Efficient implementation of polytopal Discontinuous Galerkin methods with application to multilevel solvers

Pasquale Claudio Africa
pafrica@sissa.it
SISSA International School for Advanced Studies

Marco Feder, Andrea Cangiani (SISSA),
Luca Heltai (University of Pisa)

deal.II workshop, 13 Aug 2024
Fort Collins (CO)

SISSA

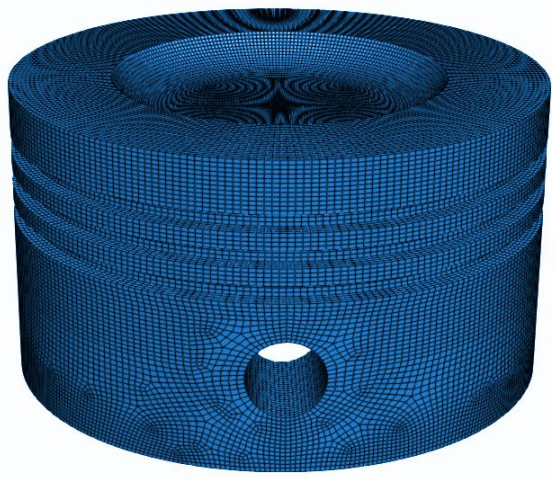
sissa mathLab
innovating with mathematics

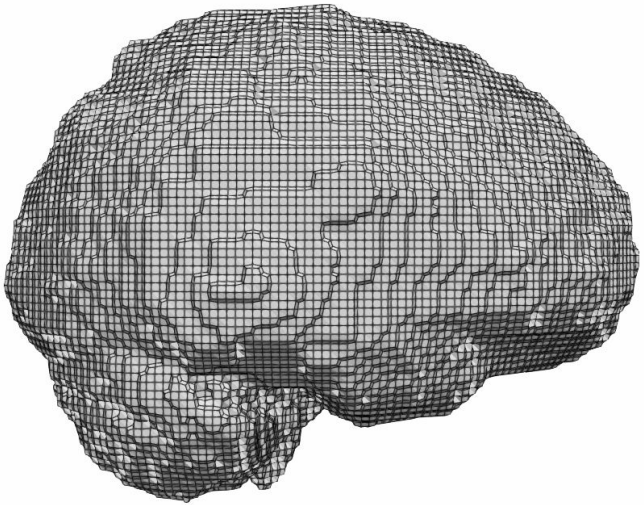
UNIVERSITÀ DI PISA

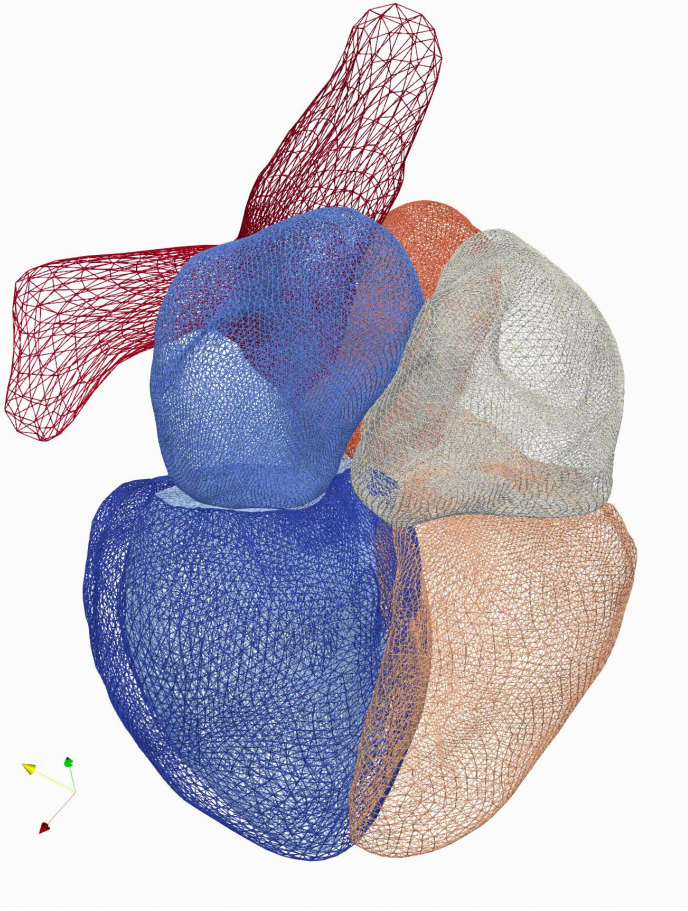
Outline

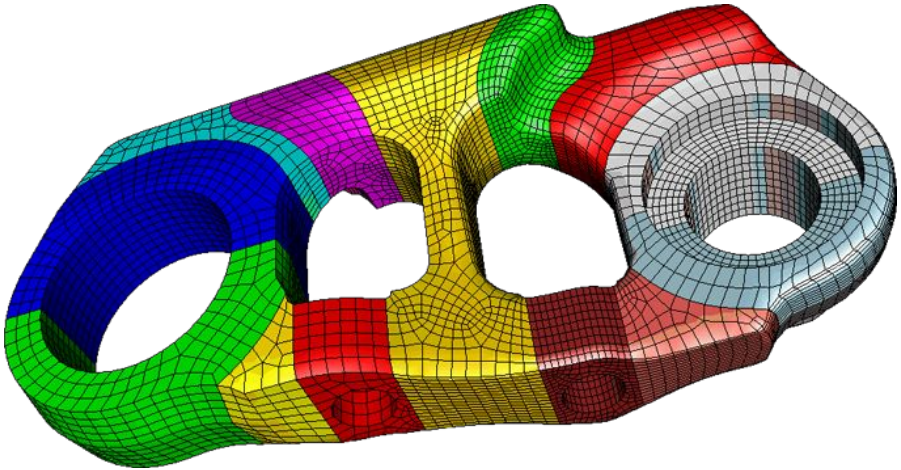
1. Motivation
2. R-tree data structure for FEM
3. Mesh agglomeration: METIS vs. R-tree
4. Polytopal DG + Agglomerated GMG
5. Implementation + parallelization strategies
6. Application to cardiac electrophysiology

Motivation









Motivation

- Traditional Finite Element Methods (FEMs) may struggle to accurately represent complex geometries coming from real-world applications.
- Grid generation is often a bottleneck due to the limitations of using only tetrahedral, hexahedral, or prismatic elements.
- Polytopal methods (polygonal FEM, mimetic FD, VEM, PolyDG, HDG, HHO) have shown to be particularly effective in representing complex geometries.
- Polytopal methods are attractive in this respect since *coarse* grids can be simply generated by merging polygonal and polyhedral elements.

➔ Providing automated and good-quality **agglomeration strategies** for polygonal and polyhedral elements remains an open and challenging task.

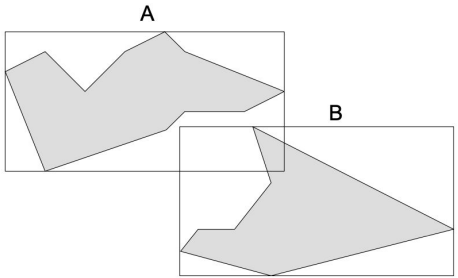
➔ It is indeed crucial to **preserve the original mesh quality**, as any deterioration could potentially affect the overall performance of the method in terms of **stability and accuracy**.

