# HYBRID TREE-BASED ADAPTIVE MESH REFINEMENT

A technical overview and use cases

Sandro Elsweijer, DLR-SC & INS Uni Bonn, 07.03.2024

# Agenda

- Introduction
- Adaptive Mesh Refinement
- Our implementation of AMR
- Features
- Application scenarios
- Future plans

# Who am I? And why am I here?

- M. Eng. in mechanical engineering at UAS Bonn-Rhein-Sieg
- Ongoing PhD in math at University of Bonn
  - Partition of Unity Methods for Additive Manufacturing
- Research associate at the German Aerospace Center
  - Tree-based Adaptive Mesh Refinement
- Zoltan Csati approached us via GitHub
  - Collaborating on some features in our software

Sandro Elsweijer

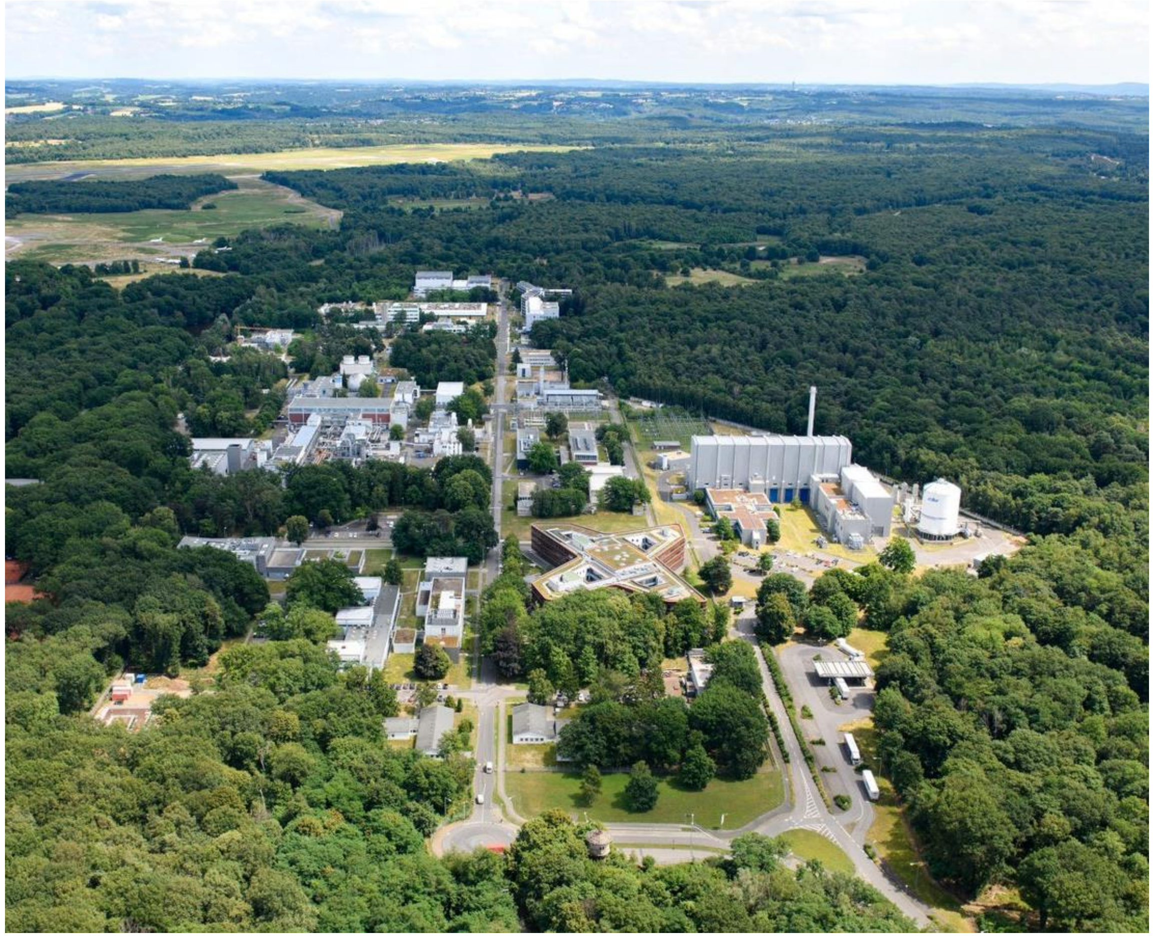
# German Aerospace Center (DLR)

- Aeronautics and space research center for Germany with focus on
  - Aeronautics
  - Space
  - Energy
  - Transport
  - Security
  - Digitalization
- Includes the German space agency
- National and international cooperation

# DLR Institute for Software Technology

Software Institute concerned with

- Distributed and intelligent systems
- Artificial Intelligence
- Visualization
- High-Performance Computing
- Quantum Computing

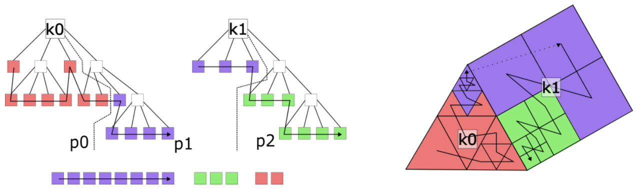


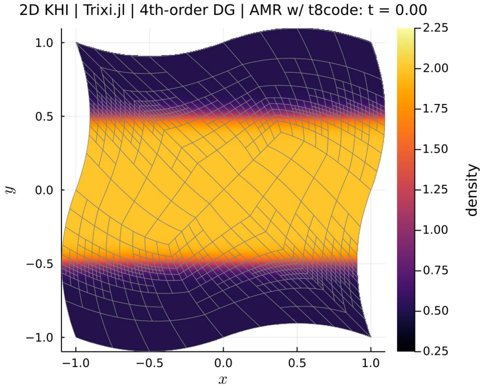
# DLR Institute for Software Technology

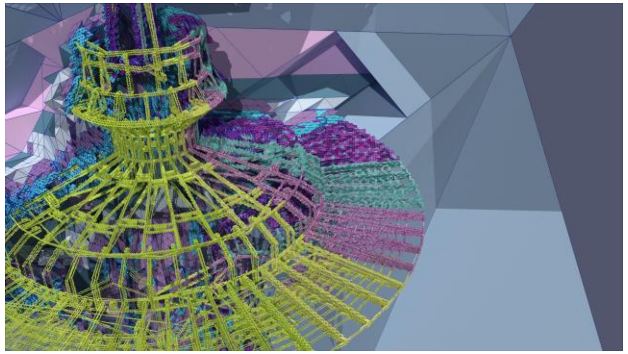
Software Institute concerned with

- Distributed and intelligent systems
- Artificial Intelligence
- Visualization
- High-Performance Computing
- Quantum Computing

