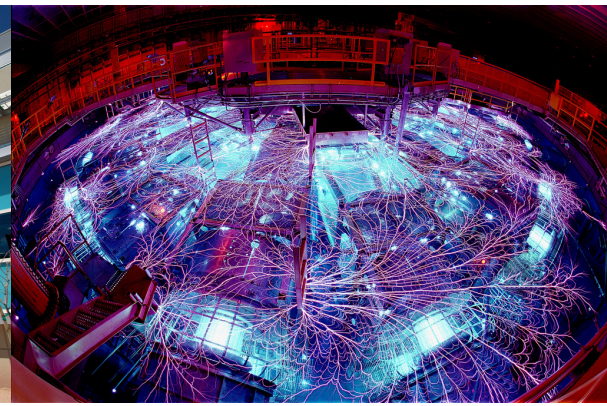
Exceptional service in the national interest

Sandia National Laboratories





Data-Driven Approaches for Patch-Based Multigrid Smoothers

Graham Harper*, Ray Tuminaro
Center for Computing Research
Sandia National Laboratories

4/18/23

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

Sandia National Laboratories is a multimission laboratory managed and opera
owned subsidiary of Honeywell International Inc., for the U.S. Department of E
DE-NA0003525.

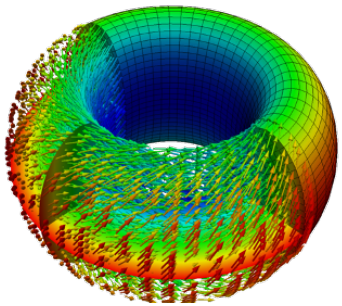
Acknowledgements

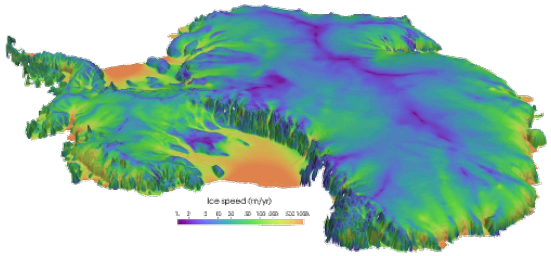
- Trilinos/MueLu/Ifpack2
- Funding: Harper's LDRD, Ridzal's LDRD, Tuminaro's ASCR
- Team: Ray Tuminaro
- Sandia National Laboratories LDRD Office, U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research

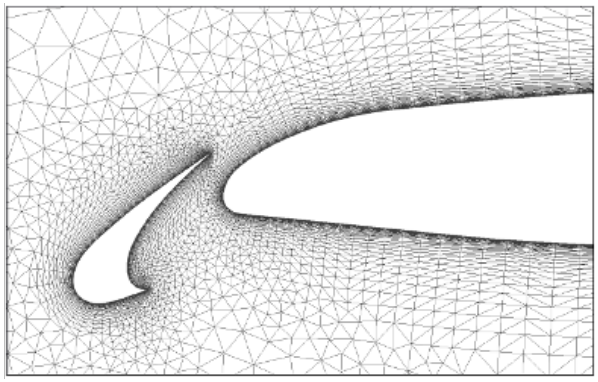
Outline

Overview for Smoothers

- Traditional simple smoothers (Jacobi) work well for low-order problems, but tend to struggle as the **polynomial degree** increases or **coupling** increases.

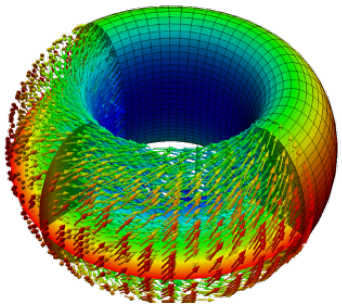


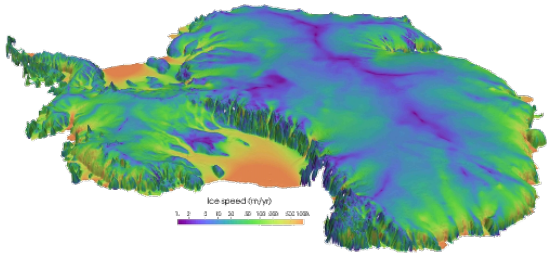


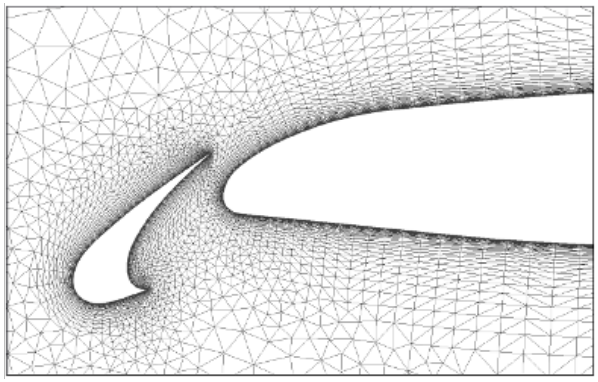


Overview for Smoothers

- Traditional simple smoothers (Jacobi) work well for low-order problems, but tend to struggle as the **polynomial degree** increases or **coupling** increases.



