



Sandia
National
Laboratories

SAND2024-14350C

Finite Element Basis Compression for Extreme-Scale Simulations and R-Adaptive Mesh Optimization to Enable Compression

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

University of New Mexico Applied Mathematics Seminar

October 21, 2024



LDRD

Laboratory Directed Research
and Development



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.

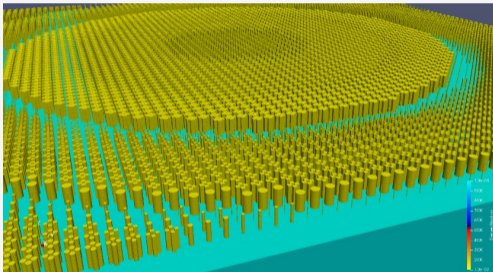
SAND2024-14350C

2 History, circa 2021: Design of electromagnetic systems



Optimal design

GOAL Electromagnetic devices with nanoscale structures.

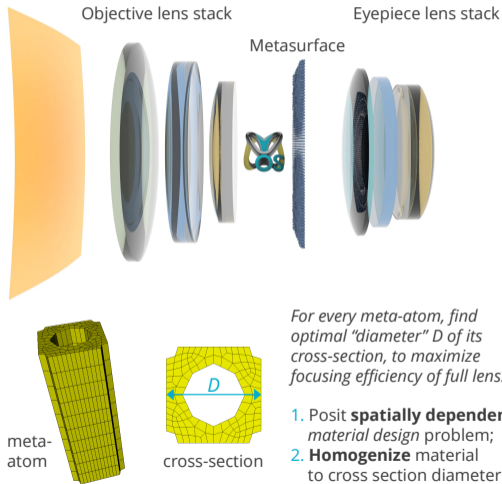


EXAMPLE A micron-thin lens with millions of *meta-atoms*.

PURPOSE Weight and space savings.

APPLICATION Next-generation night vision systems, etc.

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



3 The memory challenge



- Top supercomputers **Frontier** at Oak Ridge National Lab and **Aurora** at Argonne National Lab are demonstrating **exascale** (10^{18}) calculations^[1]...

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

3 | The memory challenge



- Top supercomputers **Frontier** at Oak Ridge National Lab and **Aurora** at Argonne National Lab are demonstrating **exascale** (10^{18}) calculations^[1]...
- ...but taking advantage of heterogeneous architectures effectively is a challenge.

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

3 | The memory challenge



- Top supercomputers **Frontier** at Oak Ridge National Lab and **Aurora** at Argonne National Lab are demonstrating **exascale** (10^{18}) calculations^[1]...
- ...but taking advantage of heterogeneous architectures effectively is a challenge.
- Solving partial differential equations (PDEs) using finite element methods (FEMs) is **limited** in prediction fidelity by **system memory** rather than processing power.

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

3 The memory challenge



- Top supercomputers **Frontier** at Oak Ridge National Lab and **Aurora** at Argonne National Lab are demonstrating **exascale** (10^{18}) calculations^[1]...
- ...but taking advantage of heterogeneous architectures effectively is a challenge.
- Solving partial differential equations (PDEs) using finite element methods (FEMs) is **limited** in prediction fidelity by **system memory** rather than processing power.
- While machine learning and dimension reduction are helpful, they often do not reduce the underlying memory constraints.

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

4 Related work



- Matrix-free approaches, such as **Jacobian-free Newton Krylov** methods^[2] 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^{[3][4]}.

[2] 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](https://doi.org/10.1016/j.jcp.2003.08.010).

[3] 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](https://doi.org/10.1016/j.cma.2015.03.014).

[4] 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](https://doi.org/10.1002/nla.2375).



- Other data reduction techniques such as **sum factorizations**^{[5][6][7]} 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**^[8].
- Our goal: **improve mesh-based scaling terms** which have not previously been studied to **enable FEMs to run at a similar memory profile to FDMs**

[5] 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](https://doi.org/10.1137/11082539X).

[6] 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](https://doi.org/10.1016/j.cma.2019.04.031).

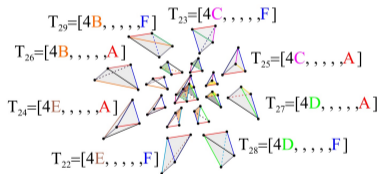
[7] J. Mora and L. Demkowicz (2019). “Fast integration of DPG matrices based on sum factorization for all the energy spaces”. In: *Computational Methods in Applied Mathematics* 19.3, pp. 523–555. DOI: [10.1515/cmam-2018-0205](https://doi.org/10.1515/cmam-2018-0205).

[8] 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](https://doi.org/10.1080/10556789208805505).

6 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**^{[9][10]} ...



...important related ideas in mesh generation and refinement:
Finite number of similarity classes, ≤ 37 , in longest edge bisection of tetrahedra.

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

[9] 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](https://doi.org/10.3390/math9121447).

[10] 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](https://doi.org/10.1016/j.amc.2024.128631).

7 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 dictionary achieves reductions in storage of more than 99% for meshes with redundant structure, enabling **billion-element simulations** in under 100 gigabytes of memory.