# Scalable Adaptive Mesh Refinement Group



**Adaptive mesh refinement (AMR)**

- Software development
- Algorithms and data structures for scalable AMR

**Numerical Simulation with AMR**

- Discontinuous Galerkin
- Partition of Unity Methods
- CFD
- Structural mechanics

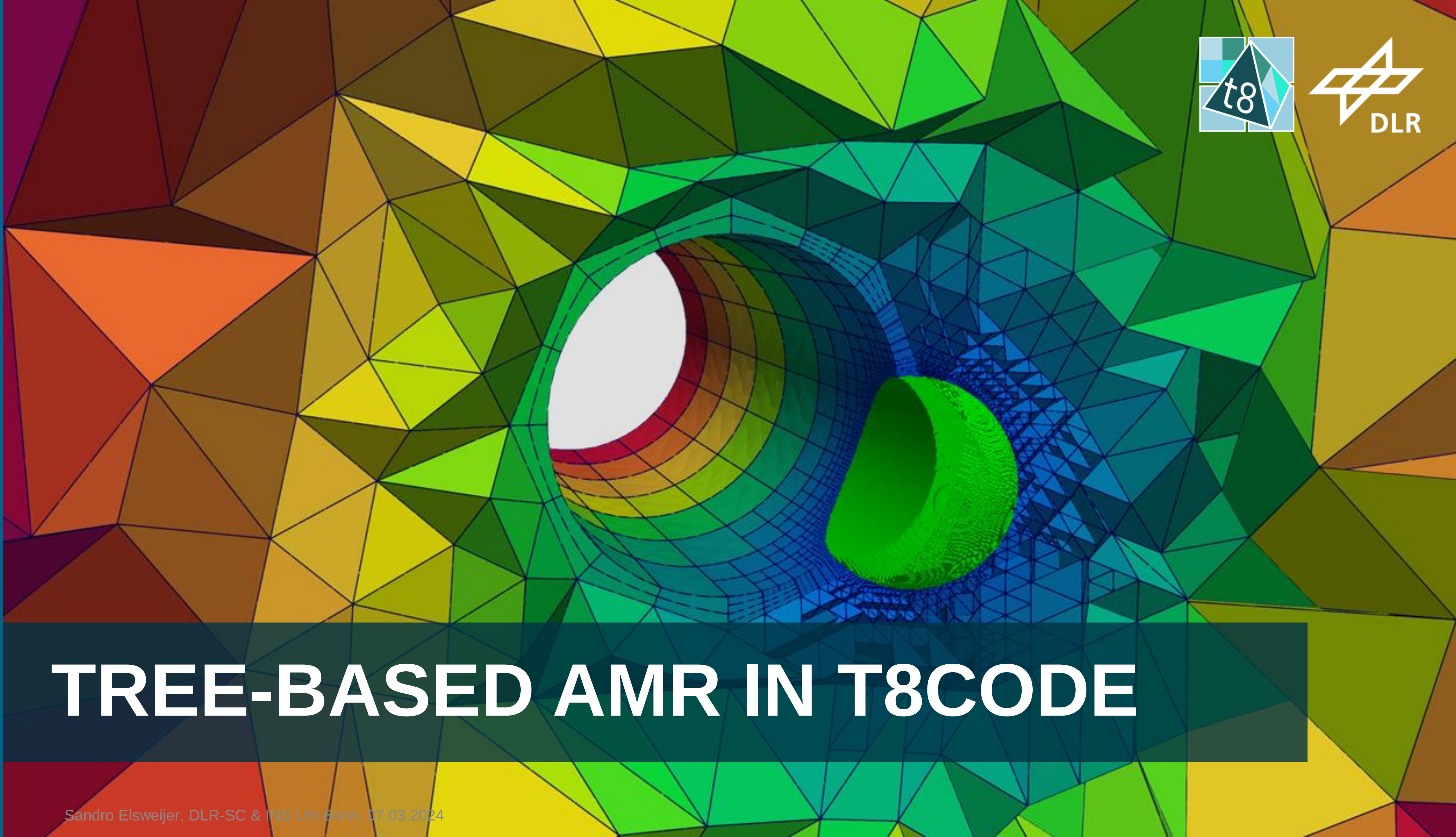
**Postprocessing with AMR**

- Processing of large scientific data-sets
- Lossy data compression
- Visualization

# Goal of this presentation

- Present our research topics
- Find common interests

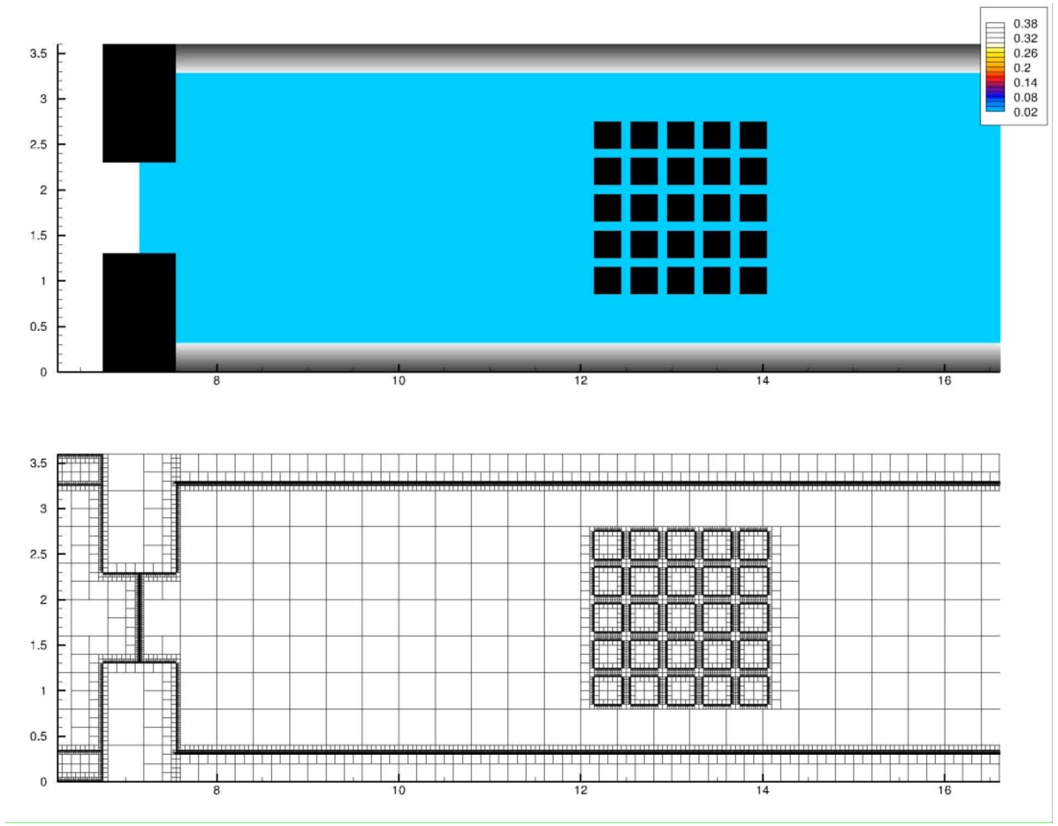
This will not be an in-detail presentation about the inner workings of AMR