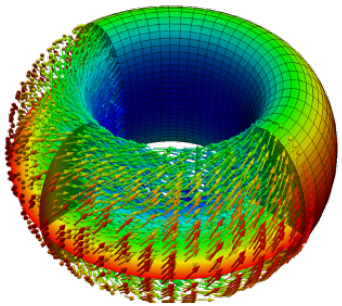


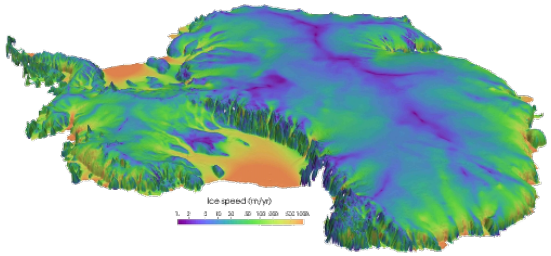


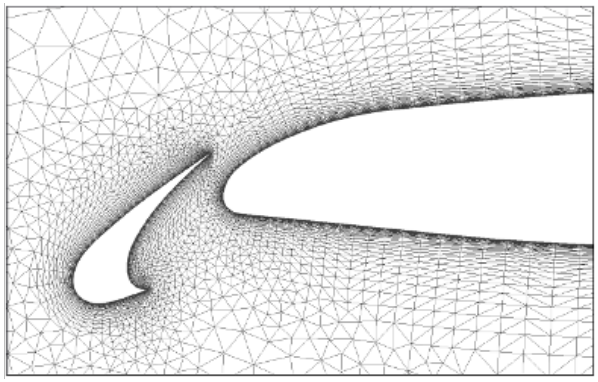
- Patch-based smoothers reduce iteration counts at the cost of storing/solving many small linear systems.

Overview for Smoothers

- Traditional simple smoothers (Jacobi) work well for low-order problems, but tend to struggle as the **polynomial degree** increases or **coupling** increases.







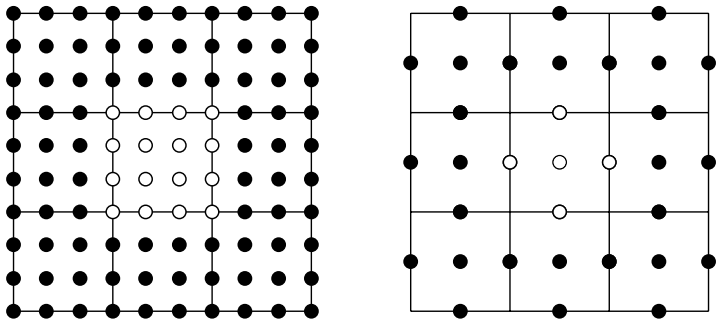
- Patch-based smoothers reduce iteration counts at the cost of storing/solving many small linear systems.
- What if we could reduce those costs?

Patch-Based Smoothers

- Recall a smoother takes the form

$$\mathbf{x} \leftarrow \mathbf{x} + \omega \tilde{M}^{-1}(A\mathbf{x} - \mathbf{b}) \tag{1}$$

- Consider an (overlapping) domain decomposition with n_p domains and boolean restriction operators R_i.

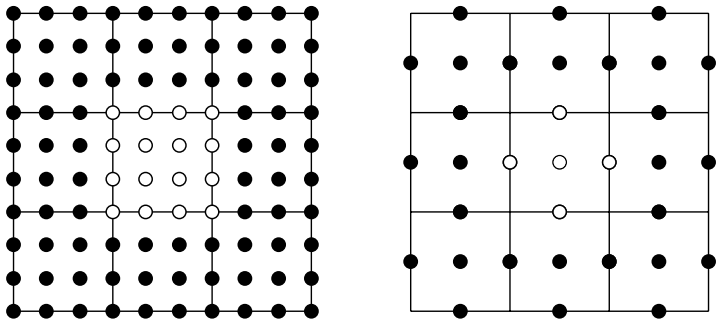


Patch-Based Smoothers

- Recall a smoother takes the form

$$\mathbf{x} \leftarrow \mathbf{x} + \omega \tilde{M}^{-1}(A\mathbf{x} - \mathbf{b}) \tag{1}$$

- Consider an (overlapping) domain decomposition with n_p domains and boolean restriction operators R_i.