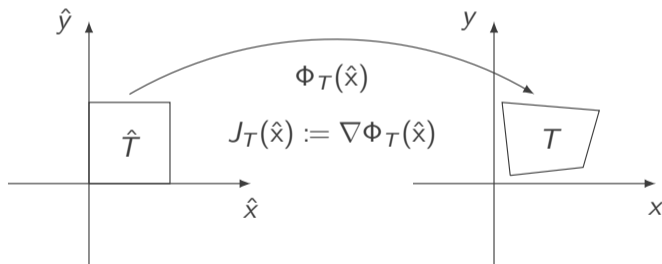
8 FEM background



- An important computational kernel in FEM:

$$\int_T k(x) D^{\alpha_1} \phi_i(x) D^{\alpha_2} \phi_j(x) dx. \quad (1)$$

- 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** $\Phi_T : \hat{T} \rightarrow T$, where x refers to physical coordinates on T and \hat{x} refers to reference coordinates on \hat{T} .

9 FEM background



- From calculus, integrals of scalar-valued functions are pulled back by the **change of coordinates formula**

$$\int_T f(x) dx = \int_{\hat{T}} \hat{f}(\hat{x}) \det(J_T(\hat{x})) d\hat{x}, \quad (2)$$

where $\hat{f}(\hat{x}) := f(\Phi(\hat{x}))$ is the **pullback** of $f(x)$.

9 FEM background



- From calculus, integrals of scalar-valued functions are pulled back by the **change of coordinates formula**

$$\int_T f(\mathbf{x}) \, d\mathbf{x} = \int_{\hat{T}} \hat{f}(\hat{\mathbf{x}}) \det(J_T(\hat{\mathbf{x}})) \, d\hat{\mathbf{x}}, \quad (2)$$

where $\hat{f}(\hat{\mathbf{x}}) := f(\Phi(\hat{\mathbf{x}}))$ is the **pullback** of $f(\mathbf{x})$.

- From differential geometry, integrals of vector fields (1-forms) are pulled back by the **covariant transformation**,

$$\int_T \nabla f(\mathbf{x}) \, d\mathbf{x} = \int_{\hat{T}} J_T^{-T}(\hat{\mathbf{x}}) \nabla \hat{f}(\hat{\mathbf{x}}) \det(J_T(\hat{\mathbf{x}})) \, d\hat{\mathbf{x}}. \quad (3)$$

9 FEM background



- From calculus, integrals of scalar-valued functions are pulled back by the **change of coordinates formula**

$$\int_T f(x) dx = \int_{\hat{T}} \hat{f}(\hat{x}) \det(J_T(\hat{x})) d\hat{x}, \quad (2)$$

where $\hat{f}(\hat{x}) := f(\Phi(\hat{x}))$ is the **pullback** of $f(x)$.

- From differential geometry, integrals of vector fields (1-forms) are pulled back by the **covariant transformation**,

$$\int_T \nabla f(x) dx = \int_{\hat{T}} J_T^{-T}(\hat{x}) \nabla \hat{f}(\hat{x}) \det(J_T(\hat{x})) d\hat{x}. \quad (3)$$

- The pullback changes depending on the discretization space within the **de Rham complex**.

space:	$HGRAD$	$\xrightarrow{\nabla}$	$HCURL$	$\xrightarrow{\nabla \times}$	$HDIV$	$\xrightarrow{\nabla \cdot}$	$HVOL$
	$H^1(\Omega)$		$H(curl, \Omega)$		$H(div, \Omega)$		$L^2(\Omega)$
pullback:	I		J^{-T}		$J \det(J)^{-1}$		$\det(J)^{-1}$



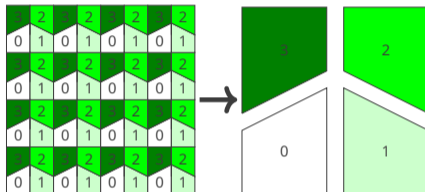
- Two mesh cells T_1 and T_2 with equal Jacobians $J_{T_1} = J_{T_2}$ must have mappings Φ_{T_1} and Φ_{T_2} which differ by only a constant.
- i.e., T_1 is a **translation** of T_2 if and only if $J_{T_1} = J_{T_2}$, i.e., T_1 and T_2 have identical *shapes*.

Lemma

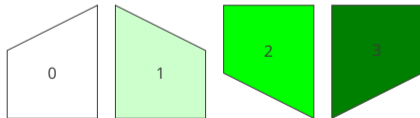
Let $\hat{\phi} \in \mathcal{S}$, with $\mathcal{S} \in \{H^1(\hat{T}), H(\text{curl}; \hat{T}), H(\text{div}; \hat{T}), L^2(\hat{T})\}$, be a finite element basis function on the reference domain. Let $D \in \{1, \text{grad}, \text{curl}, \text{div}\}$ be an operator so that $D\hat{\phi} \in L^2(\hat{T})$. 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,

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

- Identify the “unique” cells – preprocessing but may be done offline.



- Perform basis evaluations on only the unique cells.



- Reuse evaluations by the dictionary lookup.

FEM background



- Compared to methods such as partial basis storage, compressing basis data **may reduce storage costs and simulation time**.

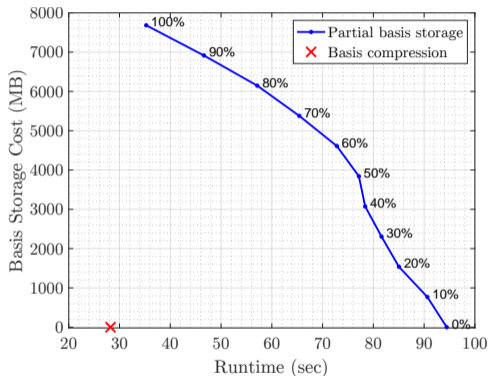


Figure: Tradeoffs between storing and recomputing basis data for a heat transfer simulation, where we show the percentage of basis data retained across the mesh. For the given mesh, containing significant redundancy, the cost of our proposed basis compression scheme is denoted by the **x** mark.