# TREE-BASED AMR IN T8CODE

# Adaptive Mesh Refinement (AMR)

- Simulation of PDEs
- Resolution as fine as needed
- And as coarse as possible
- Optimal ratio between cost and accuracy




Caviedes-Voullieme, Gerhard, Sikstel, Müller

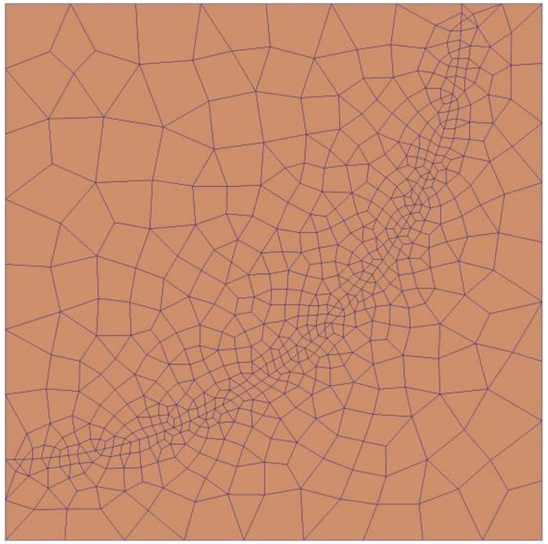
# Adaptive Mesh Refinement (AMR)