- A patch-based smoother utilizes

$$\tilde{M}^{-1} = \sum_{i=1}^{n_p} R_i^T W_i A_i^{-1} R_i \tag{2}$$

Patch-Based Smoothers

$$\tilde{M}^{-1} = \sum_{i=1}^{n_p} R_i^T W_i A_i^{-1} R_i$$

- R_i is the boolean restriction
- W_i are the global weights, $(\text{overlap})^{-1}$
- $A_i = R_i A R_i^T$ is the $p_s \times p_s$ patch matrix

Patch-Based Smoothers

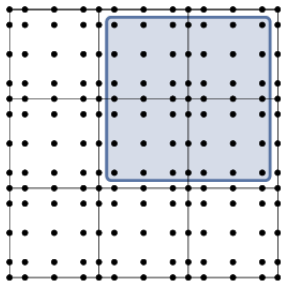
$$\tilde{M}^{-1} = \sum_{i=1}^{n_p} R_i^T W_i A_i^{-1} R_i$$

- R_i is the boolean restriction
- W_i are the global weights, $(\text{overlap})^{-1}$
- $A_i = R_i A R_i^T$ is the $p_s \times p_s$ patch matrix

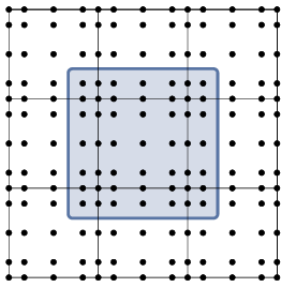
Patch methods:

- Pros: Converge more quickly, handle complex coupling more effectively
- Cons: More expensive to compute, similar storage cost to original matrix

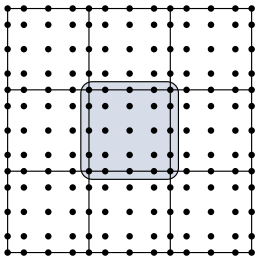
Clarifications



(a) Vertex-star patch



(b) Cell-centered patch



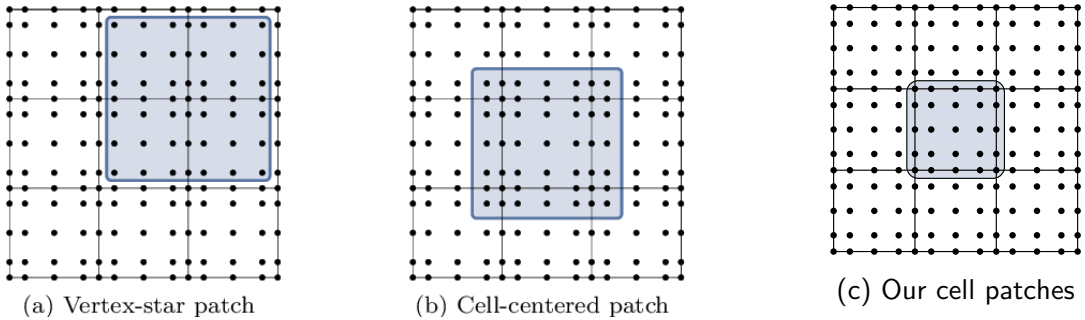
(c) Our cell patches

There are many kinds of patch smoothers:

(a) Scales for high values of p, difficult solves, depends on connectivity

*left figures: P. Brubeck, P. Farrell, *A Scalable and Robust Vertex-Star Relaxation for High-Order FEM*. SISC 2022

Clarifications



(a) Vertex-star patch (b) Cell-centered patch (c) Our cell patches

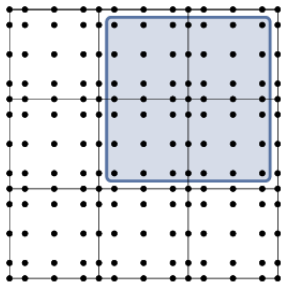
There are many kinds of patch smoothers:

(a) Scales for high values of p, difficult solves, depends on connectivity

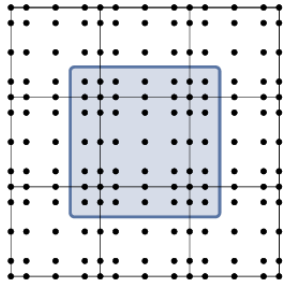
(b) Allows for structured patches, not p-robust

*left figures: P. Brubeck, P. Farrell, *A Scalable and Robust Vertex-Star Relaxation for High-Order FEM*. SISC 2022

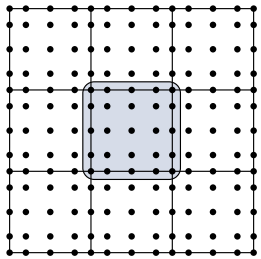
Clarifications



(a) Vertex-star patch



(b) Cell-centered patch



(c) Our cell patches

There are many kinds of patch smoothers:

(a) Scales for high values of p, difficult solves, depends on connectivity

(b) Allows for structured patches, not p-robust

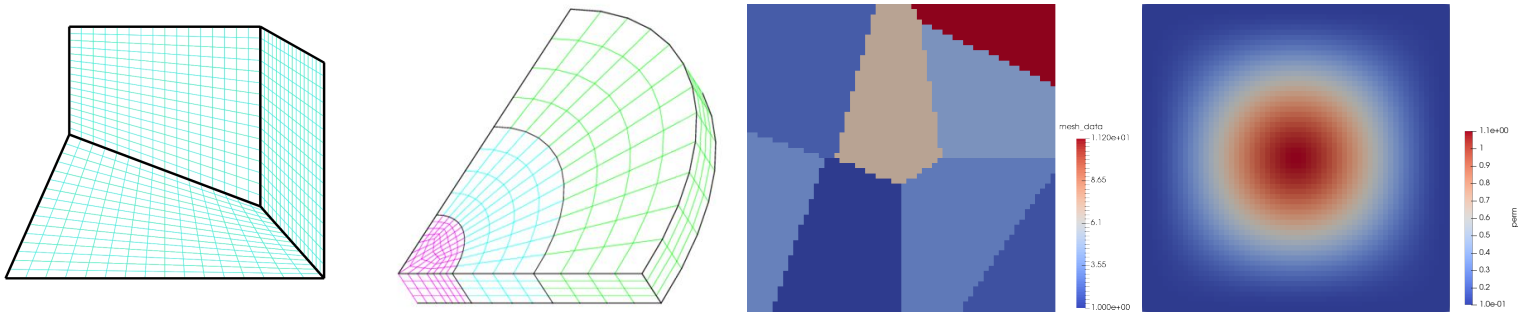
(c) Most efficient to detect, most structure to exploit