Mesh examples with redundancy

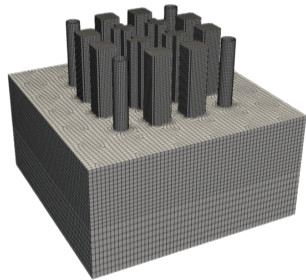
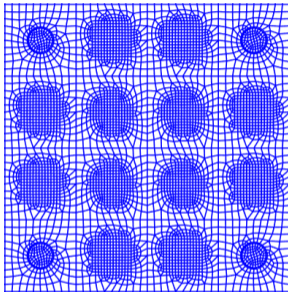
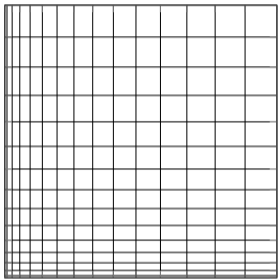


Figure: Three examples of mesh classes with redundancy: (left) structured/graded, (middle) geometrically patterned, and (right) extruded. The latter two meshes comprise large clusters of identical cell shapes. The cells in the graded mesh have shapes that are identical up to a dimensional scaling.



- Identify “unique” mesh cells by distance $d(T_i, T_j)$ using cell Jacobians,

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

- If $d(T_i, T_j) < \varepsilon$, match 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**.



Algorithm 1. Construction of the mesh dictionary.

- 1: **Input:** Mesh $\mathcal{T}_h = \{T_1, T_2, \dots, T_N\}$, tolerance ε .
 - 2: $\mathcal{B}_\varepsilon \leftarrow \{\}$, $\psi_\varepsilon \leftarrow \vec{0} \in \mathbb{N}^N$
 - 3: **for** $i = 1, 2, \dots, N$ **do**
 - 4: match \leftarrow false
 - 5: **for** $T_j \in \mathcal{B}_\varepsilon$ **do**
 - 6: **if** $d(T_i, T_j) < \varepsilon$ **then**
 - 7: match \leftarrow true; $\psi_\varepsilon(i) \leftarrow j$; **break**
 - 8: **end if**
 - 9: **end for**
 - 10: **if** match = false **then**
 - 11: append T_i to \mathcal{B}_ε ; $\psi_\varepsilon(i) \leftarrow |\mathcal{B}_\varepsilon|$
 - 12: **end if**
 - 13: **end for**
 - 14: **Output:** Dictionary of cells, \mathcal{B}_ε , 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,

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

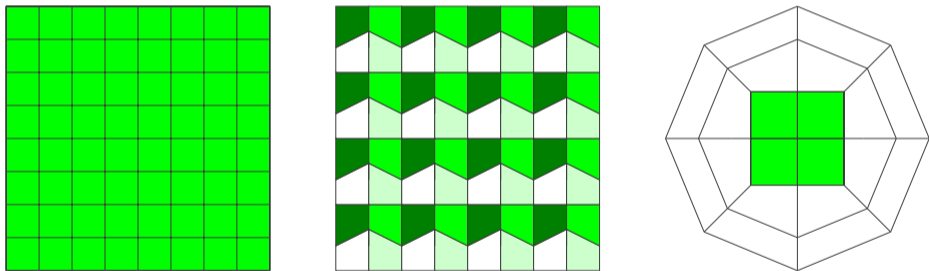


Figure: Illustration of (left) a structured square mesh with one unique shape and a lossless compression ratio ($\varepsilon = \varepsilon_0$) of 98.4% (middle) a trapezoidal mesh with only four unique shapes and a lossless compression ratio of 93.75%, and (right) a circle mesh with a lossless compression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

17 Compression curve



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

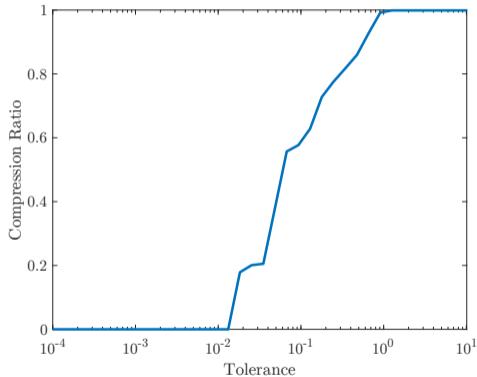
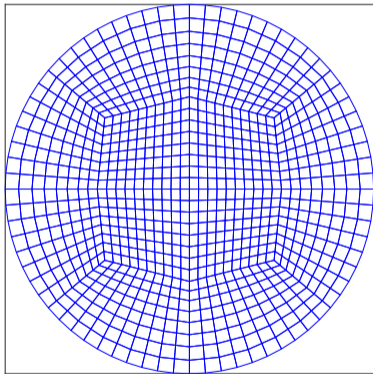
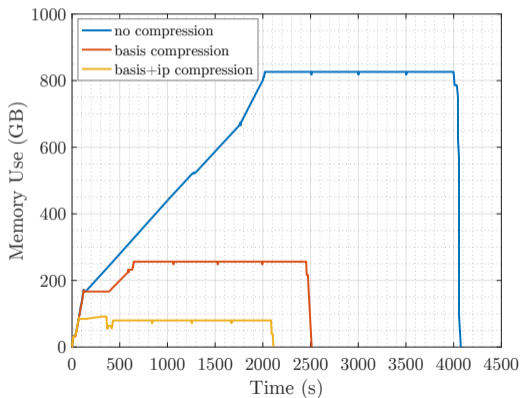


Figure: Compression curve for a circle mesh; no compression is observed for $\varepsilon < 10^{-2}$.