## Unstructured AMR

– Memory intensive

– Graph partitioning required

+ Supports complex domains



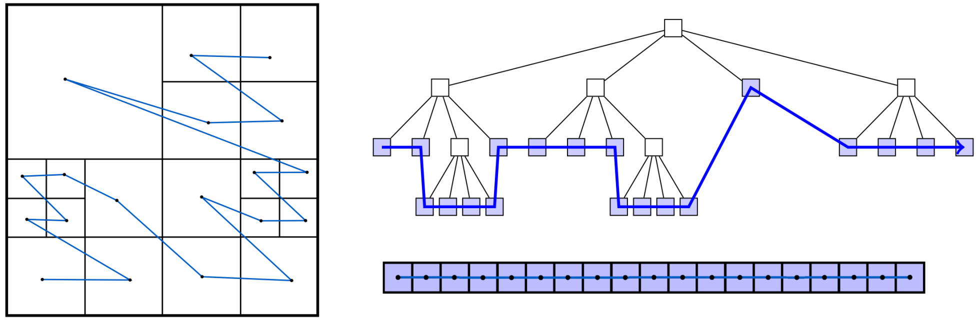
## Tree-based AMR

+ Structured, hence memory efficient

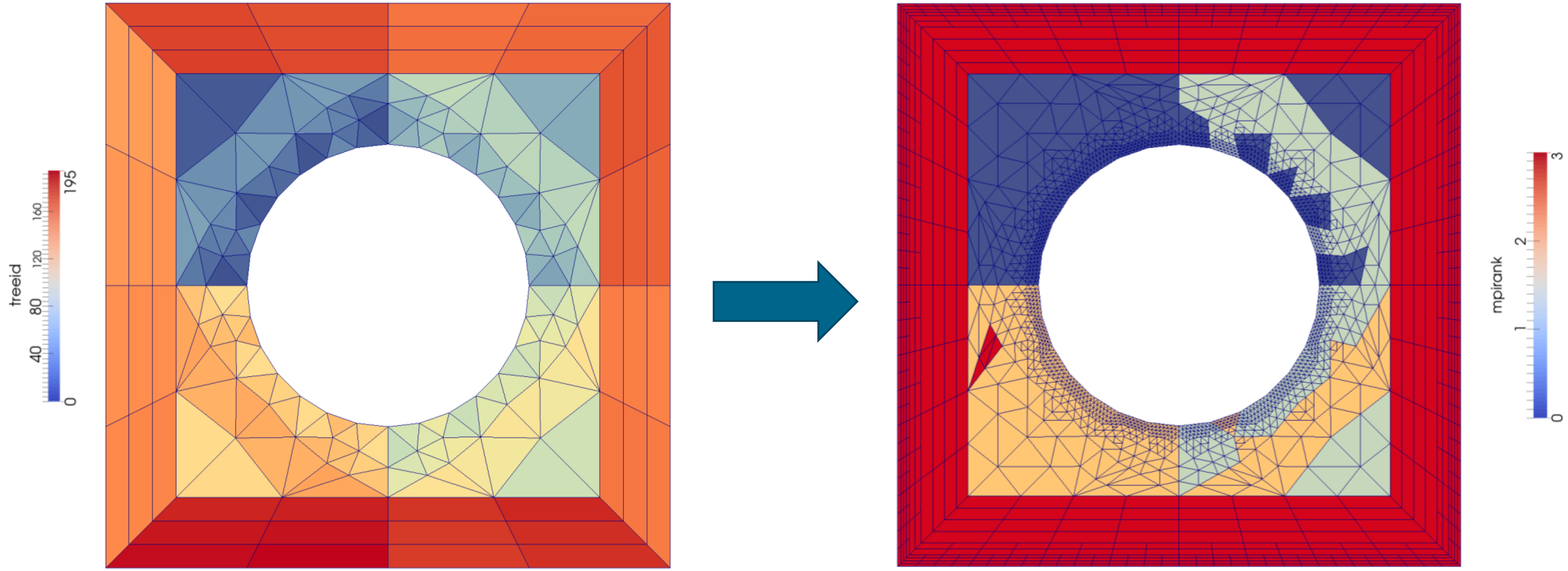
+ Partitioning inherently easy

+ Good scaling

+ Hierarchy



# Adaptive Mesh Refinement (AMR)

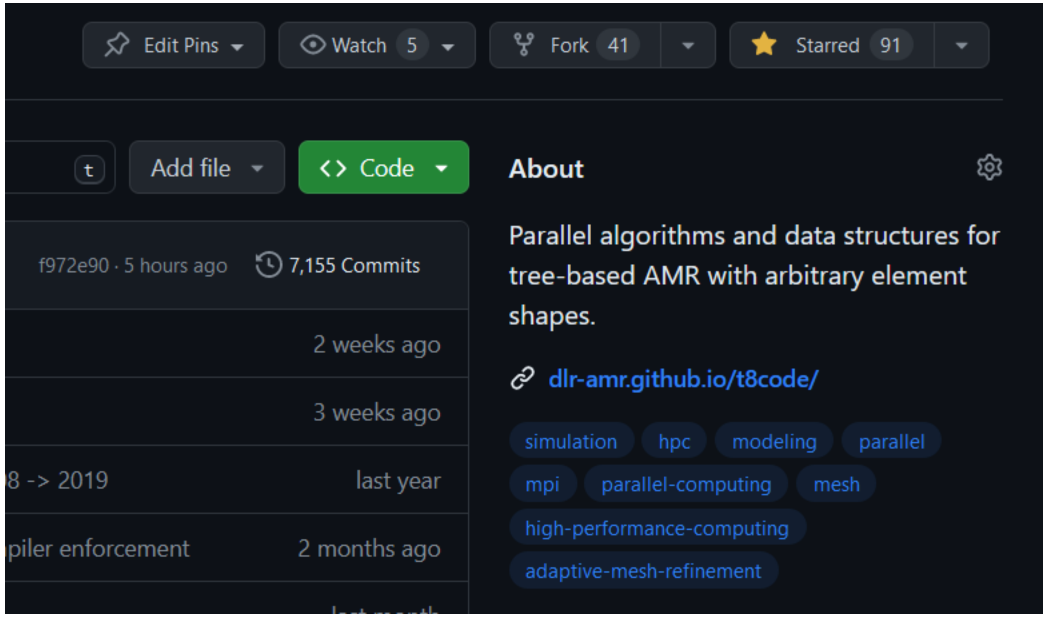



**Forest-of-trees approach**

+ Structured, hence memory efficient
+ Partitioning inherently easy
+ Good scaling
+ Hierarchy
+ Supports complex domains

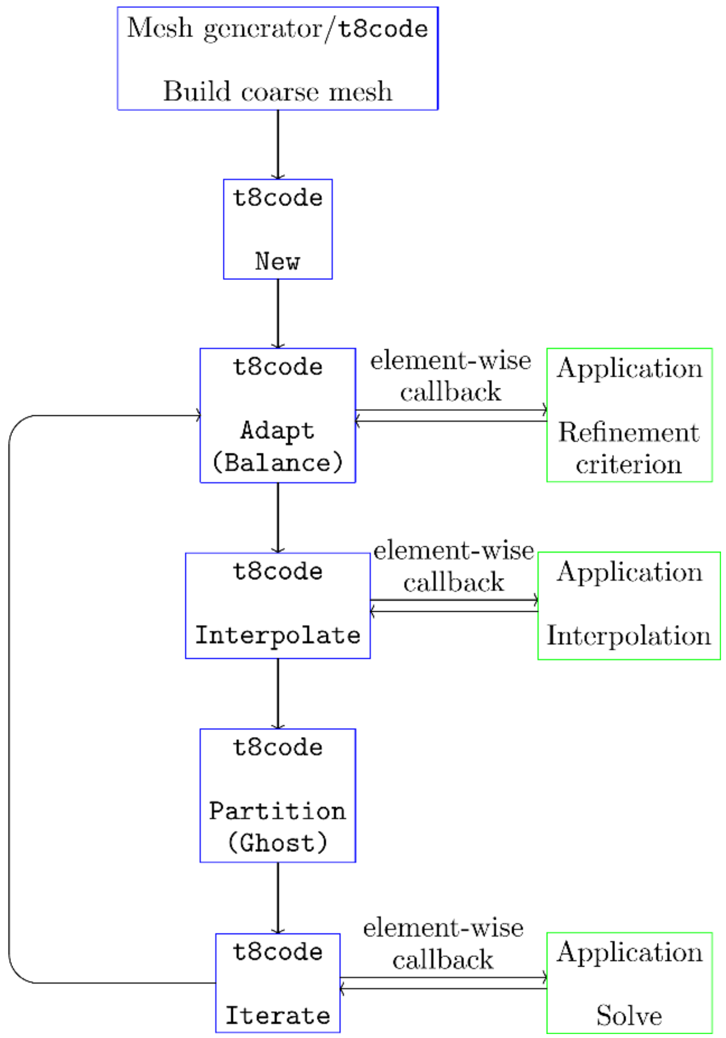
# t8code

- Parallel management of adaptive meshes and data
- Open Source on GitHub
- C, C++ and Julia API
- Fortran API under development