Bases with sparse patches also exist, but are not considered here.

*left figures: P. Brubeck, P. Farrell, *A Scalable and Robust Vertex-Star Relaxation for High-Order FEM*. SISC 2022

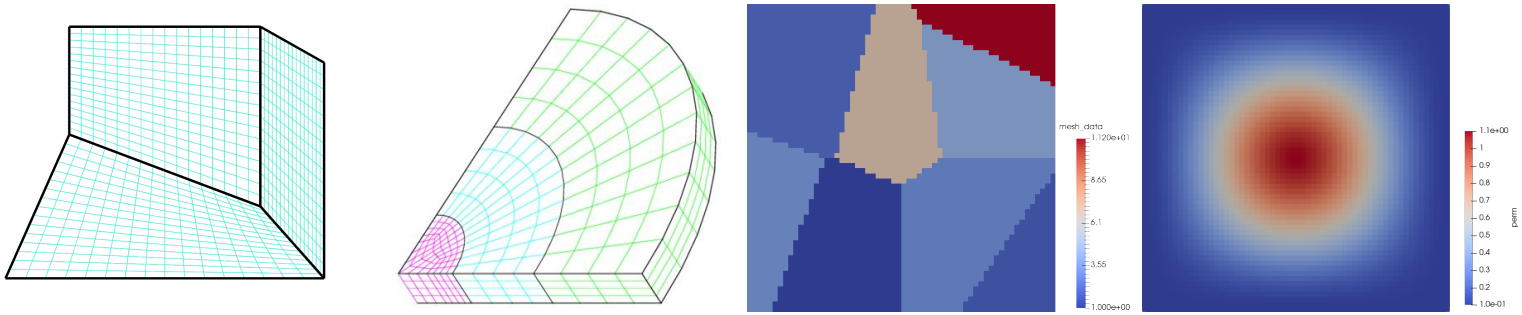
Observations

1. In many applications of interest, if you zoom in far enough, patterns appear.



Observations

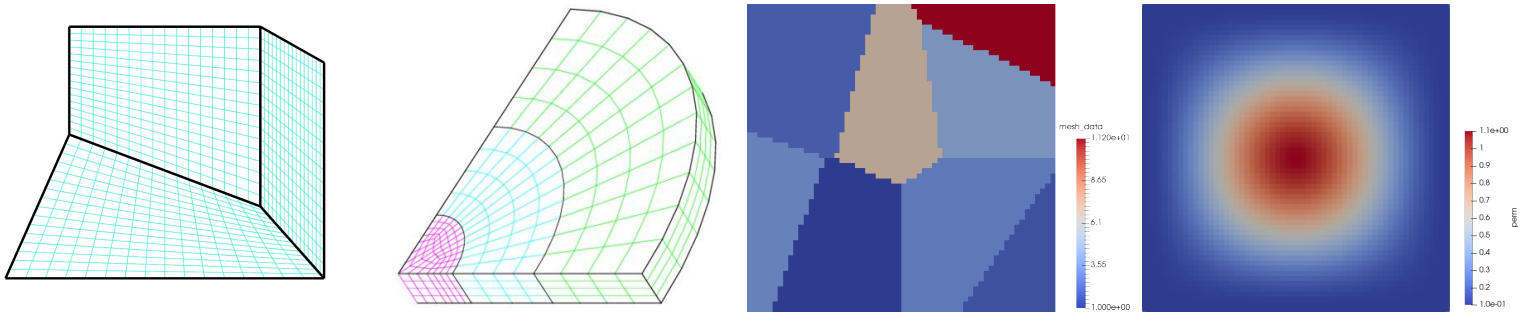
1. In many applications of interest, if you zoom in far enough, patterns appear.



2. If patches are "similar enough," why store and solve them multiple times?

Observations

1. In many applications of interest, if you zoom in far enough, patterns appear.



2. If patches are "similar enough," why store and solve them multiple times?

Question: How "bad" can the approximation get before the solver (smoother-only or multigrid) starts to struggle?

The Main Idea

- Construct a database of patches $\mathcal{B} = \{B_1, B_2, \ldots, B_{m_p}\}$.
- Criterion 1: $\mathcal{B}$ should approximate $\{A_i\}$ sufficiently well.
- Criterion 2: $\mathcal{B}$ should be small compared to $\{A_i\}$.