Memory savings for a heat transfer example



- Memory profiles and timing data for a heat transfer simulation in **MrHyDE**^[11] 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**.



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^[12] with a sparsity-promoting optimization formulation.
- The latter is inspired by the **constrained optimization alternative**^[13] to r-adaptive schemes based on solving the **Monge-Ampère equation**^{[14][15]}.
- 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**^[16].

[12] 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](https://doi.org/10.1016/j.eswa.2008.01.039).

[13] 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](https://doi.org/10.1016/j.jcp.2016.02.050).

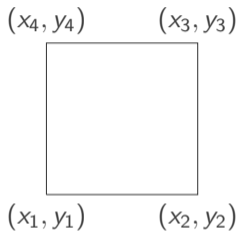
[14] J.-F. Cossette, P. K. Smolarkiewicz, and P. Charbonneau (2014). “The Monge–Ampère trajectory correction for semi-Lagrangian schemes”. In: *Journal of Computational Physics* 274, pp. 208–229. DOI: [10.1016/j.jcp.2014.05.016](https://doi.org/10.1016/j.jcp.2014.05.016).

[15] 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](https://doi.org/10.1016/j.jcp.2014.06.009).

[16] 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](https://doi.org/10.1137/20M1378399).



- Focus on the **quadrilateral** case with straight line geometry.
- Mesh $\mathcal{T}_h = \{T_1, T_2, \dots, T_N\}$ of a domain $\Omega \subset \mathbb{R}^2$, with n_p vertices $\hat{p} = \{\hat{p}_1, \hat{p}_2, \dots, \hat{p}_{n_p}\}$.
- Dictionary \mathcal{B}_ε , 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}_ε .
- Requirement: Maintain topology, with relatively small mesh motion.
- Optimization variables: Vertices $p = \{p_1, p_2, \dots, p_{n_p}\}$, where $p_i = (x_i, y_i)$, $i = 1, 2, \dots, n_p$.
- Let the double-indexing $p_{i,j} = (x_{i,j}, y_{i,j})$ denote vertex j of mesh cell i , $i = 1, 2, \dots, N$, $j = 1, 2, \dots, n_{cv}$, with n_{cv} denoting the number of vertices in each cell.

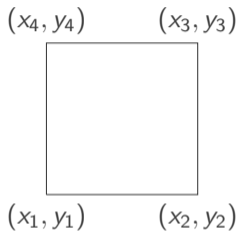


- The reference to physical mapping is given by

$$\Phi(\hat{x}, \hat{y}) = \begin{bmatrix} x_1 + (x_2 - x_1)\hat{x} + (x_4 - x_1)\hat{y} + (x_1 - x_2 + x_3 - x_4)\hat{x}\hat{y} \\ y_1 + (y_2 - y_1)\hat{x} + (y_4 - y_1)\hat{y} + (y_1 - y_2 + y_3 - y_4)\hat{x}\hat{y} \end{bmatrix}.$$

- Jacobian is given by

$$J(\hat{x}, \hat{y}) = \nabla \Phi(\hat{x}, \hat{y}) = \begin{bmatrix} (x_2 - x_1) + (x_1 - x_2 + x_3 - x_4)\hat{y} & (x_4 - x_1) + (x_1 - x_2 + x_3 - x_4)\hat{x} \\ (y_2 - y_1) + (y_1 - y_2 + y_3 - y_4)\hat{y} & (y_4 - y_1) + (y_1 - y_2 + y_3 - y_4)\hat{x} \end{bmatrix}.$$



- The reference to physical mapping is given by

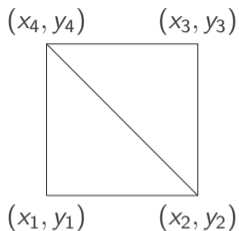
$$\Phi(\hat{x}, \hat{y}) = \begin{bmatrix} x_1 + (x_2 - x_1)\hat{x} + (x_4 - x_1)\hat{y} + (x_1 - x_2 + x_3 - x_4)\hat{x}\hat{y} \\ y_1 + (y_2 - y_1)\hat{x} + (y_4 - y_1)\hat{y} + (y_1 - y_2 + y_3 - y_4)\hat{x}\hat{y} \end{bmatrix}$$

- Jacobian is given by

$$J(\hat{x}, \hat{y}) = \nabla \Phi(\hat{x}, \hat{y}) = \begin{bmatrix} (x_2 - x_1) + (x_1 - x_2 + x_3 - x_4)\hat{y} & (x_4 - x_1) + (x_1 - x_2 + x_3 - x_4)\hat{x} \\ (y_2 - y_1) + (y_1 - y_2 + y_3 - y_4)\hat{y} & (y_4 - y_1) + (y_1 - y_2 + y_3 - y_4)\hat{x} \end{bmatrix}$$