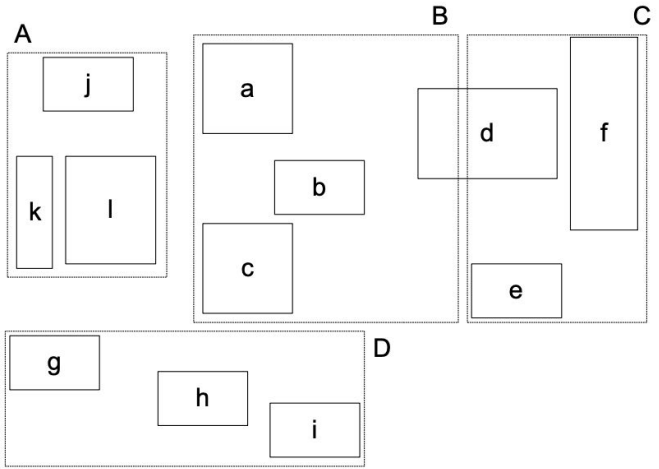

WHO ARE WE?
DEAL.II USERS
WHAT DO WE WANT?
SIMULATIONS
OF COMPLEX MODELS
WHEN DO WE WANT IT?
YESTERDAY!
imgflip.com

R-tree data structure for FEM

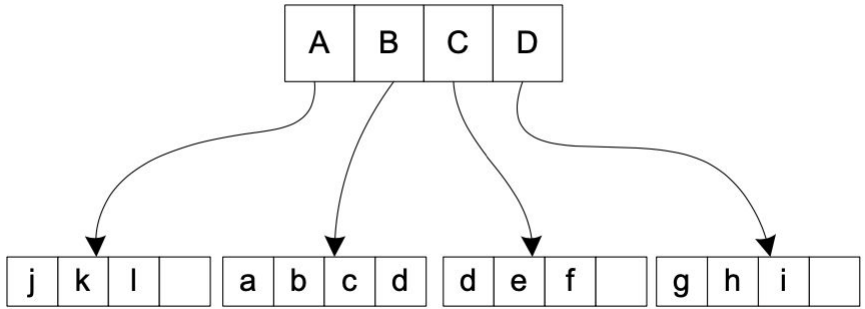
R-tree data structure

- Used for the dynamical organisation of generic geometric objects
- Objects represented by their bounding box



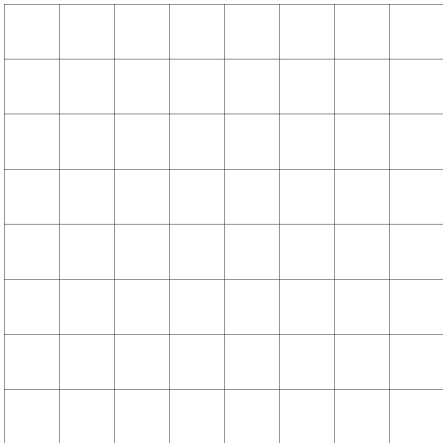


- Leaves of the tree: real objects
- Internal node stores:
  - *Pointer* to child node
  - *Box* of all entries within child node

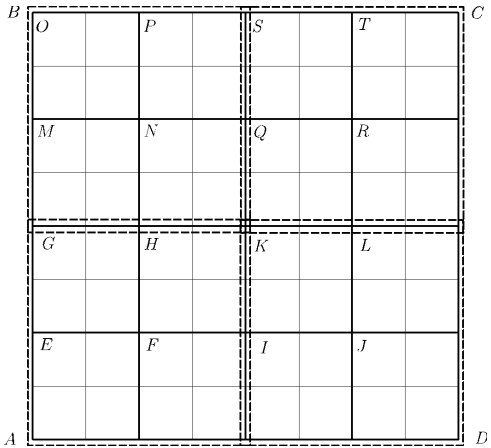


R-tree and finite elements

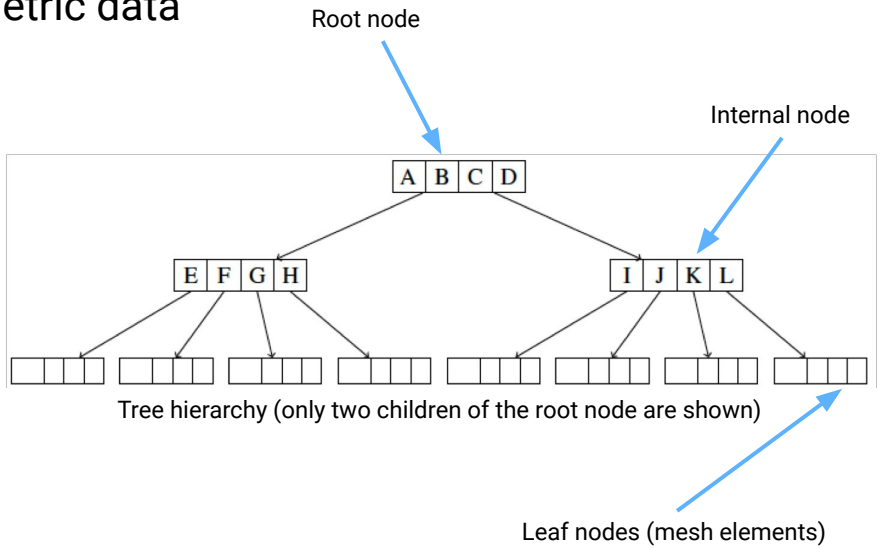
- Basic idea: use mesh elements as the geometric data
- Represents object by their bounding box



Original 8x8 mesh $(0, 1)^2$



Bounding boxes and mesh elements



- **A** has child node with entries: **E**, **F**, **G**, **H**
- Each box bounds 4 mesh elements

Mesh agglomeration: METIS vs. R-tree

Agglomeration strategy

Overall procedure

- *Build* R-tree out of bounding boxes $\{\text{BBOX}(T_i)\}_{i=1,\dots,N}$
- *Select* target level $l \in \{1, \dots, L\}$
- For every node on level, *descend* recursively its children until leaf nodes are reached
- *Agglomerate* together leaf nodes sharing the same ancestor

- Proper parent-child relationships
- Generate sequence $\{\mathcal{T}_k\}_k$ of nested, *agglomerated*, meshes for **multilevel methods**
- Intergrid transfers are cheap: canonical injection between nested spaces (cfr. non-nested multigrid)

- We compare to METIS, popular way to generate polytopic meshes

Algorithm 1 Creation of agglomerates.

Input: R-tree R of order (M, m)
Input: $l \in \{1, \dots, L\}$ target level.
Output: v vector s.t. $v[n]$ stores leafs associated to node n

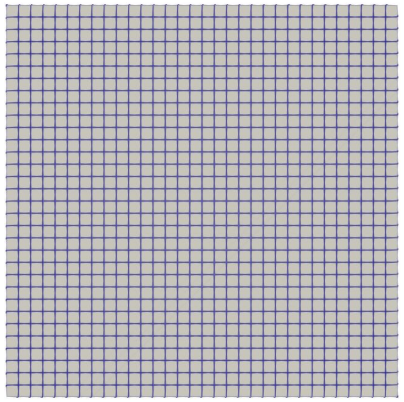
```
1: function ComputeAgglomerates(R, l)
2:     for node n in N_l do
3:         v[n] ← ExtractLeafs(l, n)
4:     end for
5: end function
```

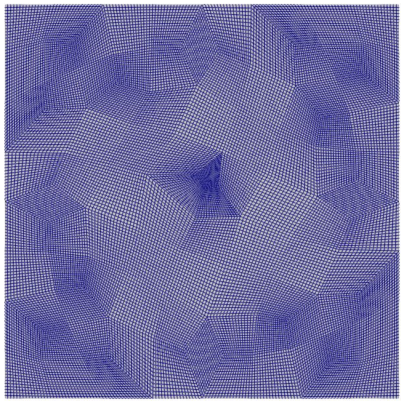
Algorithm 2 Recursive extraction of leafs from a node n on level l.

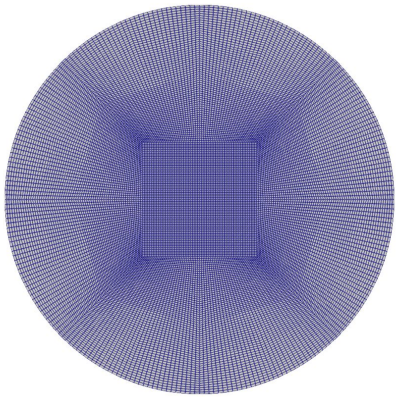
Input: $l \in \{1, \dots, L\}$ target level
Input: $n \in N_l$
Output: vector $v[n]$ containing leafs which share the ancestor node n.

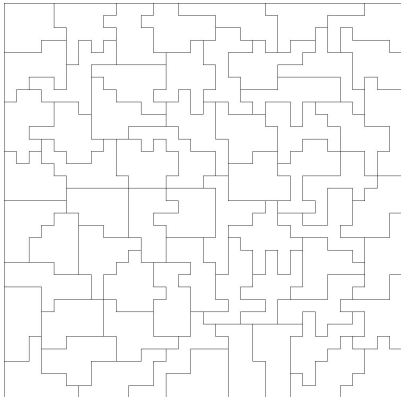
```
1: function ExtractLeafs(l, n)
2:     if l = 1 then
3:         v[n] ← {T_n^1, ..., T_n^M}
4:     else
5:         ExtractLeafs(l − 1, n)
6:     end if
7: end function
```

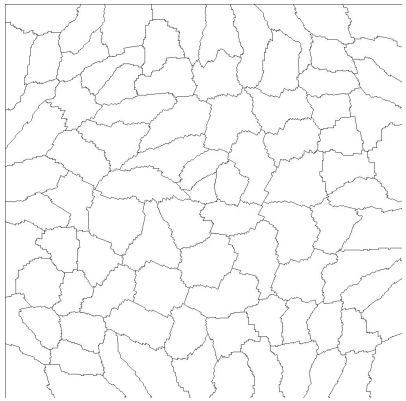
2D examples (METIS – graph *k*-way)

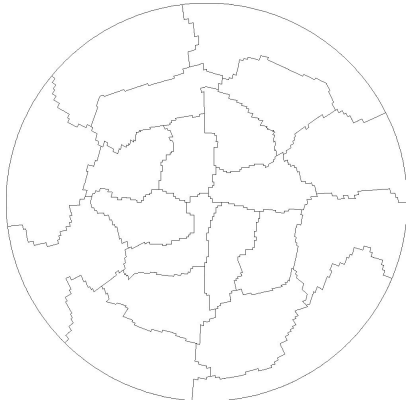




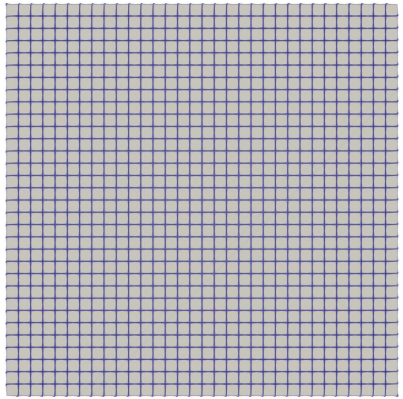


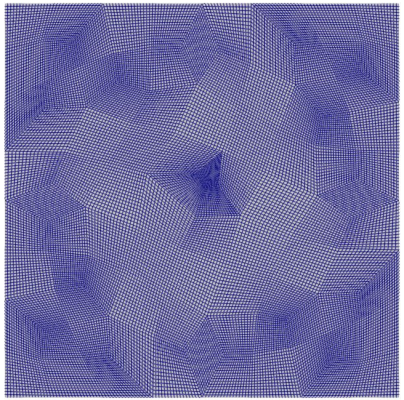


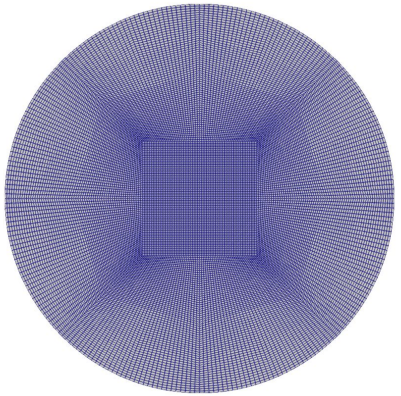


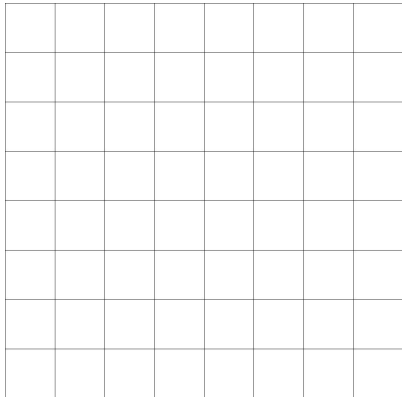


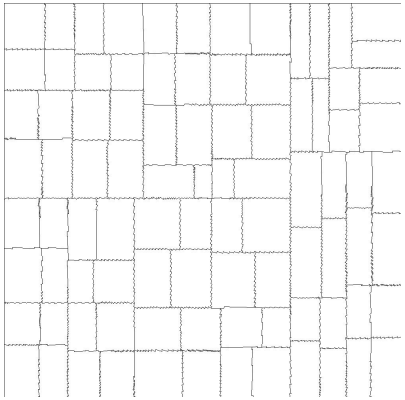
2D examples (R-tree)

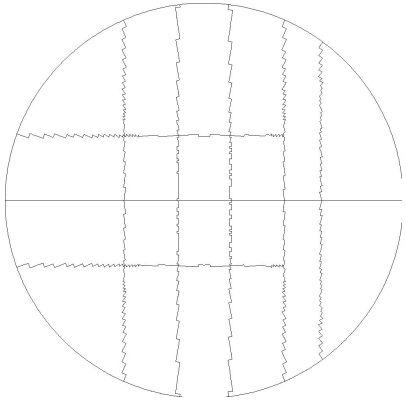




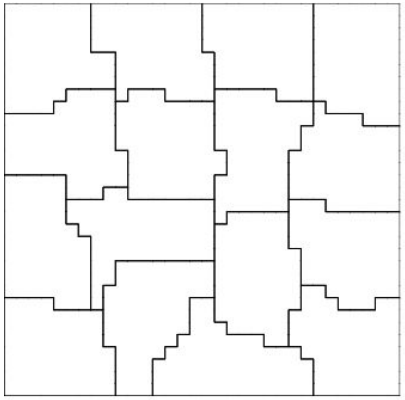




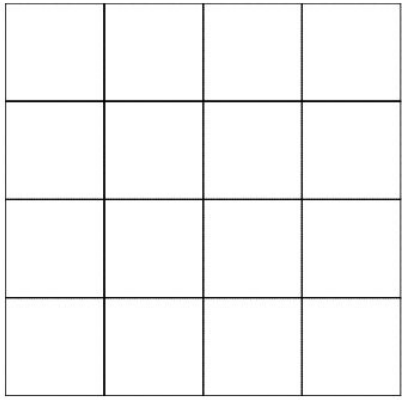




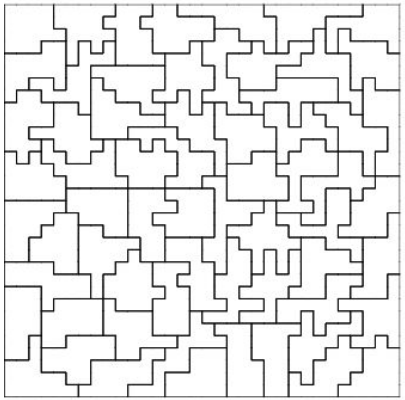
2D example: structured square



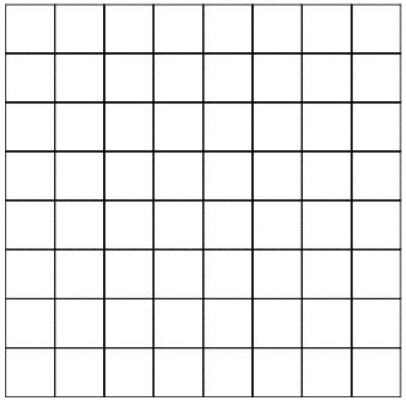
(a) METIS, n_partitions = 16.



(b) R-tree, extraction_level = 2.



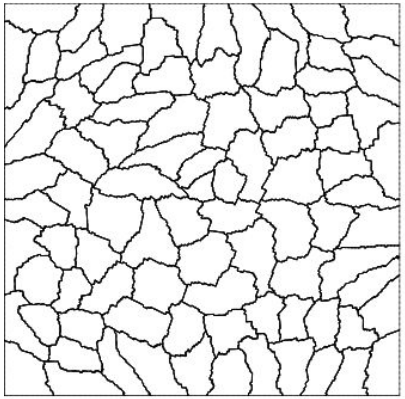
(c) METIS, n_partitions = 64.



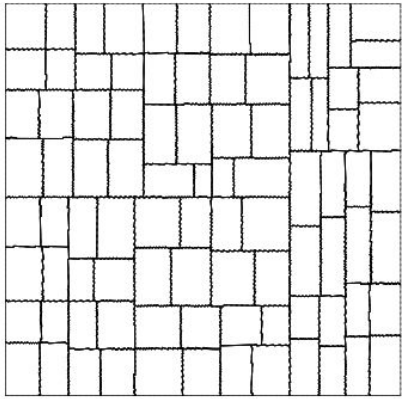
(d) R-tree, extraction_level = 3.