The Main Idea

- Construct a database of patches $\mathcal{B} = \{B_1, B_2, \ldots, B_{m_p}\}$.
- Criterion 1: $\mathcal{B}$ should approximate $\{A_i\}$ sufficiently well.
- Criterion 2: $\mathcal{B}$ should be small compared to $\{A_i\}$.

Seek $\mathcal{B} = \{B_1, B_2, \ldots, B_{m_p}\}$ and mapping
$\phi : \{1, \ldots, n_p\} \to \{1, \ldots, m_p\}$ minimizing

$$\mathcal{L}(\mathcal{B}, \phi) = \beta|\mathcal{B}| + \sum_{k=1}^{n_p} \|I - A_k B_{\phi(k)}^{-1}\|_2^2, \tag{3}$$

balancing efficiency and accuracy.
Also referred to as a **sparse approximation problem**.

Greedy Algorithm

Algorithm 1 Greedy Construction of $\mathcal{B}$, $\phi()$

1: **Input:** $\{A_1, A_2, \ldots, A_{n_p}\}$, ε
2: $\mathcal{B} := \{\}$, $\vec{\phi} = 0$
3: **for** $i = 1, \ldots, n_p$ **do**
4: match:=false;
5: **for** $j = 1, \ldots, m_p$ **do**
6: **if** $\|I - A_i B_j^{-1}\|_2 < \varepsilon$ **then**
7: match=true, $\phi(i) = j$, break;
8: **end if**
9: **end for**
10: **if** match==false **then**
11: append A_i to $\mathcal{B}$, $\phi(i) = |\mathcal{B}|$;
12: **end if**
13: **end for**
14: **Output:** $\mathcal{B} = \{B_1, B_2, \ldots, B_{m_p}\}$, $\vec{\phi}$

← check patch against database

← if no matches, store patch explicitly

Clustering Algorithm

Algorithm 3 Clustering Construction of $\mathcal{B}$, $\phi()$

1: **Input:** $\{A_1, A_2, \ldots, A_{n_p}\}$, m_p
2: $\vec{\phi} = \text{randperm}(n_p, m_p)$;
3: $B_i = A_{\phi(i)}$, $i = 1, \ldots, m_p$;
4: **while** not converged **do**
5: **for** $i = 1, \ldots, n_p$ **do**
6: **for** $j = 1, \ldots, m_p$ **do**
7: $d_{ij} = d(A_i, B_j)$;
8: **end for**
9: $\phi(i) = \text{argmin}_j(d_{ij})$;
10: **end for**
11: $B_i = \frac{1}{n_i} \sum_{j \text{ in cluster } i} A_j$, $i = 1, \ldots, m_p$;
12: **end while**
13: **Output:** $\mathcal{B} = \{B_1, B_2, \ldots, B_{m_p}\}$, $\vec{\phi}$

$\leftarrow$ choose number of clusters

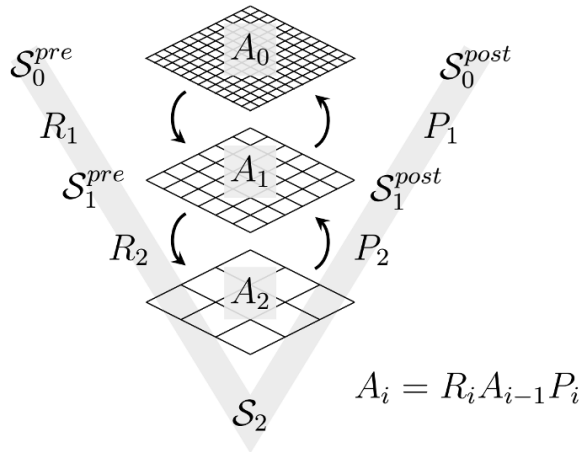
$\leftarrow d(A_i, B_j) = \|I - A_i B_j^{-1}\|_2$

*treat boundaries separately!

Multigrid Method

We use the following multigrid configuration:

- $\omega = 0.5$,
- $(\nu_1, \nu_2) = (1, 0)$,
- $N = 2$ V-cycle levels,
- P linear interpolation, $R = P^T$,
- $A_c = RAP$



With the data-driven smoother, compute $\mathcal{B}, \vec{\phi}$ during setup; use them during apply.

Example Test Cases

The following examples utilize Poisson's equation:

$$(k\nabla p_h, \nabla q_h)_{\mathcal{T}_h} = (f, q_h)_{\mathcal{T}_h}. \tag{4}$$

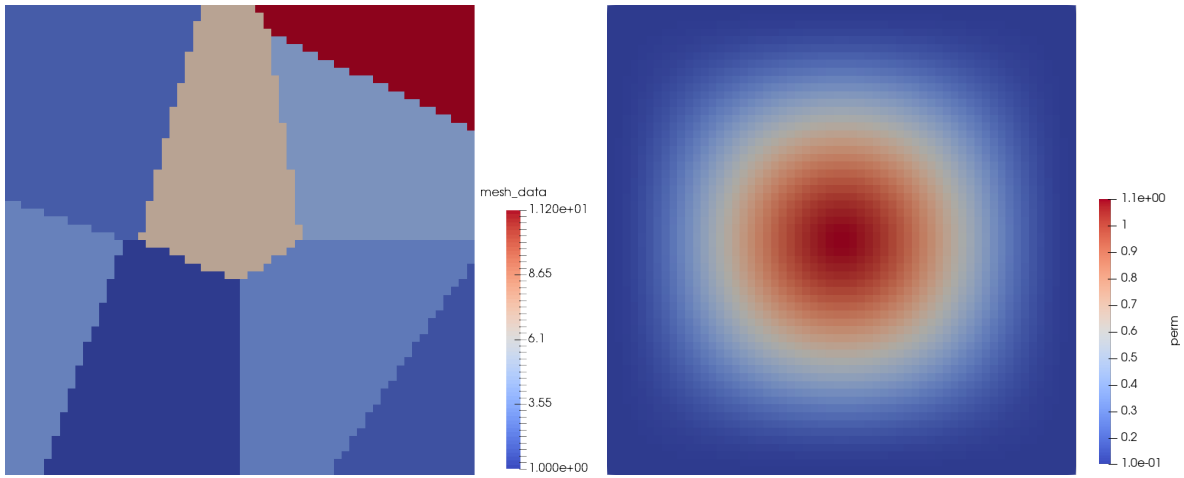
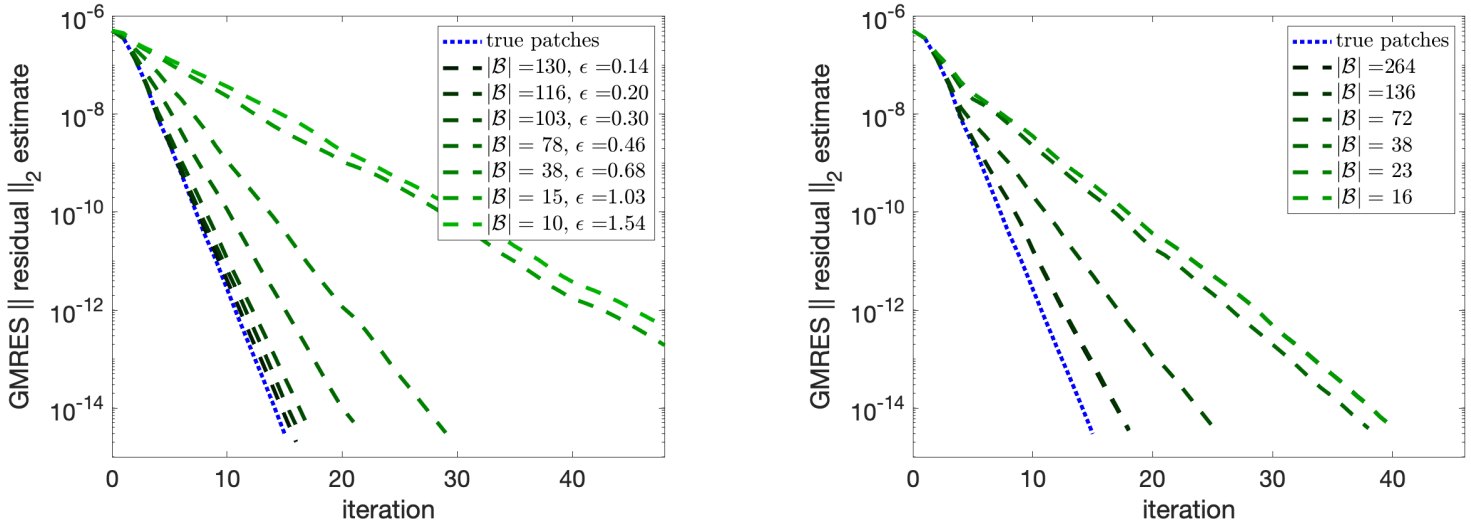


Figure: (left) piecewise discontinuous coefficient, (right) smooth varying coefficient

Results (Piecewise Constant Permeability)

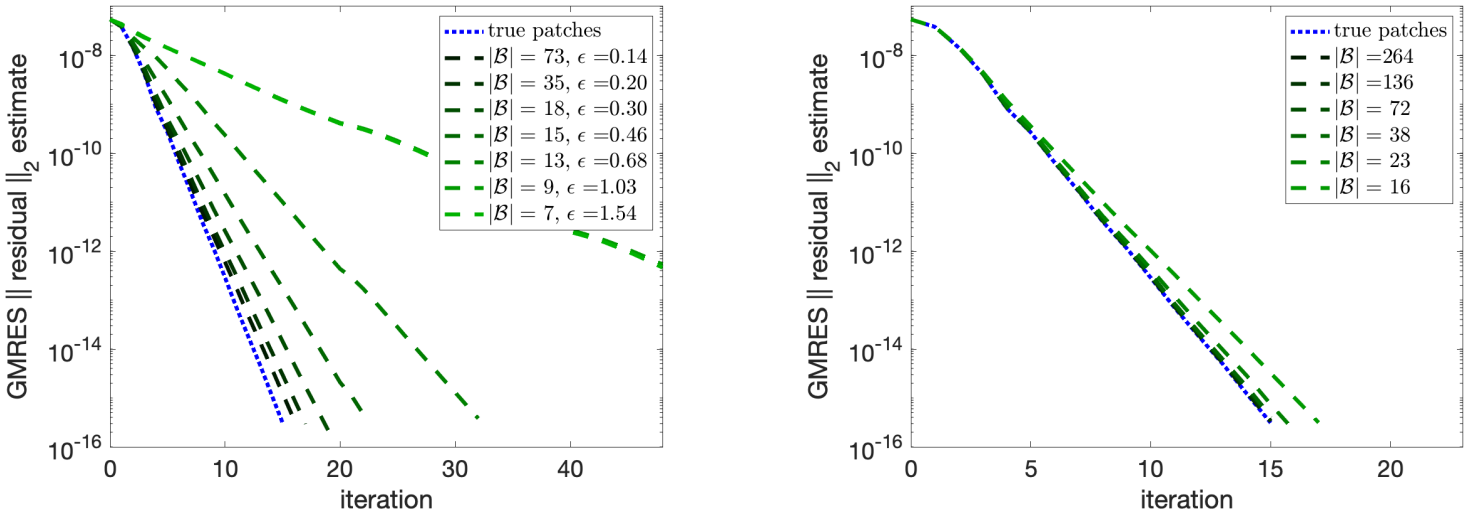


(a) greedy algorithm

(b) clustering algorithm

Figure: Multigrid residual histories on 60×60 mesh with $p = 5$** for piecewise constant problem, varying database sizes $\mathcal{B}$.

Results (Smooth Permeability)



(a) greedy algorithm

(b) clustering algorithm

Figure: Multigrid residual histories on 60×60 mesh with $p = 5$ for smooth problem, varying database sizes $\mathcal{B}$.

Results (Database Mappings)

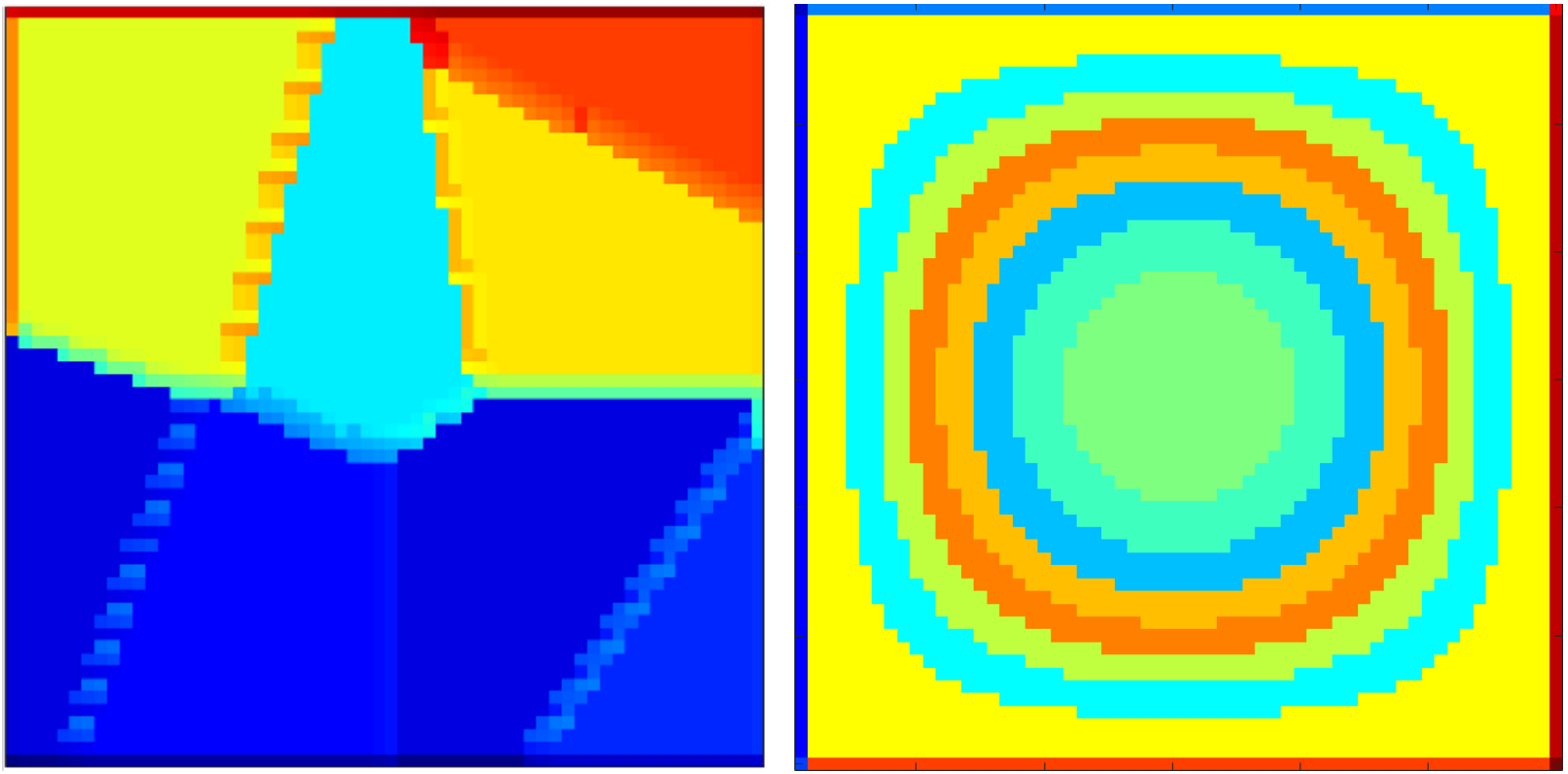


Figure: Visualization of database ϕ mapping. (left) greedy algorithm on piecewise constant permeability, (right) clustering algorithm on smooth permeability

Results (Compressibility Profile)

For Burgers' equation with a shock on a 100×100 mesh:

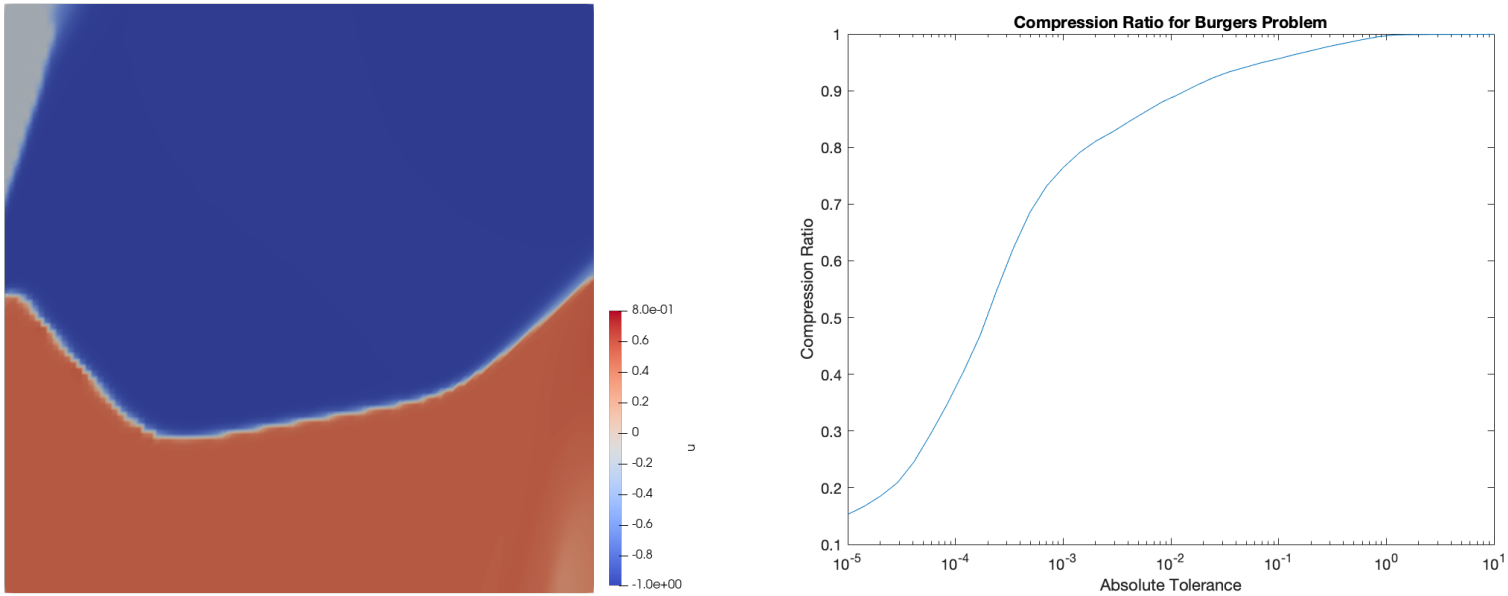


Figure: (left) Burgers' equation profile with a shock, (right) the compressibility profile associated with the problem

Timing Results

Implemented in **Trilinos/Ifpack2** as a preconditioner. Used as additive Schwarz inner solve.

Configuration	m_p	$p_s \times p_s$	Setup (s)	Apply (s)
No database	3600	36×36	2.274	0.4157
Database	9	36×36	2.442	0.2624

Apply speedup of 30% on small test problem on an Intel Xeon workstation. *Expected to extend well.

Interested in trying it?
`https://github.com/trilinos/Trilinos/pull/11517`

Conclusion

Takeaways:

- Memory cost $= \mathcal{O}(\text{Jacobi})$
- Faster apply than without database
- Generalizes for any structure detection method

Current/Future Work:

- Performance improvements
- Time-dependent problems
- Ridzal's LDRD: R-adaptivity to detect and enhance compressibility
- Summer student looking into clustering on eigenvalues
- Unsupervised deep learning

`https://github.com/trilinos/Trilinos/pull/11517`

Thank You

- Questions?