- This quantity varies spatially, multiplying the amount of data by 4-8x

Problem formulation: Quantities of interest



- New plan: split into subtriangles (1, 2, 4) and (3, 4, 2).
- On a general mesh cell T_i , $1 \leq i \leq N$, define

$$c_{i,x2} = x_{i,2} - x_{i,1},$$

$$c_{i,x4} = x_{i,4} - x_{i,1},$$

$$c_{i,y2} = y_{i,2} - y_{i,1},$$

$$c_{i,y4} = y_{i,4} - y_{i,1},$$

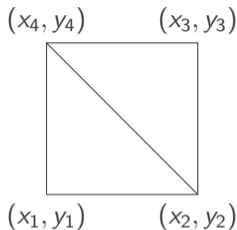
$$d_{i,x4} = x_{i,4} - x_{i,3},$$

$$d_{i,x2} = x_{i,2} - x_{i,3},$$

$$d_{i,y4} = y_{i,4} - y_{i,3},$$

$$d_{i,y2} = y_{i,2} - y_{i,3}.$$

- Two constant Jacobians – no longer spatially dependent.



- New plan: split into subtriangles (1, 2, 4) and (3, 4, 2).
- On a general mesh cell T_i , $1 \leq i \leq N$, define

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

- Two constant Jacobians – no longer spatially dependent.
- Row vector $K_i(\mathbf{p})$, $i = 1, 2, \dots, N$, 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 , $i = 1, 2, \dots, N$, 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 $K(\mathbf{p}), \mu \in \mathbb{R}^{N \times 8}$.



- Focus on lossless finite element basis compression.
- Goal: match rows of $K(\mathbf{p})$ and μ , **maximizing the number of zero rows** in $K(\mathbf{p}) - \mu$.
- Row matching with **group sparsity**, such as those using the mixed ℓ_1/ℓ_q norm, $q > 1$, have been suggested^[17], with **non-smooth quantities of interest** of the form

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

- However, at the cost of group sparsity, the objective $\sum_{i=1}^N \|K_i(\mathbf{p}) - \mu_i\|_2^2$, is smooth and easier to minimize with derivative-based methods.
- Problem: ℓ_2 optimization will not perfectly match groups at the optimum. That is, $K_i(\mathbf{p}) - \mu_i \neq 0$ at the optimum for all $i = 1, 2, \dots, N$.

Problem formulation: Objective function



- New idea:

identify a subset of cell indices, $\mathcal{Z} \subseteq \{1, 2, \dots, N\}$,
such that $K_i(\mathbf{p}) - \mu_i = 0$ 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 the “lossless” objective

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

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

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



- Maintain the quality of the **finite element approximation** given by the original mesh.
- Mesh cells should **not become tangled or inverted**.
- Restrict the **volume change** for each mesh cell. Optionally: maintain **cell convexity**.
- Do not move nodes defining important geometric features (e.g., boundaries).
- Let $v_i(\mathbf{p})$ be the volume of cell i , $i = 1, 2, \dots, N$, given by the shoelace formula

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

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

Problem formulation: Constraints



- Let $v_{\min} = \min_{i \in \{1, 2, \dots, N\}} v_i(\hat{p})$ and $v_{\max} = \max_{i \in \{1, 2, \dots, N\}} v_i(\hat{p})$.
- 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

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

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

$$\underset{p}{\text{minimize}} \quad L(p) \tag{4a}$$

$$\text{subject to} \quad v^{\text{lo}} \leq v(p) \leq v^{\text{up}}. \tag{4b}$$

Lemma

The feasible set defined in (4b) is nonempty for both definitions of the bounds, global and local.



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

- **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**, $s \in \mathbb{R}^N$, $s_i = v_i(p)$, $i = 1, 2, \dots, N$:

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

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

$$\underset{p, s}{\text{minimize}} \quad L(p) + \frac{1}{2r} \left\| r \left(\frac{\lambda}{r} + s - \max \left(\min \left(\frac{\lambda}{r} + s, v^{\text{lo}} \right), v^{\text{up}} \right) \right) \right\|_2^2 \quad (5a)$$

$$\text{subject to} \quad v(p) - s = 0, \quad (5b)$$

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 (5a) is differentiable, despite the non-smooth \min and \max functions.

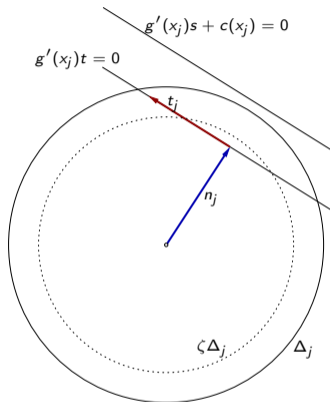


- **Trust-region step:** $s_j = n_j + t_j$
- **Quasi-normal step n_j :**
reduces linear infeasibility

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

- **Tangential step t_j :**
improves optimality while staying in the null space of the linearized constraints

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





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 \tilde{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:

$$\|e^1\| + \|e^2\| \leq \mathcal{F}(\|b^1\|, \|b^2\|, \|y^1\|, \Delta_j)$$

- In this application, with variables (p, s) , augmented systems are:

$$\begin{pmatrix} I & g'^T \\ g' & 0 \end{pmatrix} \rightarrow \begin{pmatrix} I & & g'_p{}^T \\ & I & g'_s{}^T \\ g'_p & g'_s & 0 \end{pmatrix}.$$

The key computational kernel: Augmented system



- Here g'_p corresponds to the Jacobian $v'_p(p)$ and g'_s is $-I$.
- As $v(p)$ is defined by the shoelace formula, the entries of g'_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'_p appears similar to a **discrete divergence** on the mesh.
- Similarly, $g'_p{}^T$ resembles a **discrete gradient**.
- So, the Schur complement of the augmented system matrix is

$$S = g'_p g'_p{}^T + I,$$

where the $g'_p g'_p{}^T$ 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 $g'_p g'_p{}^T$ may dominate, multigrid methods are very effective in approximating the application of S^{-1} .



- 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.
- Initialized via random permutation σ_k^N .
- Shape targets μ_i are defined by the corresponding medoid for each data point, $\mu_i = K_{C_{\psi_i}}(\mathbf{p})$, $i = 1, 2, \dots, N$.

Algorithm 2. Shape clustering.

```

1: Input: Jacobians  $J \in \mathbb{R}^{N \times 8}$ , number of clusters  $k$ .
2: Initialization:  $\psi \leftarrow \bar{0} \in \mathbb{N}^N$ ,  $\text{conv} \leftarrow \text{false}$ , draw permutation  $C \leftarrow \sigma_k^N \in \mathbb{N}^k$ .
3: while not conv do
4:   for  $i = 1, 2, \dots, N$  do // Assign  $\psi_i$  to closest cluster index based on  $\|\cdot\|_{\ell_2}^2$ .
5:      $d \leftarrow \bar{0} \in \mathbb{R}^k$ 
6:     for  $j = 1, 2, \dots, k$  do
7:        $d_j \leftarrow \|J(i, :) - J(C_j, :)\|_{\ell_2}^2$ 
8:     end for
9:      $\psi_i \leftarrow s : d_s = \min(d)$ 
10:  end for
11:   $C_{\text{old}} \leftarrow C$ 
12:  for  $j = 1, 2, \dots, k$  do // Assign  $C_j$  to global index for cluster medoid.
13:     $\mathcal{I} = \{l : \psi_l = j\}$ ;  $m = |\mathcal{I}|$ 
14:     $v = \bar{0} \in \mathbb{N}^m$ ;  $l = 1$ 
15:    for  $i = 1, 2, \dots, N$  do // Construct cluster-to-data lookup  $\vec{v}$ .
16:      if  $\psi_i = j$  then
17:         $v_l \leftarrow i$ ;  $l \leftarrow l + 1$ 
18:      end if
19:    end for
20:     $J_{\text{med}} \leftarrow \text{median}(J(v, :))$  // Column-wise median over  $m$  cluster members.
21:     $d \leftarrow \bar{0} \in \mathbb{R}^m$ 
22:    for  $l = 1, 2, \dots, m$  do
23:       $d_l \leftarrow \|J(v_l, :) - J_{\text{med}}\|_{\ell_2}^2$ 
24:    end for
25:     $C_j \leftarrow v_s : d_s = \min(d)$  // Cluster medoid is closest member to  $J_{\text{med}}$ .
26:  end for
27:  if  $C_{\text{old}} = C$  then // Converged if assignments are unchanged.
28:    conv  $\leftarrow$  true
29:  end if
30: end while
31: 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 $\|K_i(\mathbf{p}) - \mu_i\|_2^2$, $i = 1, 2, \dots, N$, 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**.



Algorithm 3. r-adaptive optimization with bracketing.

```

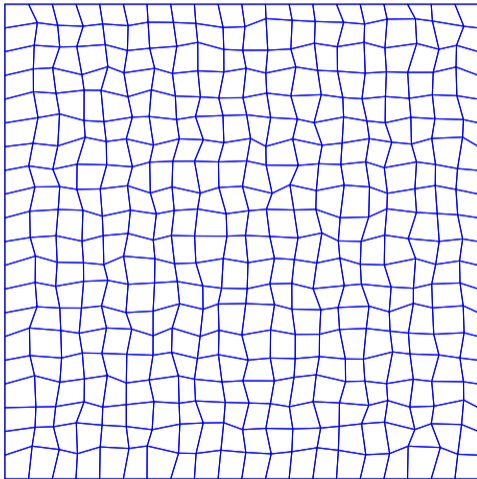
1: Input: Mesh  $\mathcal{T}_h$  with points  $\hat{\mathbf{p}}$ ; compression tolerance  $\varepsilon$ ; ALESQP tolerance tol.
2: Initialization: Set ALESQP parameters as in [6, p. 258] and set ALESQP stopping tolerance to tol. Set bracket  $(b_{\text{bot}}, b_{\text{mid}}, b_{\text{top}}) \leftarrow (0, 50, 100)$ . Set  $b_{\text{pick}} \leftarrow b_{\text{mid}}$ . Choose numbers of clusters,  $k$ , ranking iterations,  $n_{\text{rank}}$ , bracketing iterations,  $n_{\text{bracket}}$ , and clustering iterations,  $n_{\text{clust}}$ . Set  $\mathbf{p} \leftarrow \hat{\mathbf{p}}$ ,  $L_{\text{prev}} \leftarrow \infty$  and  $\tau \leftarrow 10^{-8}$ .
3: for  $i = 1, 2, \dots, n_{\text{rank}}$  do // Ranking phase to find candidates.
4:   Compute targets  $\mu \in \mathbb{R}^{N \times 8}$  using Algorithm 5.1.
5:   Compute weights  $\alpha \in \mathbb{R}^N$  using formula (5.4).
6:   Solve problem (5.1) with the relaxed objective function  $L_r$  using ALESQP.
7:   if  $L_r(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$  then // Objective tolerance met.
8:     break
9:   end if
10:  if  $|L_r(\mathbf{p}) - L_{\text{prev}}|/|L_{\text{prev}}| \leq 10^{-3}$  then // Stagnation detected.
11:    break
12:  end if
13:  Set  $L_{\text{prev}} \leftarrow L_r(\mathbf{p})$ .
14: end for
15: if  $L_r(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$  then // Ranking phase sufficient.
16:   Set  $b_{\text{pick}} \leftarrow 99$  and skip bracketing phase.
17: end if
18: Set  $\mathbf{p}_{\text{init}} = \mathbf{p}$ ,  $L_{\text{rank}} = L_r(\mathbf{p})$ .