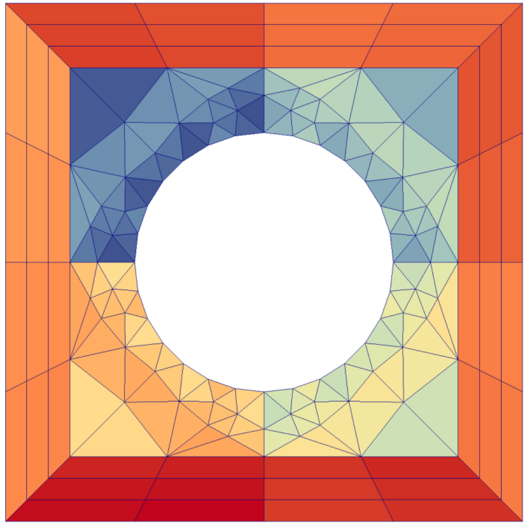
# Core algorithms

## Typical simulation workflow



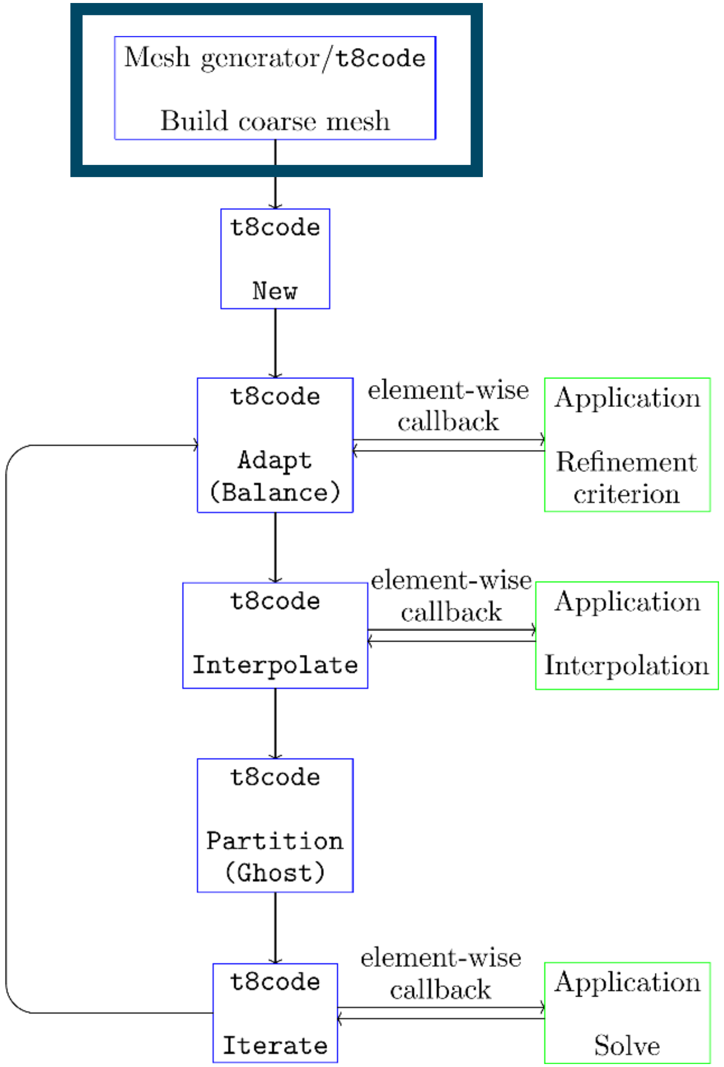
# Core algorithms

## Mesh generation



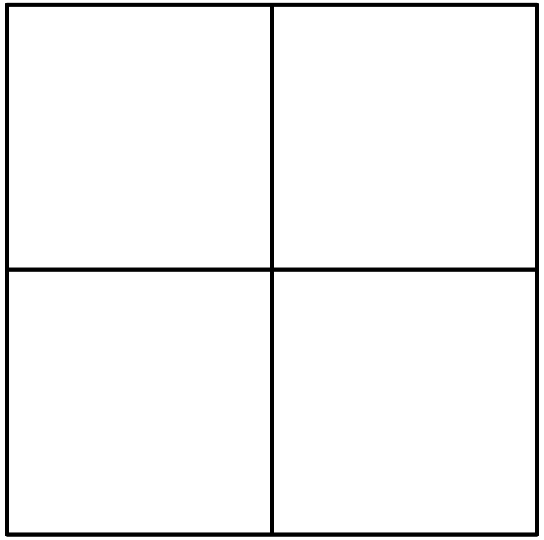
Generate the coarse input mesh

- Use external mesh generator
- Build a basic domain by hand

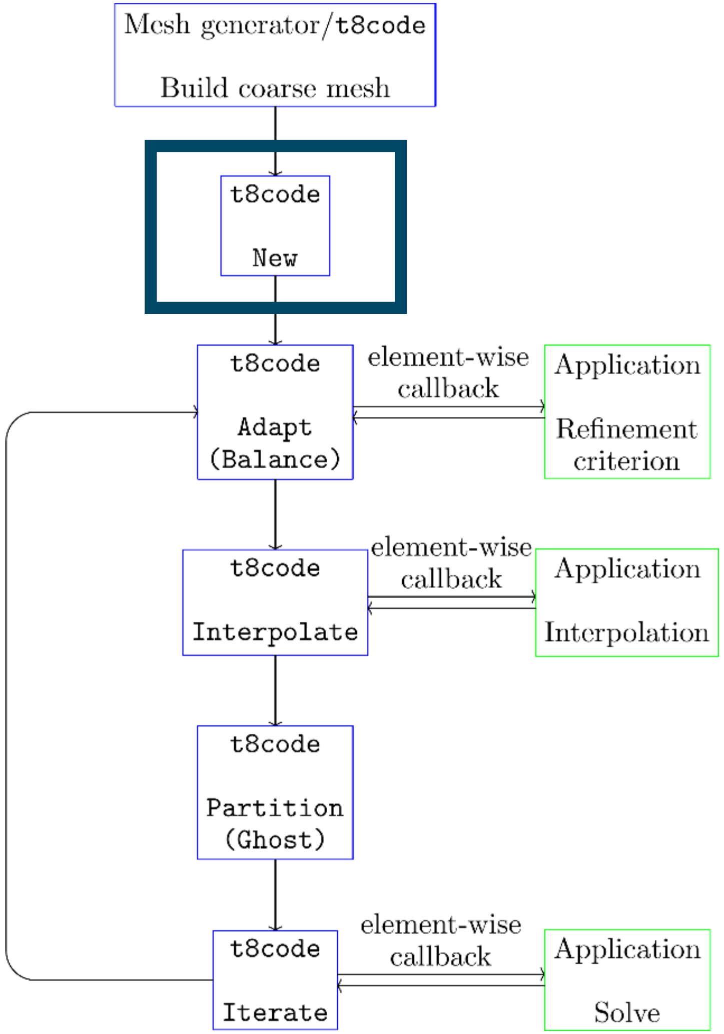


# Core algorithms

## New



- Creates a forest for the given input mesh
- Refines to a user specified uniform level