METIS: Geometrical features not preserved!

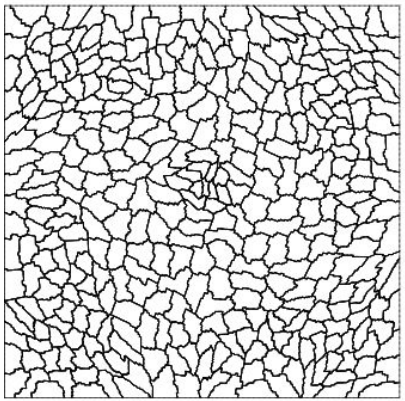
2D example: unstructured square



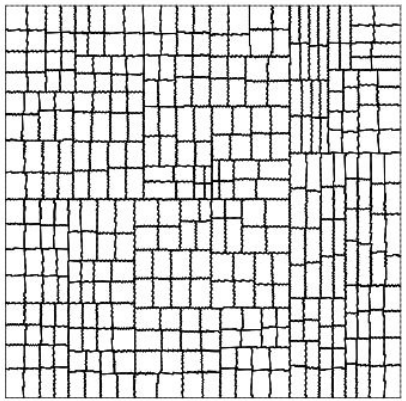
(a) METIS, `n_partitions` $= 91$.



(b) R-tree, `extraction_level` $= 4$.

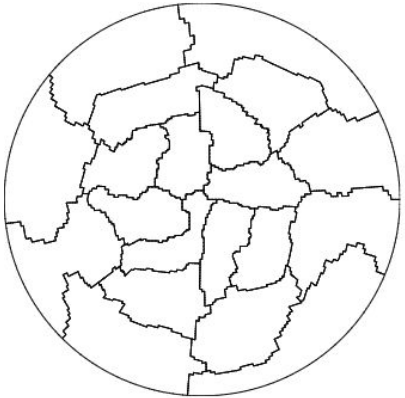


(c) METIS, `n_partitions` $= 364$.

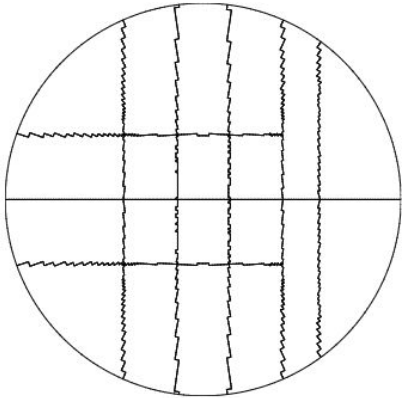


(d) R-tree, `extraction_level` $= 5$.

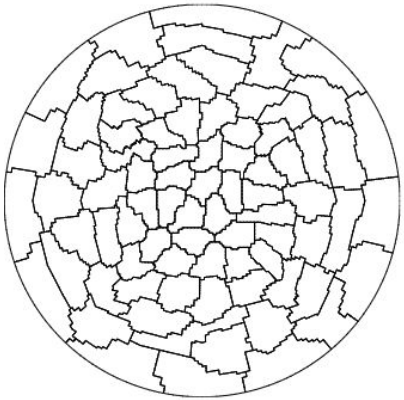
2D example: structured ball



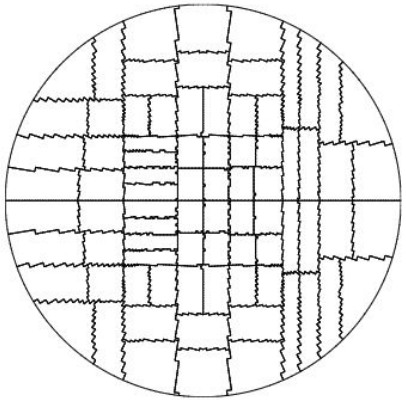
(a) METIS, `n_partitions` = 20.



(b) R-tree, `extraction_level` = 3.

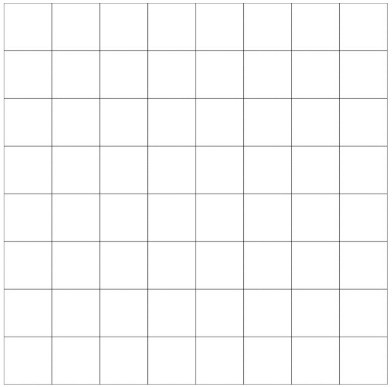


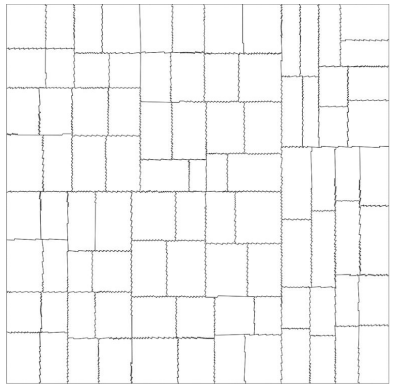
(c) METIS, `n_partitions` = 80.

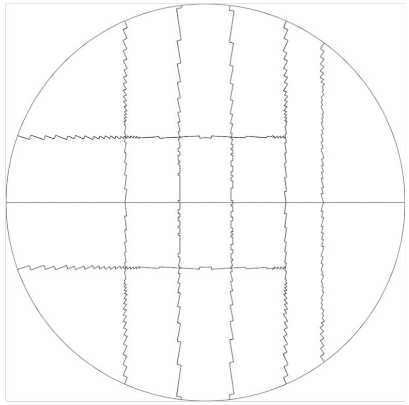


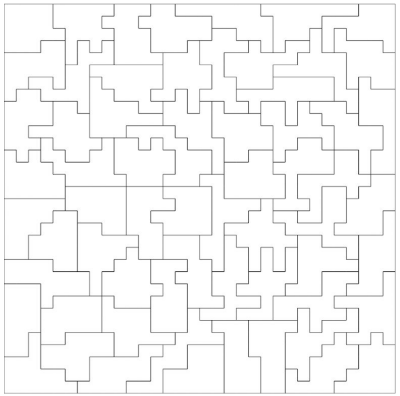
(d) R-tree, `extraction_level` = 4.

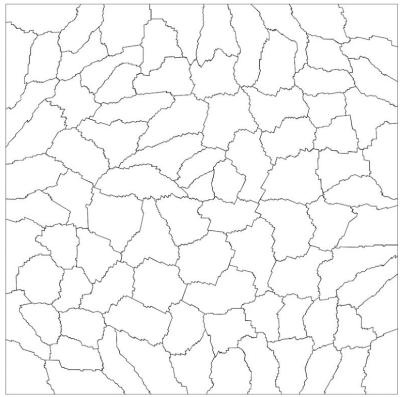
Visual comparison

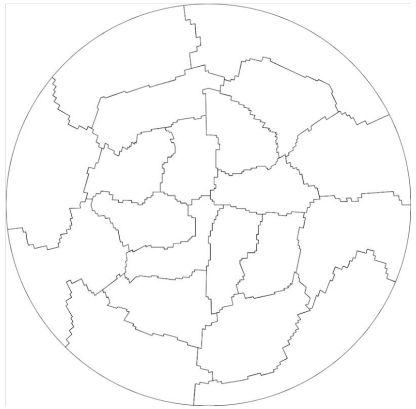












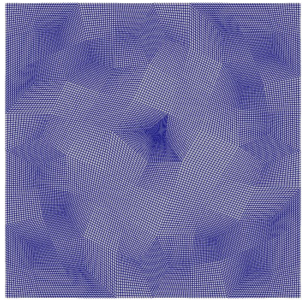
R-tree strategy:

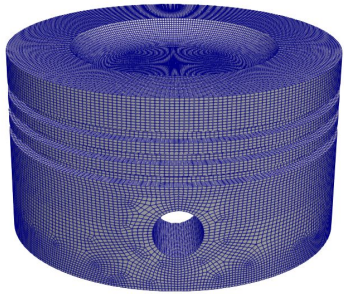
- Produces agglomerates tightly close to each bounding box
- Preserves geometrical features
- Better quality indicators for polytopal elements (uniformity ratio, circle ratio, box ratio)

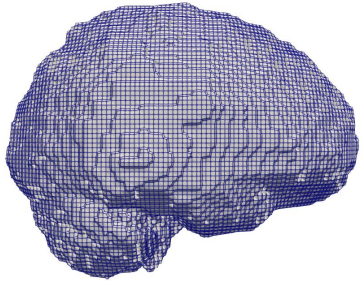
How *fast* is it?

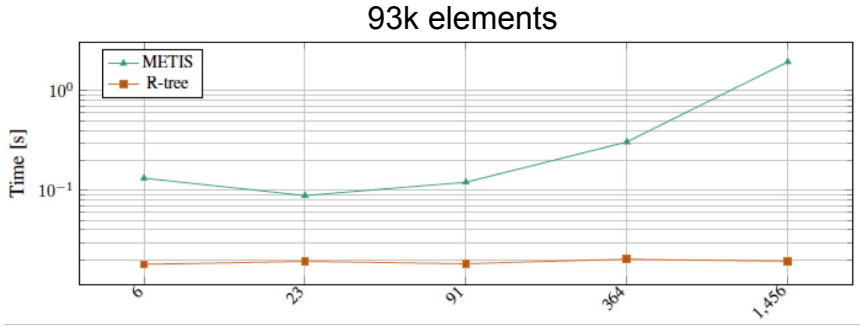
- We measure wall-clock time against graph *k*-way algorithm implemented by METIS

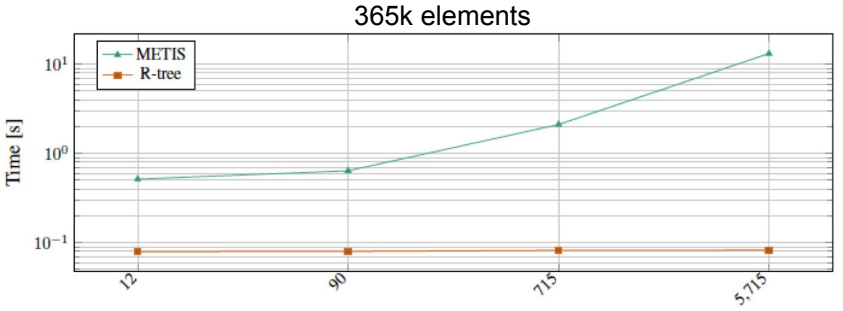
- We time (cumulatively):
  - Building the R-tree
  - Visiting tree and the hierarchy
  - Flagging original mesh cells

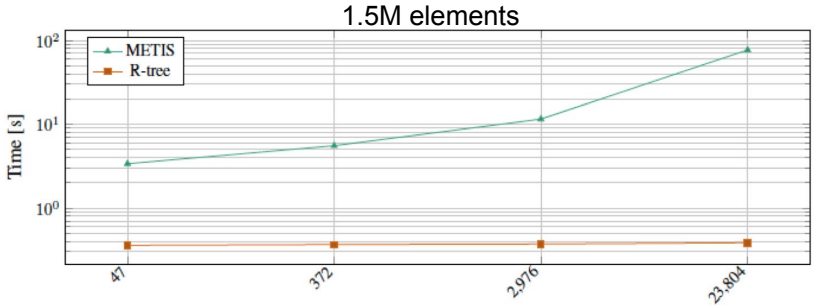












R-tree: targets and properties

Targets

- Minimisation of area covered by each box
- Minimisation of overlap between boxes
- Minimisation of boxes' perimeters

Several possible variants.
Choice: ***R*-tree***

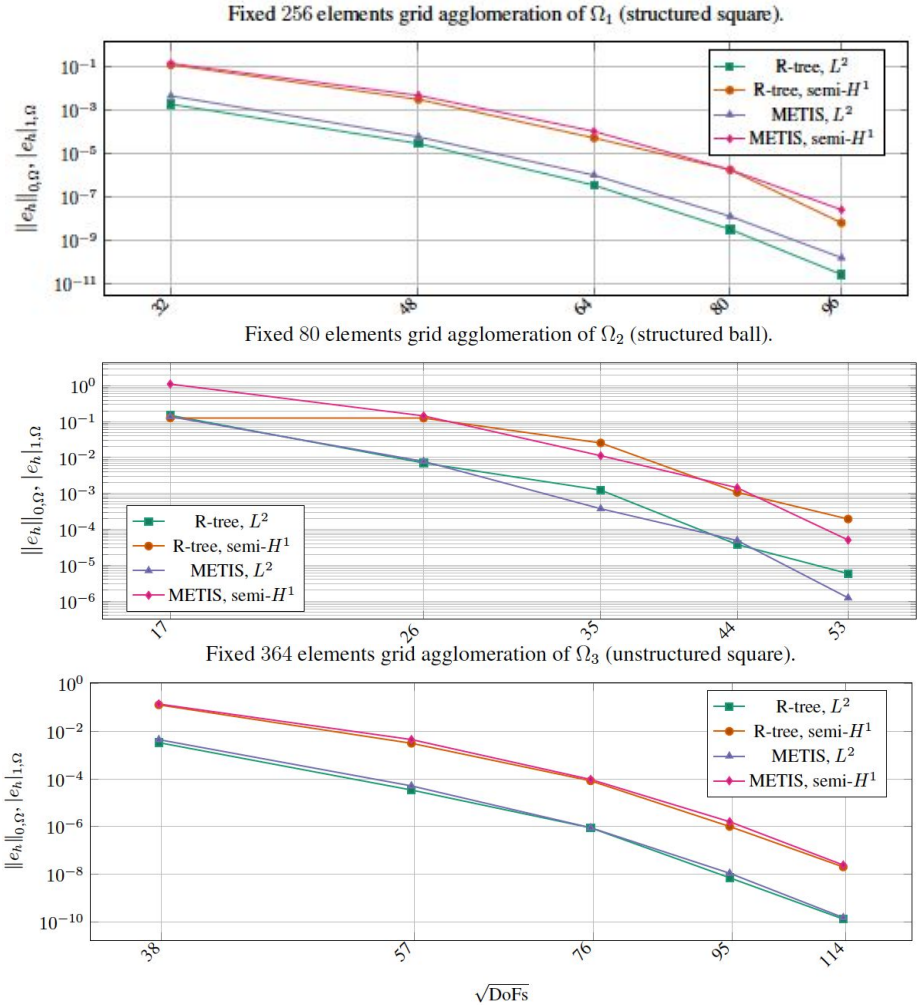
Properties

- R-tree shows constant wall-clock times
- *Independent* of the extraction level *l*
- R-tree manipulation:
  - Relies on `Boost.Geometry` library
  - Hierarchy traversal *not* natively supported
  - Requires implementation of node *visitor*

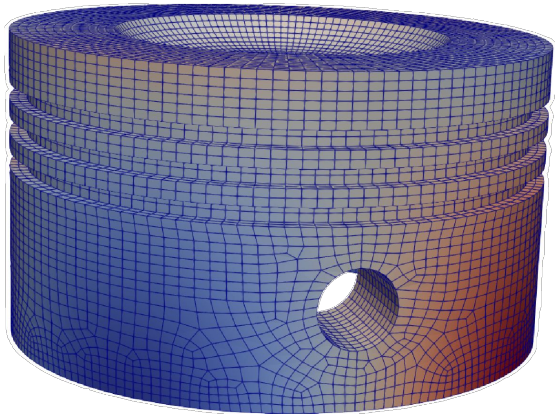
Polytopal DG + Agglomerated GMG

Polytopal DG

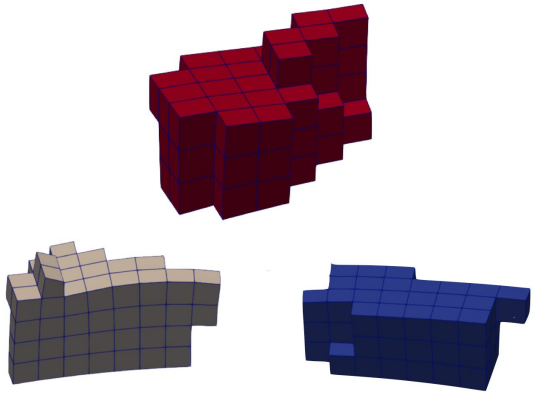
- Discontinuous nodal space defined on bounding boxes
- *p*-convergence for a simple diffusion-reaction equation
- Quadrature formulas coming from sub tessellation, readily available from underlying mesh



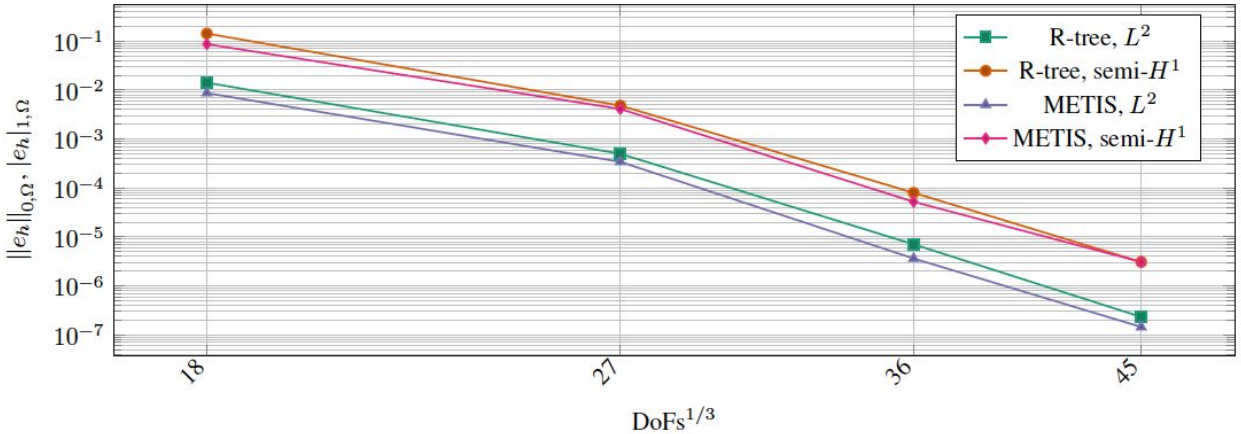
Polytopal DG: 3D piston model



	Number of DoFs	
p	Number of DoFs (original mesh)	Number of DoFs (agglomerated mesh)
1	365,712	25,425
2	1,234,278	48,366
3	2,925,696	108,492
4	5,714,250	175,020



Fixed 731 elements grid agglomeration of 3D piston model.



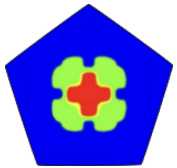
Implementation + parallelization strategies

Implementation

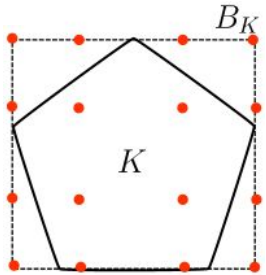
- Open-source C++ library `polyDEAL`: https://github.com/fdrmrc/Polydeal
- Builds on top of the `deal.II` library
- PETSc & `Trilinos` as linear algebra backends
- Distributed memory implementation handled through MPI
- Minimal changes required:
  `DoFHandler<dim>` → `AgglomerationHandler<dim>`
- Memory-distributed agglomerated multigrid
- Iterators on mesh-like containers:

```
for (const auto &polytope : agglomeration_handler.active_polytope_iterators())
  {
    if (polytope->is_locally_owned())
      {
        // do stuff with polytope
      }
  }
```

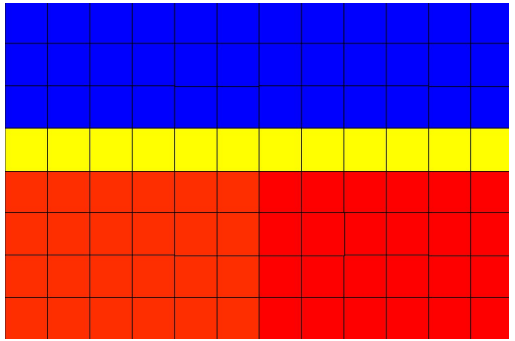
```
for (const auto &cell : dof_handler.active_cell_iterators())
  {
    if (cell->is_locally_owned())
      {
        // do stuff
      }
  }
```



Parallelization strategies



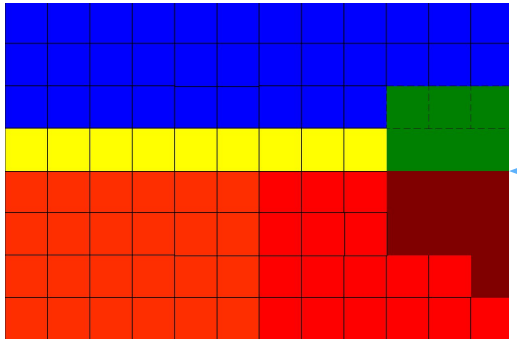
- Standard nodal DG space over the bounding box of each polytope
- Agglomeration is performed within each locally owned partition



Rank 0

Ghost

Rank 1



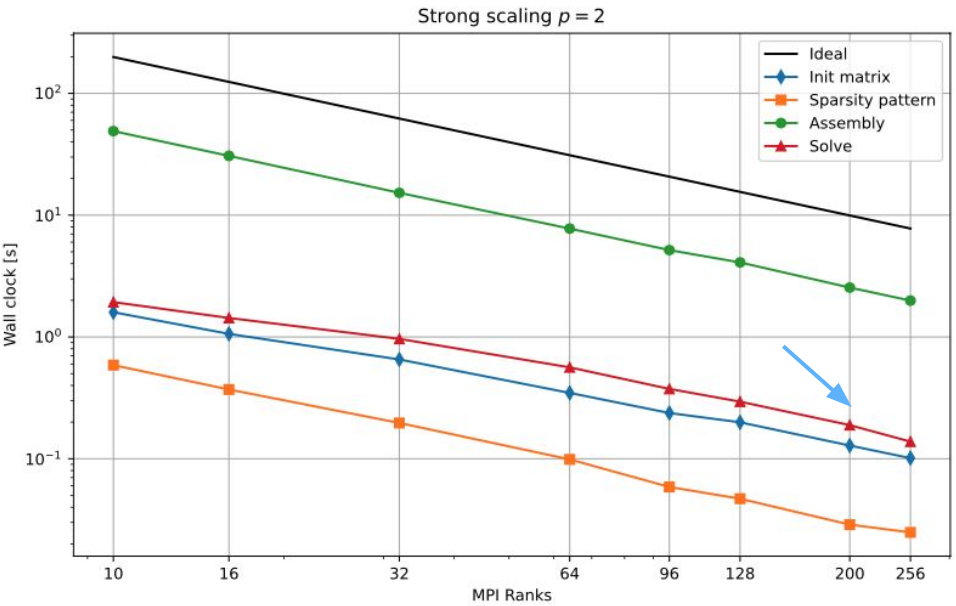
- Green polytope is ghosted
- Sub-elements are in general remote

- Flux DG terms require knowledge of metadata that are *not* just ghosted, but *fully* remote
- MPI non-blocking communication during setup phase on processors' boundaries

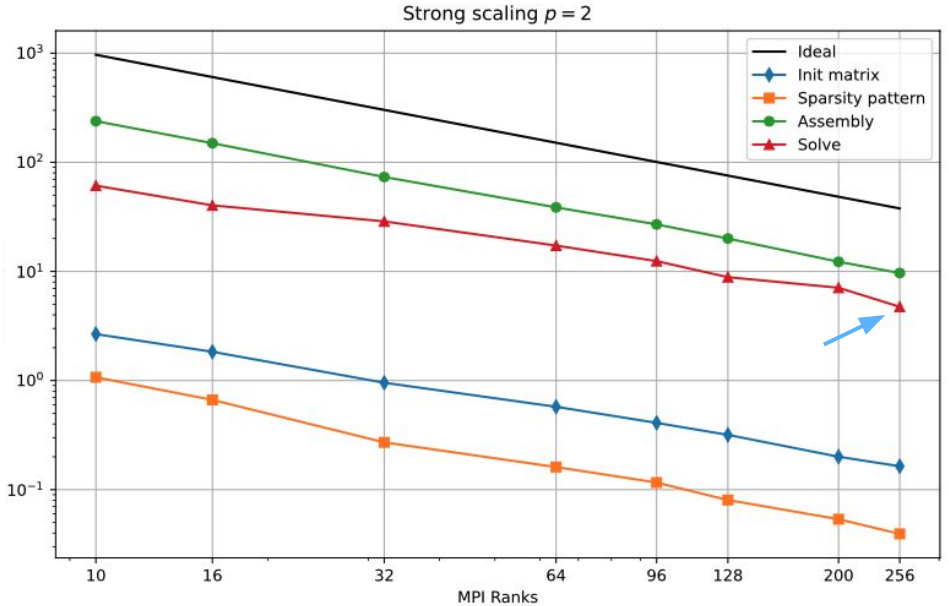
Benchmarks

- $-\Delta u = f$ on unit cube, Q^2 elements
- Solver: CG + AMG (`TrilinosML`)
- Complexity shifted to assembly phase

From **262k** original hexas → **7k** agglomerates

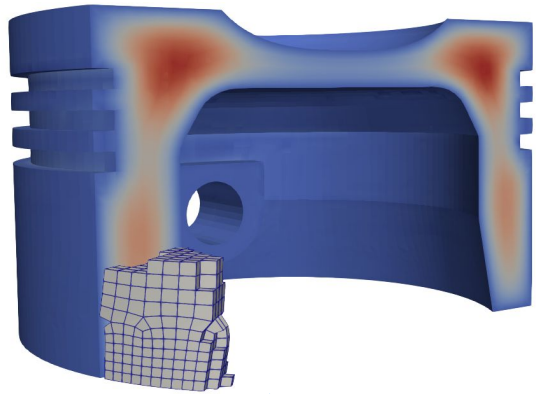


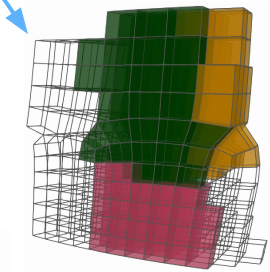
From **2.7m** original hexas → **70k** agglomerates

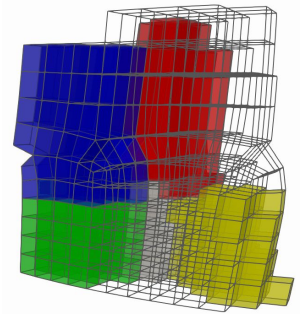


Agglomerated MultiGrid

- Basic idea: *use agglomeration to build coarser operators*
- Keep original mesh as finest level
- Use agglomerates just to create intergrid transfers $\{P_{l+1}^l\}_l$
- Start from $A_0 = \frac{M}{\Delta t} + A$
- Build coarser operators algebraically using $A_{l+1} = \left(P_{l+1}^l\right)^T A_l P_{l+1}^l$
- No need to assemble on each level
- Trivial to add in existing multigrid framework







Unstructured square

levels \ p	1	2	3	4	5
4	9	10	11	11	14
5	10	10	11	12	14
6	10	10	11	12	14

Piston

levels \ p	1	2	3	4
4	9	11	14	15
5	10	11	14	16
6	10	11	14	16

CG + MG iteration counts for SIPDG

Application to cardiac electrophysiology

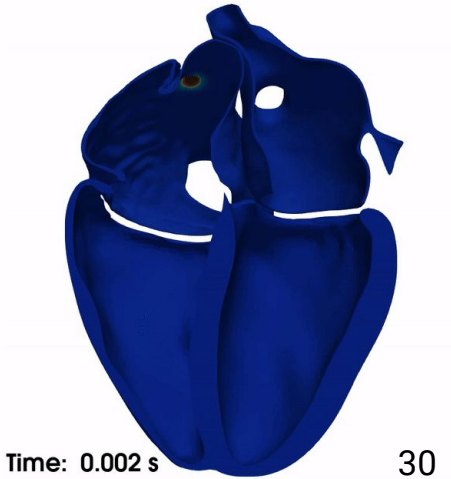
Cardiac electrophysiology in a nutshell

$$
\begin{cases}
\chi\left(C_{\mathrm{m}}\dfrac{\partial u}{\partial t}+\mathcal{I}_{\mathrm{ion}}(u,\boldsymbol{w},\boldsymbol{z})\right)-\nabla\cdot(\boldsymbol{D}_{\mathrm{M}}\nabla u)=\chi\mathcal{I}_{\mathrm{app}}(\mathbf{x},t), & \text{in } \Omega\times(0,T],\\
(\boldsymbol{D}_{\mathrm{M}}\nabla u)\cdot\mathbf{n}=0, & \text{on } \partial\Omega\times(0,T],\\
\dfrac{d\boldsymbol{w}}{dt}=\boldsymbol{H}(u,\boldsymbol{w},\boldsymbol{z}), & \text{in } \Omega\times(0,T],\\
\dfrac{d\boldsymbol{z}}{dt}=\boldsymbol{G}(u,\boldsymbol{w},\boldsymbol{z}), & \text{in } \Omega\times(0,T],\\
u(\mathbf{x},0)=u_0(\mathbf{x}),\ \boldsymbol{w}(\mathbf{x},0)=\boldsymbol{w}_0(\mathbf{x}),\ \boldsymbol{z}(\mathbf{x},0)=\boldsymbol{z}_0(\mathbf{x}), & \text{in } \Omega.
\end{cases}
$$

Motivation:

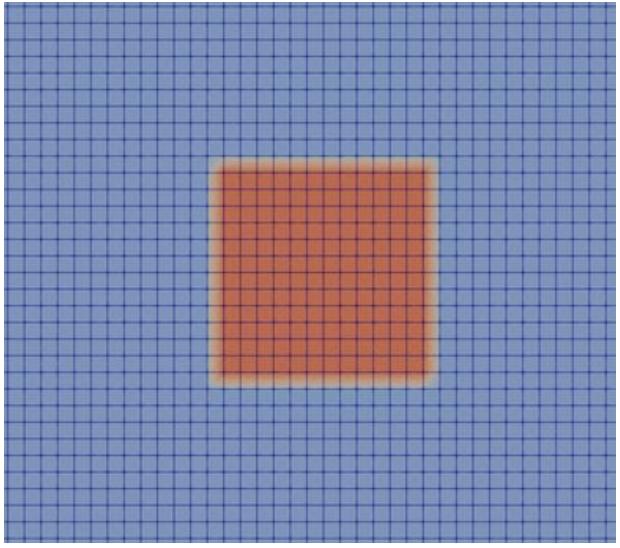
- solution characterized by a sharp and fast propagating wavefront
- extreme numerical accuracy (in space/time) required
- high−dimensional problems

→ **need for a high performance solver**



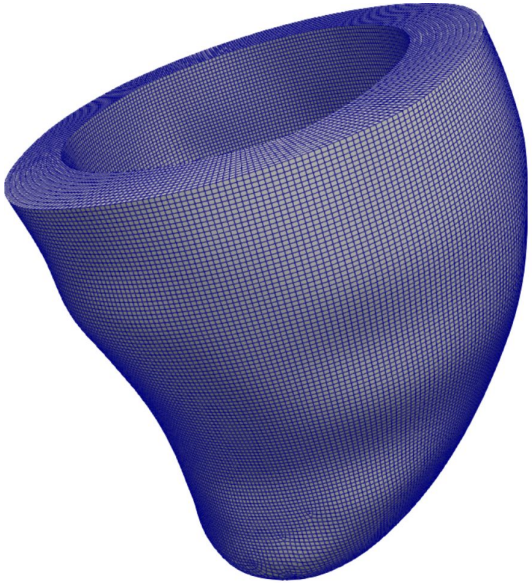
Application to cardiac electrophysiology

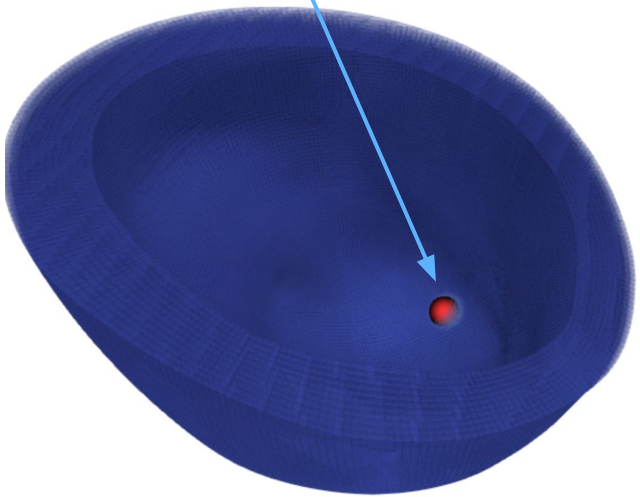
- Monodomain problem with high order DG
- Exploit polytopic/agglomerated multigrid to precondition the resulting DG system
- 4 levels with ~90, 360, 720 elements.
- Polynomial order 3 → ~8 iterations/time step (≪ AMG).

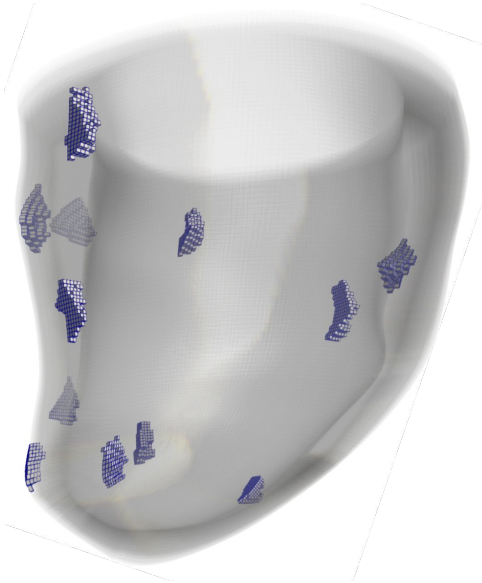


Application to cardiac electrophysiology

- Realistic mesh of left ventricle (370k hexas)
- View of some of the agglomerates
- Apply impulse at specific points







Application to cardiac electrophysiology

- Semi-implicit discretization yields

$$\left(\frac{M}{\Delta t} + A\right)\boldsymbol{U}_{n+1} = MI^h_{app,n+1} - MI^h_{ion}(\boldsymbol{U}_n, \boldsymbol{W}_{n+1}) + \frac{M}{\Delta t}\boldsymbol{U}_n$$

- Preconditioned conjugate-gradient with agglomerated multigrid
- Only need the operator evaluation on the fine level

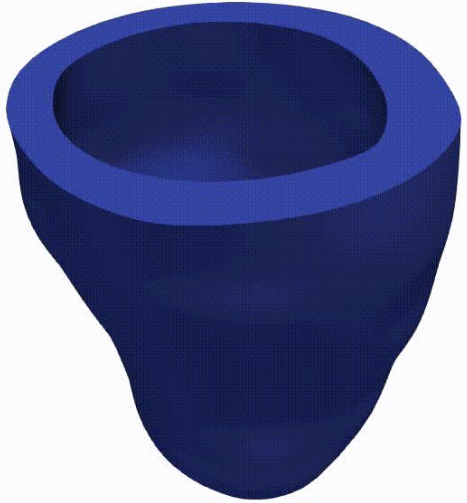
- Ratio close to 8
- Well balanced levels

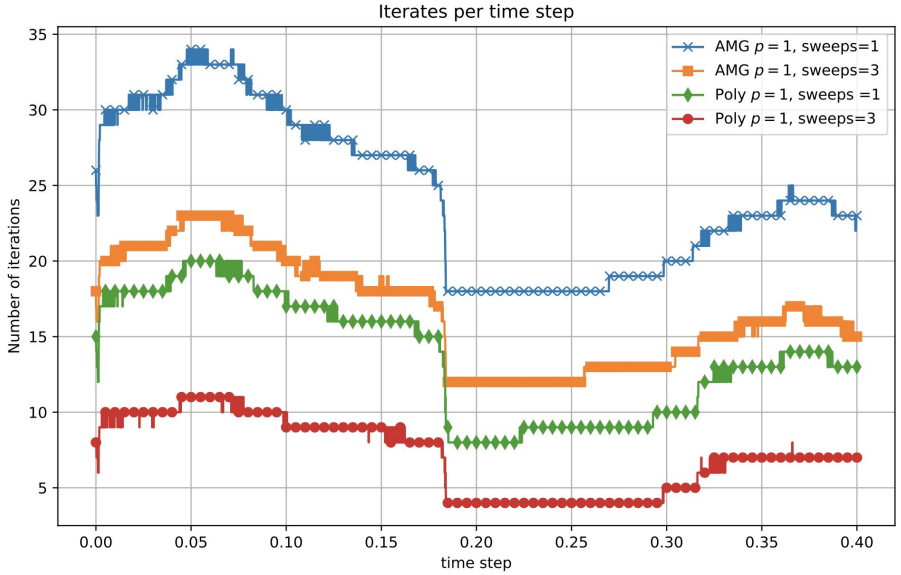
Hierarchy for ventricle mesh (**128** processors)		
Level l	Card($\mathcal{T}_l$)	Card($\mathcal{T}_{l-1}$)/Card($\mathcal{T}_l$)
$l = 0$	374 022	–
$l = 1$	46 809	~ 7.9
$l = 2$	5 908	~ 7.9
$l = 3$	768	~ 7.7

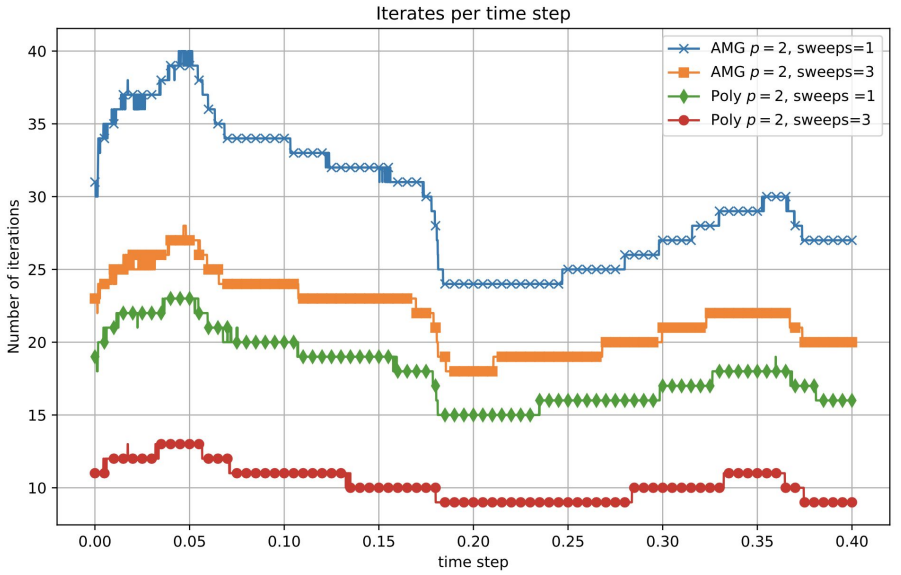
Hierarchy for ventricle mesh (**256** processors)		
Level l	Card($\mathcal{T}_l$)	Card($\mathcal{T}_{l-1}$)/Card($\mathcal{T}_l$)
$l = 0$	374 022	–
$l = 1$	46 871	~ 7.9
$l = 2$	5 987	~ 7.8
$l = 3$	768	~ 7.8

Iteration counts

- Solve until $T = 0.4s$, with $\Delta t = 10^{-4}s$
- Lower iteration counts compared to CG+AMG
- Robust with respect to number of levels and MPI ranks







Outlook and future work

- ✓ Distributed implementation of polyDG and agglomerated MG
- ✓ Successfully applied to large geometries
- ✓ Application to cardiac electrophysiology
- ⏲ Matrix-free operator evaluation

References

[1] A. Guttmann - R-Trees: a Dynamic Index Structure for Spatial Searching (1984).

[2] P. F. Antonietti, M. Sarti, M. Verani - Multigrid algorithms for hp-version interior penalty discontinuous Galerkin methods on polygonal and polyhedral meshes (2015).

[3] A. Cangiani, Z. Dong, E. H. Georgoulis - hp-Version discontinuous Galerkin methods on essentially arbitrarily-shaped elements (2022).

[4] PCA, M. Salvador, P. Gervasio, L. Dede', A. Quarteroni - A matrix-free high-order solver for the numerical solution of cardiac electrophysiology (2023).

[5] M. Feder, A. Cangiani, L. Heltai - R3MG: R-tree based agglomeration of polytopal grids with applications to multilevel methods (2024).

Research funded by:

- NextGenerationEU (project ID ECS_00000043)

Thanks!

Pasquale Claudio Africa
pafrica@sissa.it

M. Feder, A. Cangiani, L. Heltai

SISSA