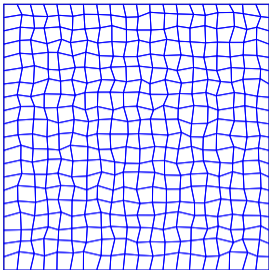
19: for  $i = 1, 2, \dots, n_{\text{bracket}}$  do // Bracketing phase.
20:   Set  $\mathbf{p} \leftarrow \mathbf{p}_{\text{init}}$ ,  $L_{\text{prev}} = L_{\text{rank}}$ .
21:   for  $i = 1, 2, \dots, n_{\text{clust}}$  do // Clustering.
22:     Compute targets  $\mu \in \mathbb{R}^{N \times 8}$  using Algorithm 5.1.
23:     Compute weights  $\alpha \in \{0, 1\}^N$  using formula (5.5) with  $\eta = \lfloor b_{\text{mid}}/100 \cdot N \rfloor$ .
24:     Solve problem (5.1) using ALESQP.
25:     if  $L(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$  then // Objective tolerance met.
26:       break
27:     end if
28:     if  $|L(\mathbf{p}) - L_{\text{prev}}|/|L_{\text{prev}}| \leq 10^{-3}$  then // Stagnation detected.
29:       break
30:     end if
31:     Set  $L_{\text{prev}} \leftarrow L(\mathbf{p})$ .
32:   end for
33:   if  $L(\mathbf{p}) \leq \max(\tau, \varepsilon^2/N)$  then // Bracketing procedure.
34:     if  $b_{\text{mid}} > b_{\text{pick}}$  then  $b_{\text{pick}} \leftarrow b_{\text{mid}}$ 
35:     end if
36:     Set  $b_{\text{bot}} \leftarrow b_{\text{mid}}$  and  $b_{\text{mid}} \leftarrow \lfloor (b_{\text{mid}} + b_{\text{top}})/2 \rfloor$ .
37:   else
38:     Set  $b_{\text{top}} \leftarrow b_{\text{mid}}$  and  $b_{\text{mid}} \leftarrow \lfloor (b_{\text{mid}} + b_{\text{bot}})/2 \rfloor$ .
39:   end if
40: end for
41: Repeat clustering with  $\eta \leftarrow \lfloor b_{\text{pick}}/100 \cdot N \rfloor$  and  $\tau \leftarrow \varepsilon^2/N$ . // Refine result.
42: Output: Mesh  $\mathcal{T}_{h,r}$  with points  $\mathbf{p}$ .

```

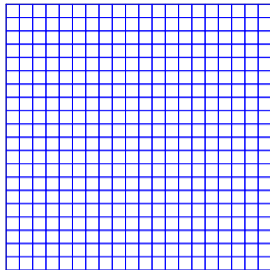
Example 1: Structured mesh recovery.

- 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.

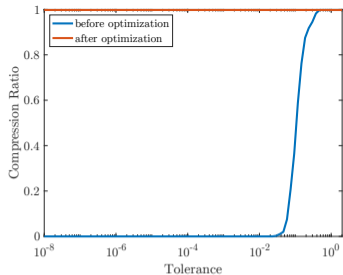




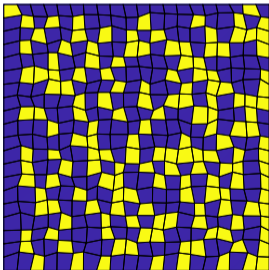
(a) Mesh before.



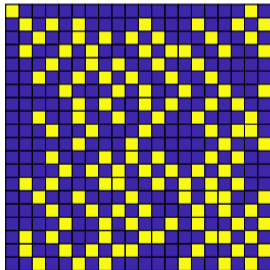
(b) Mesh after.



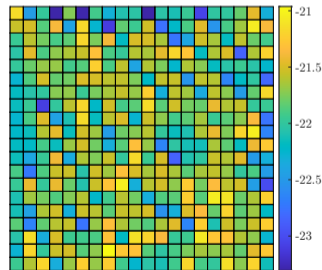
(c) Mesh compression.



(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

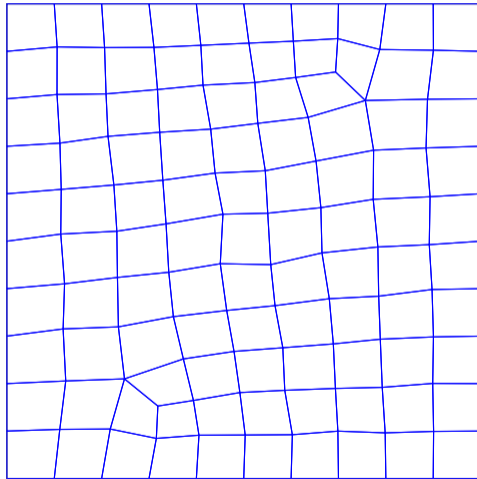
Iteration counts for the randomly perturbed square mesh with increasing grid sizes.

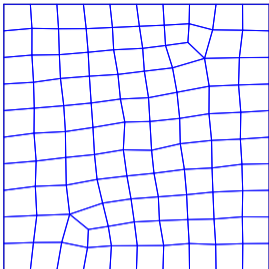
- CLS denotes the total number of clustering stages,
- AL denotes the total number of (outer) augmented Lagrangian iterations in ALESQP,
- SQP denotes the total number of (inner) SQP iterations,
- CG denotes the total number of projected CG iterations in the tangential-step QP subproblem, and
- Avg.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

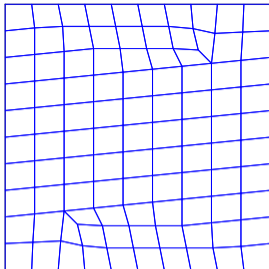
Example 2: Square mesh with poor topology.

- 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 $\gamma = 0.4$.

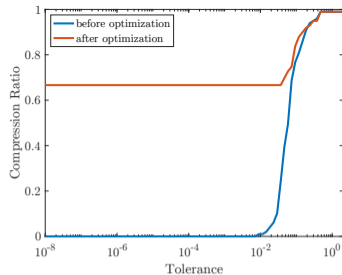




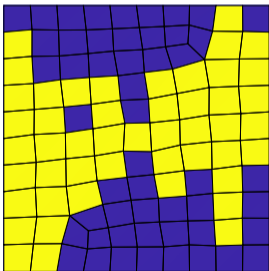
(a) Mesh before.



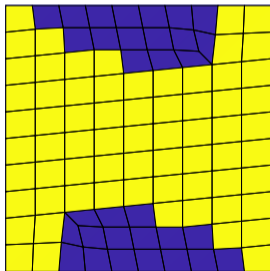
(b) Mesh after.



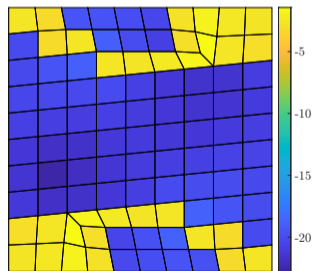
(c) Mesh compression.



(d) First cluster assignment.



(e) Last cluster assignment.

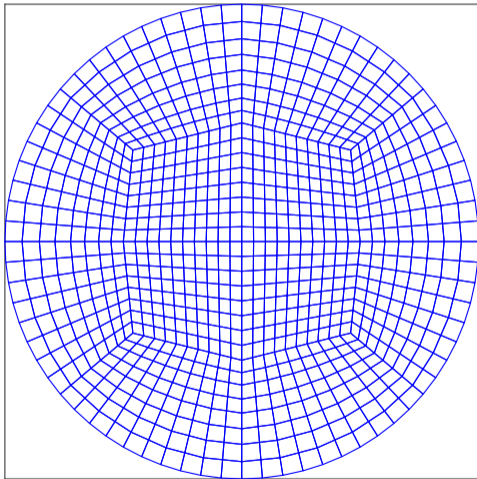


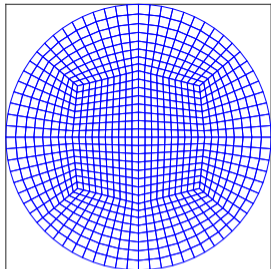
(f) \log_{10} misfit with respect to the cluster medoids.



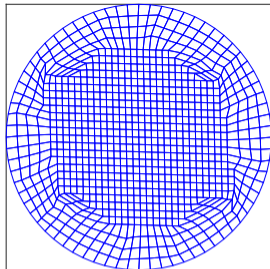
Example 3: Circle mesh.

- 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 $\gamma = 0.4$.

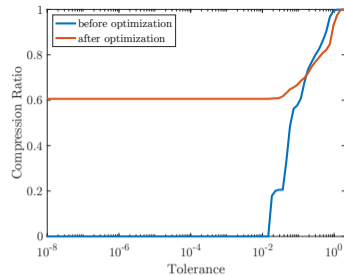




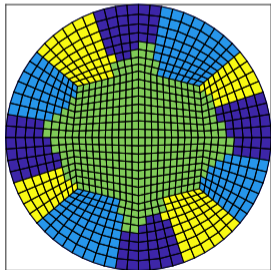
(a) Mesh before.



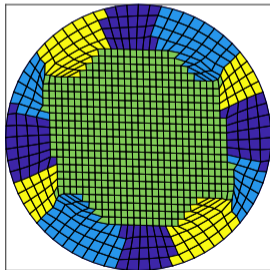
(b) Mesh after.



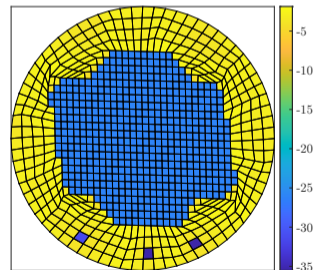
(c) Mesh compression.



(d) First cluster assignment.



(e) Last cluster assignment.



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





- 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.



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.



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.



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.



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. 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.



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](https://arxiv.org/abs/2410.07646)