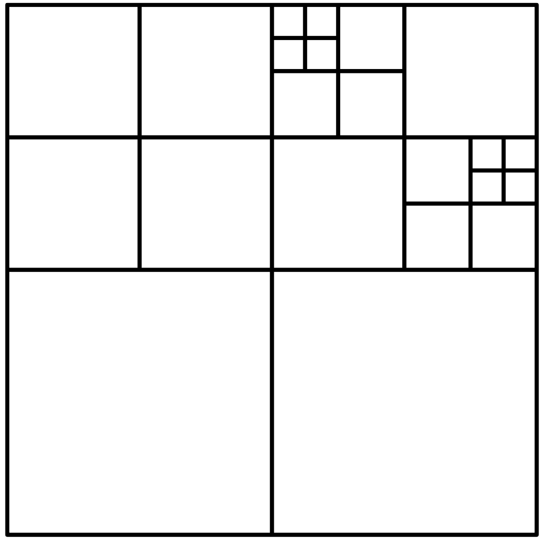




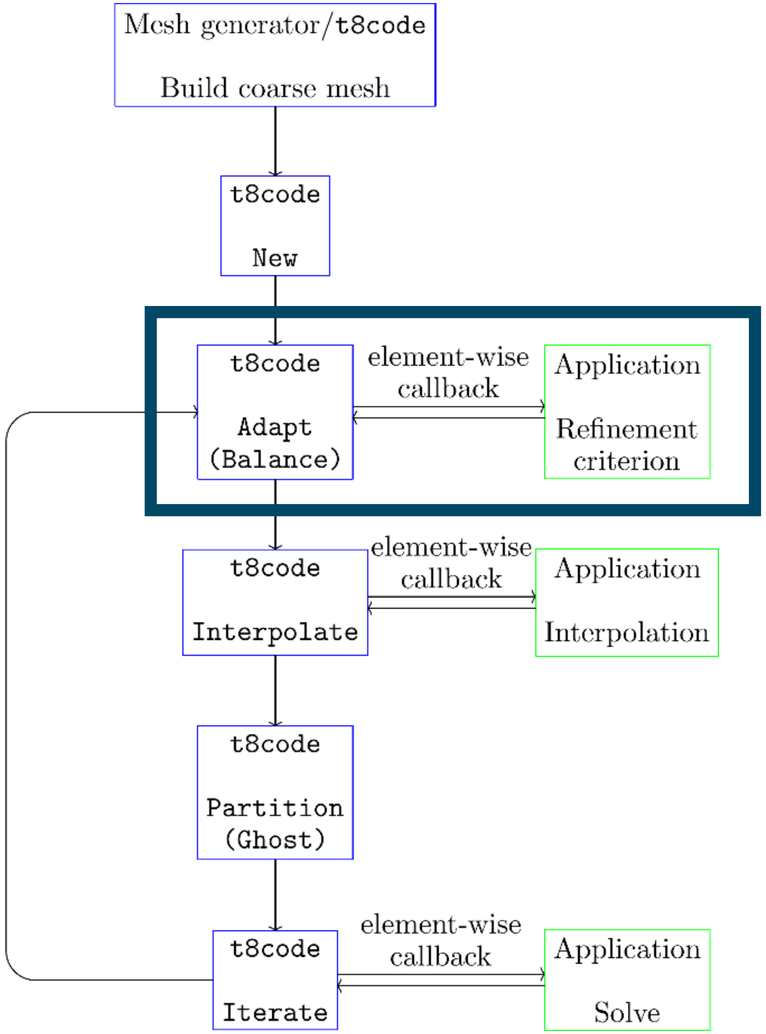

Sandro Elsweijer, DLR-SC & INS Uni Bonn, 07.03.2024

# Core algorithms

## Adapt

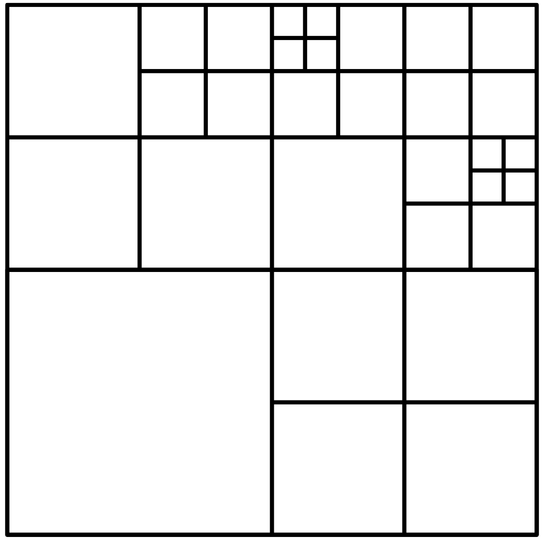


- Refines and coarsens the mesh
- Refinement criterion provided by the user

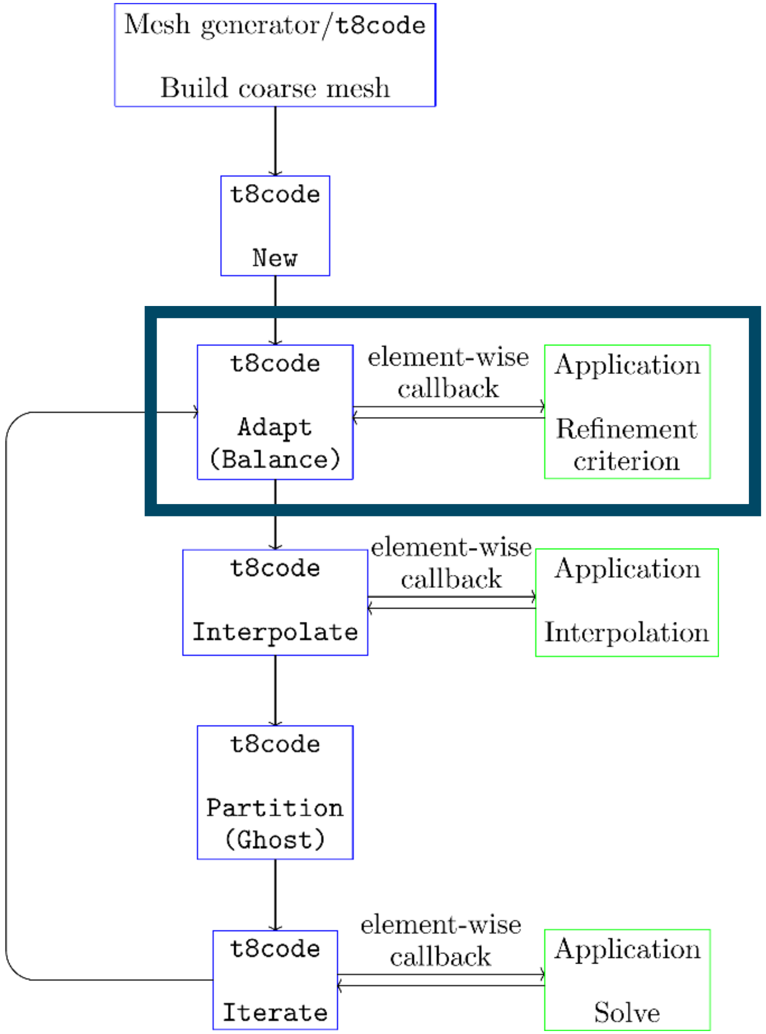


# Core algorithms

## Balance



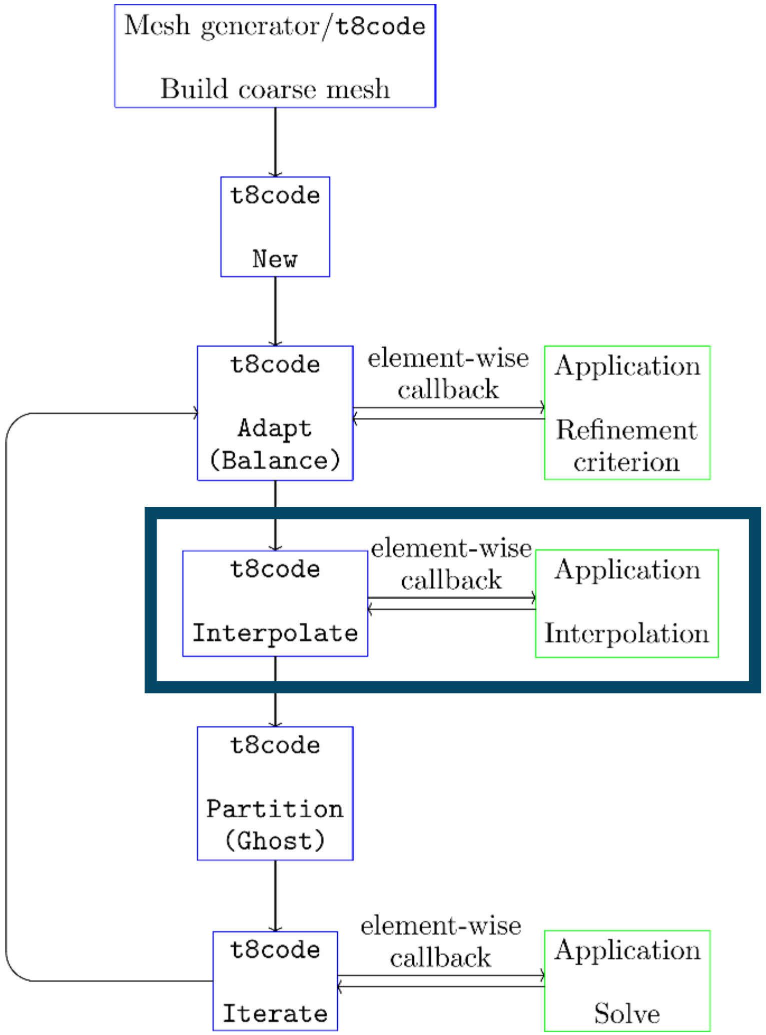
- Establishes a 2:1 balancing
- Enables mortar method for resolving of hanging nodes



# Core algorithms

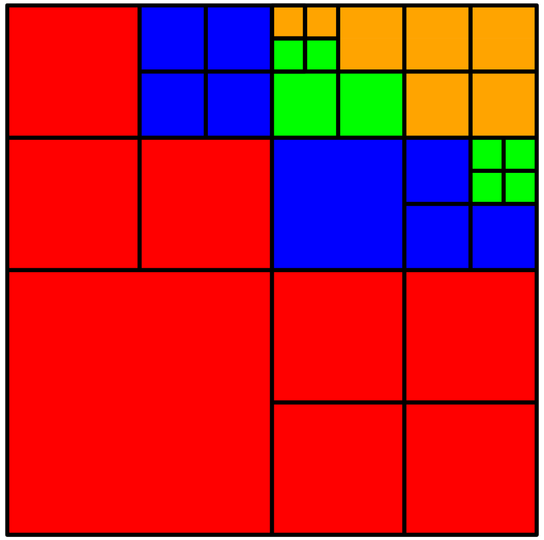
## Interpolate

- Interpolates the user data
- From old to new mesh

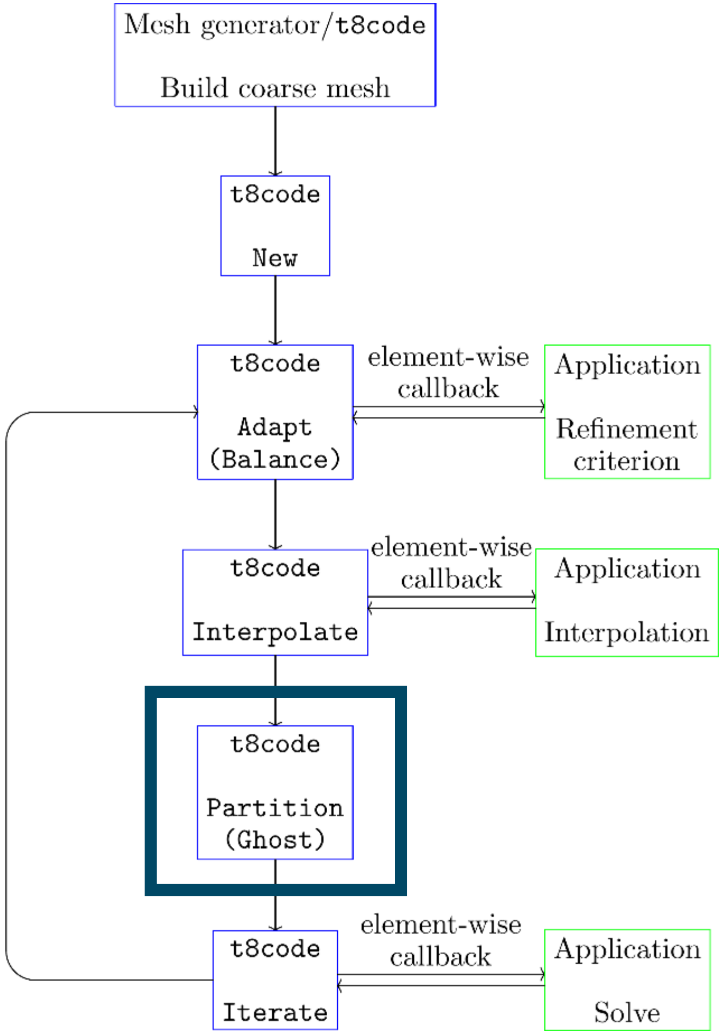


# Core algorithms

## Partition

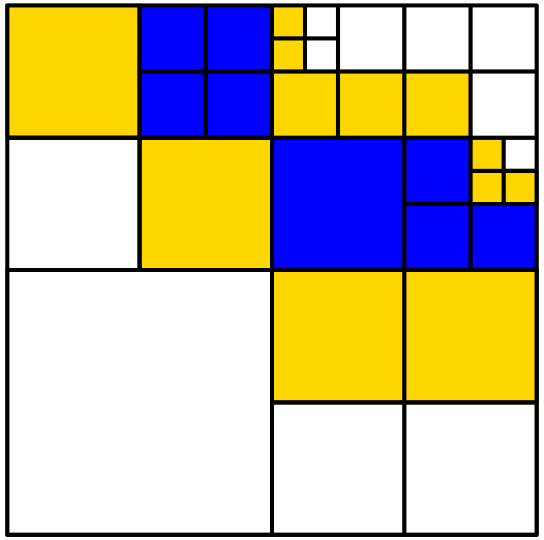


- Partitions mesh and data across multiple processes

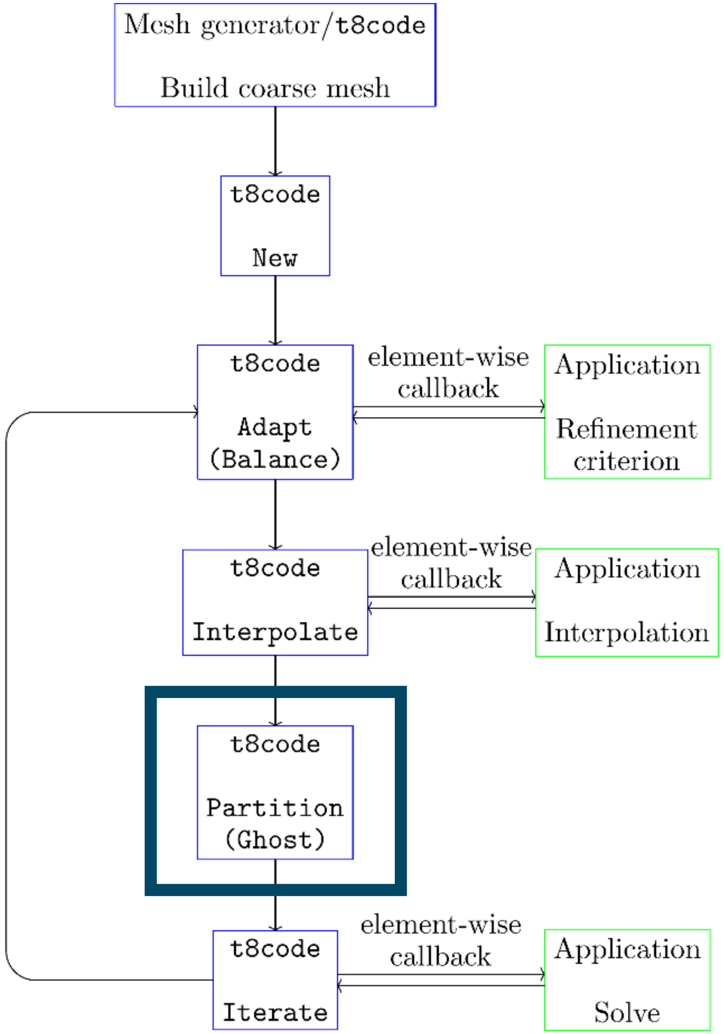


# Core algorithms

## Ghost



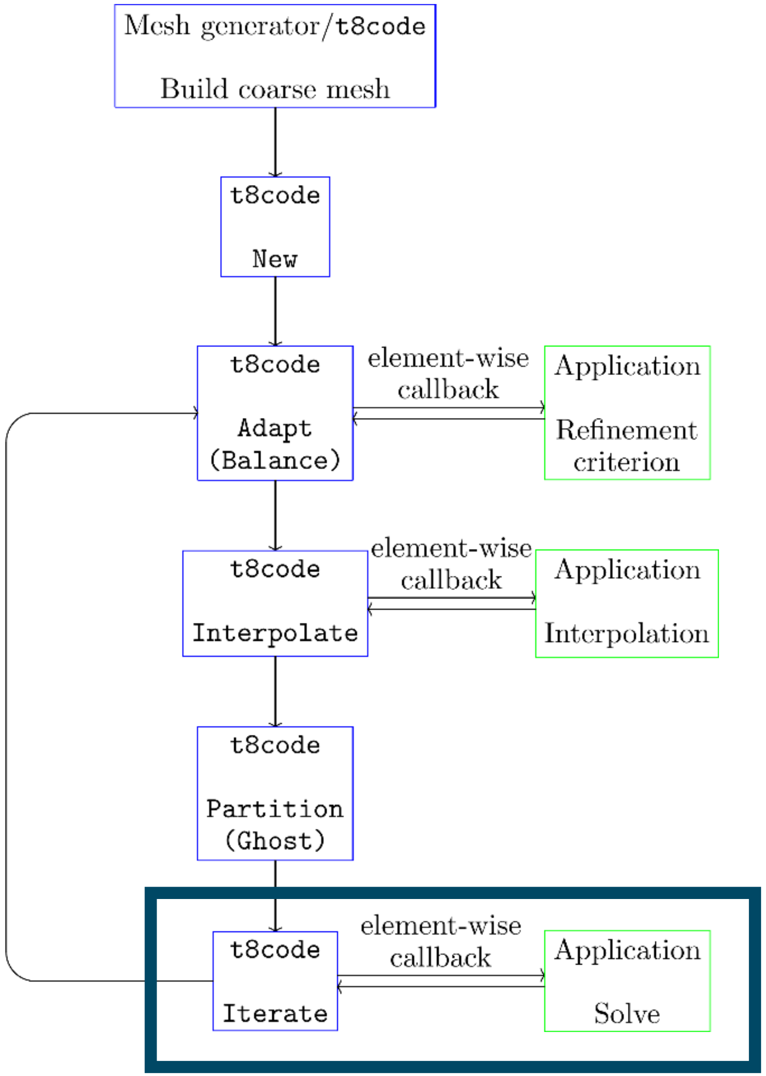
- Makes neighbor data available across processes



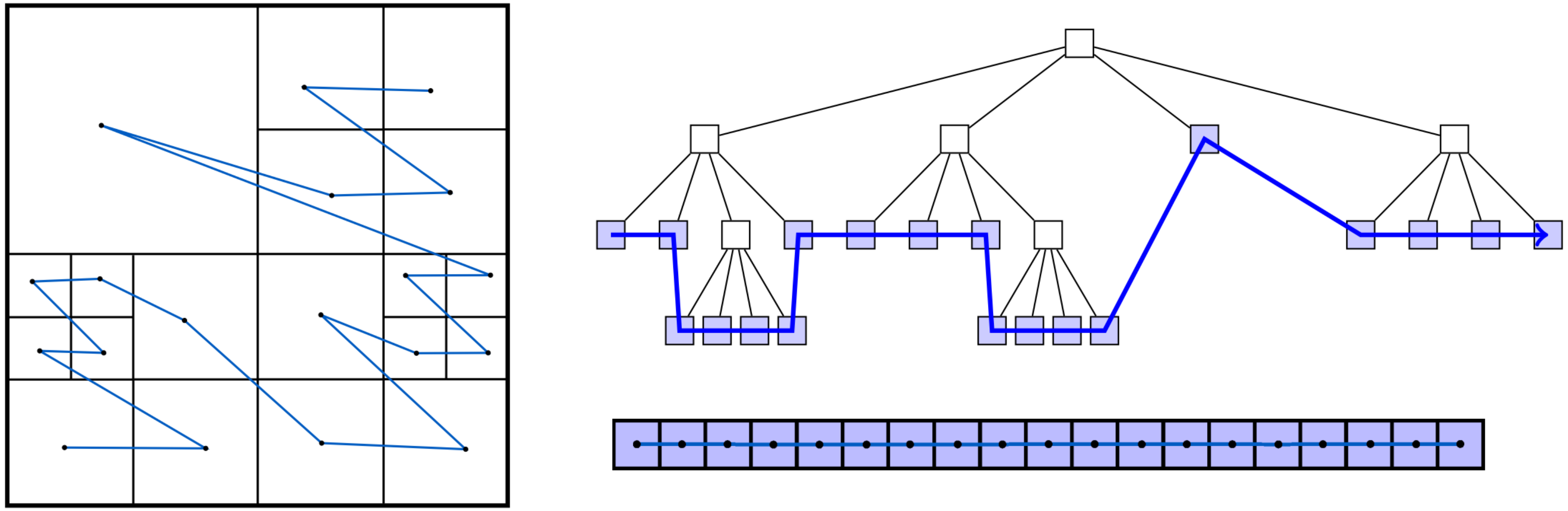
# Core algorithms

## Iterate

- