(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$ **then** // Bracketing procedure.
        if $b_{\text{mid}} > b_{\text{pick}}$ **then** $b_{\text{pick}} \leftarrow b_{\text{mid}}$
        end if
        Set $b_{\text{bot}} \leftarrow b_{\text{mid}}$ and $b_{\text{mid}} \leftarrow \lfloor (b_{\text{mid}} + b_{\text{top}})/2 \rfloor$.
    else
        Set $b_{\text{top}} \leftarrow b_{\text{mid}}$ and $b_{\text{mid}} \leftarrow \lfloor (b_{\text{mid}} + b_{\text{bot}})/2 \rfloor$.
    end if
end for
Repeat clustering with $\eta \leftarrow \lfloor b_{\text{pick}}/100 \cdot N \rfloor$ and $\tau \leftarrow \varepsilon^2/N$. // Refine result.
Output: Mesh $\mathcal{T}_{h,r}$ with points $\mathbf{p}$.

Numerical results: Structured mesh recovery

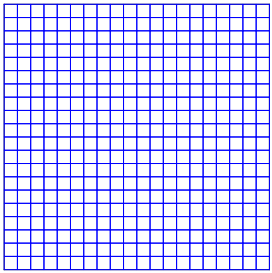
Example 1: Structured mesh recovery.

- T_h is a uniform mesh of square cells on the domain $[-0.5, 0.5]^2$ with a random perturbation of strength $0.4\sqrt{N}$ applied to each non-boundary vertex.
- The volume constraints are formulated using the global bounds, with $\gamma = 0.4$.
- We set the number of clusters to $k = 2$.
- Algorithm parameters are $\varepsilon = 10^{-10}$, $\text{tol} = 10^{-12}$, $n_{\text{rank}} = 6$, $n_{\text{bracket}} = 8$, and $n_{\text{clust}} = 200$. The ALESQP parameters, other than tol, come directly from Antil, Kouri, and Ridzal 2023, p. 258.

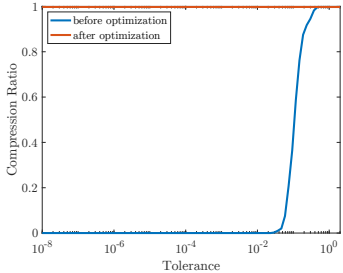




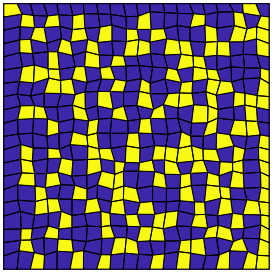
(a) Mesh before.



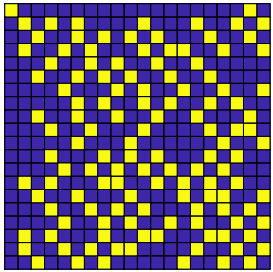
(b) Mesh after.



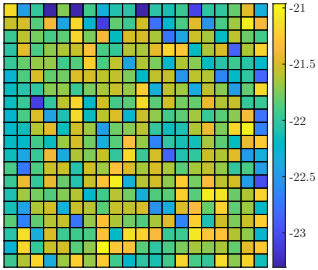
(c) Mesh compression.



(d) First cluster assignment.



(e) Last cluster assignment.



(f) $\log_{10}$ misfit with respect to the cluster medoids.

Numerical results: Scalability of structured mesh recovery

Mesh	CLS	AL	SQP	CG	Avg. GMRES
20×20	7	11	12	223	4.51
40×40	38	71	52	944	3.91
80×80	29	52	39	726	3.03
160×160	28	49	38	851	2.87
320×320	30	54	43	1522	2.91

Iteration counts for the randomly perturbed square mesh with increasing grid sizes.

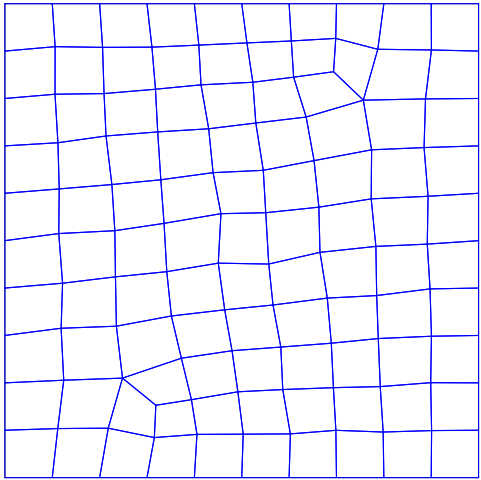
- CLS denotes the total number of clustering stages,
- AL denotes the total number of (outer) augmented Lagrangian iterations in ALESQP,
- SQP denotes the total number of (inner) SQP iterations,
- CG denotes the total number of projected CG iterations in the tangential-step QP subproblem, and
- Avg. GMRES denotes the average number of GMRES iterations per call to GMRES.

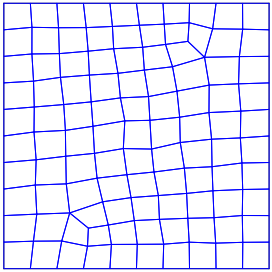
With the exception of the 20×20 mesh, all iteration numbers remain in narrow ranges with increasing mesh size. Note the low average numbers of GMRES iterations, around three.

Numerical results: Square mesh with poor topology

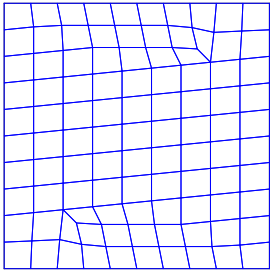
Example 2: Square mesh with poor topology.

- 10×10 boundary division and 99 quadrilaterals in the interior.
- This mesh is taken as a two-dimensional slice from the front corner of the three-dimensional extruded mesh, previously shown.
- We use two clusters, $k = 2$.
- We formulate the constraints using the local bounds, with $\gamma = 0.4$.

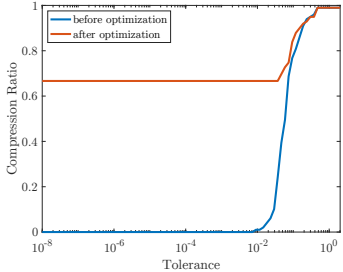




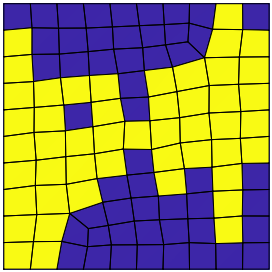
(a) Mesh before.



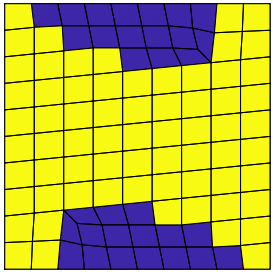
(b) Mesh after.



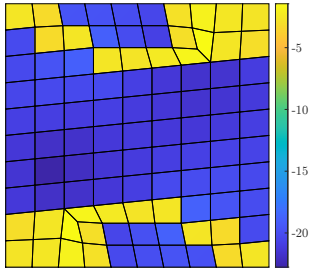
(c) Mesh compression.



(d) First cluster assignment.



(e) Last cluster assignment.

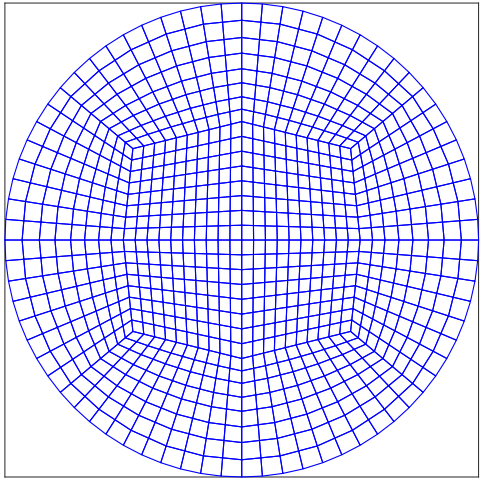


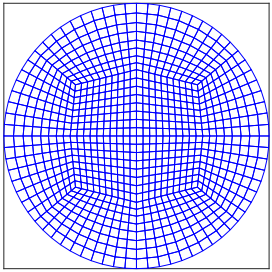
(f) $\log_{10}$ misfit with respect to the cluster medoids.

Numerical results: Circle mesh

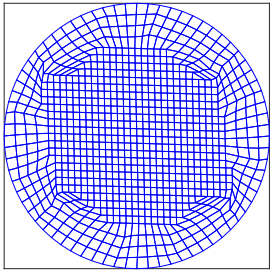
Example 3: Circle mesh.

- The mesh contains 896 cells and 933 vertices, of which 861 nodes are free.
- Here we use four clusters, $k = 4$.
- We formulate the constraints using the local bounds, with $\gamma = 0.4$.

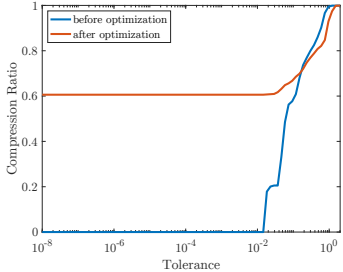




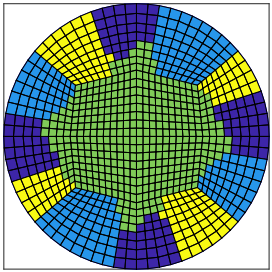
(a) Mesh before.



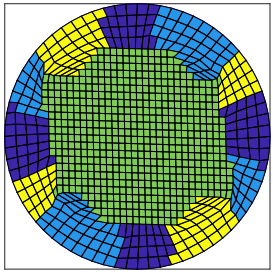
(b) Mesh after.



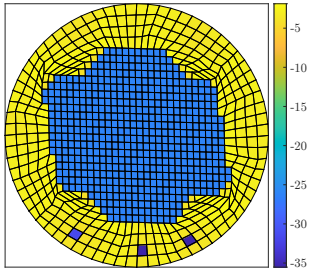
(c) Mesh compression.



(d) First cluster assignment.



(e) Last cluster assignment.



(f) $\log_{10}$ misfit with respect to the cluster medoids.

Conclusions

- Demonstrated two complementary approaches to reducing the memory burden of large-scale finite element simulations.
- First contribution is a dictionary-based compression scheme, with massive reductions in memory use in the MrHyDE finite element simulator.
- Second contribution is an r-adaptive mesh optimization algorithm that combines the recently developed ALESQP method, *k*-medoids clustering, and bracketing, to enhance redundancy in meshes that do not naturally possess it.
- Demonstrated significant lossless compression for a variety of challenging meshes, while maintaining cell shape quality through volume inequality constraints.
- Future research directions include extensions to other types of constraints, such as convexity constraints, and algorithmic improvements.
- In the large-scale setting, with meshes containing millions or billions of finite elements, the developed algorithms are meant to be applied to subdomains resulting from typical spatial decomposition, rather than the full mesh.

References I

Ainsworth, M., G. Andriamaro, and O. Davydov (2011). “Bernstein–Bézier finite elements of arbitrary order and optimal assembly procedures”. In: *SIAM Journal on Scientific Computing* 33.6, pp. 3087–3109. DOI: 10.1137/11082539X.

Anderson, R. et al. (2021). “MFEM: A modular finite element methods library”. In: *Computers & Mathematics with Applications* 81, pp. 42–74. DOI: 10.1016/j.camwa.2020.06.009.

Antil, H., D. P. Kouri, and D. Ridzal (2023). “ALESQP: An augmented Lagrangian equality-constrained SQP method for optimization with general constraints”. In: *SIAM Journal on Optimization* 33.1, pp. 237–266. DOI: 10.1137/20M1378399.

Bach, F. et al. (2012). “Structured sparsity through convex optimization”. In: *Statistical Science* 27.4, pp. 450–468. DOI: 10.1214/12-STS394.

Bangerth, W., R. Hartmann, and G. Kanschat (2007). “deal.II–a general-purpose object-oriented finite element library”. In: *ACM Transactions on Mathematical Software (TOMS)* 33.4, 24–es. DOI: 10.1145/1268776.1268779.

Bochev, P. et al. (2012). “Solving PDEs with Intrepid”. In: *Scientific Programming* 20.2, pp. 151–180. DOI: 10.3233/SPR-2012-0340.

Boffi, D., F. Brezzi, M. Fortin, et al. (2013). *Mixed finite element methods and applications*. Vol. 44. Springer. DOI: 10.1007/978-3-642-36519-5.

References II

- Bressan, A. and S. Takacs (2019). “Sum factorization techniques in isogeometric analysis”. In: *Computer Methods in Applied Mechanics and Engineering* 352, pp. 437–460. DOI: 10.1016/j.cma.2019.04.031.
- Browne, P. A. et al. (2014). “Fast three dimensional r-adaptive mesh redistribution”. In: *Journal of Computational Physics* 275, pp. 174–196. DOI: 10.1016/j.jcp.2014.06.009.
- Clevenger, T. C. and T. Heister (2021). “Comparison between algebraic and matrix-free geometric multigrid for a Stokes problem on adaptive meshes with variable viscosity”. In: *Numerical Linear Algebra with Applications* 28.5, e2375. DOI: 10.1002/nla.2375.
- Cossette, J.-F., P. K. Smolarkiewicz, and P. Charbonneau (2014). “The Monge–Ampère trajectory correction for semi-Lagrangian schemes”. In: *Journal of Computational Physics* 274, pp. 208–229. DOI: 10.1016/j.jcp.2014.05.016.
- D’Elia, M. et al. (2016). “Optimization-based mesh correction with volume and convexity constraints”. In: *Journal of Computational Physics* 313, pp. 455–477. DOI: 10.1016/j.jcp.2016.02.050.
- Griewank, A. (1992). “Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation”. In: *Optimization Methods and Software* 1.1, pp. 35–54. DOI: 10.1080/10556789208805505.

References III

- Heinkenschloss, M. and D. Ridzal (2014). “A matrix-free trust-region SQP method for equality-constrained optimization”. In: *SIAM Journal on Optimization* 24.3, pp. 1507–1541. DOI: 10.1137/130921738.
- Knoll, D. A. and D. E. Keyes (2004). “Jacobian-free Newton–Krylov methods: a survey of approaches and applications”. In: *Journal of Computational Physics* 193.2, pp. 357–397. DOI: 10.1016/j.jcp.2003.08.010.
- May, D. A., J. Brown, and L. Le Pourhiet (2015). “A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow”. In: *Computer methods in applied mechanics and engineering* 290, pp. 496–523. DOI: 10.1016/j.cma.2015.03.014.
- Mora, J. and L. Demkowicz (2019). “Fast integration of DPG matrices based on sum factorization for all the energy spaces”. In: *Computational Methods in Applied Mathematics* 19.3, pp. 523–555. DOI: 10.1515/cmam-2018-0205.
- *MrHyDE library* (n.d.). Accessed: 09/25/2024. URL: https://github.com/sandialabs/MrHyDE.
- Park, H.-S. and C.-H. Jun (2009). “A simple and fast algorithm for K-medoids clustering”. In: *Expert systems with applications* 36.2, pp. 3336–3341. DOI: 10.1016/j.eswa.2008.01.039.
- Suárez, J. P., A. Trujillo, and T. Moreno (2021). “Computing the exact number of similarity classes in the longest edge bisection of tetrahedra”. In: *Mathematics* 9.12, p. 1447. DOI: 10.3390/math9121447.

References IV

Top500.org (2024). https://www.top500.org/lists/top500/list/2024/06. (Visited on 06/30/2024).

Trujillo-Pino, A., J. P. Suárez, and M. A. Padrón (2024). "Finite number of similarity classes in Longest Edge Bisection of nearly equilateral tetrahedra". In: *Applied Mathematics and Computation* 472, p. 128631. DOI: 10.1016/j.amc.2024.128631.

Thank you for listening

G. Harper, D. Ridzal and T. Wildey,
R-Adaptive Mesh Optimization to Enhance Finite Element Basis Compression,
arxiv:2410.07646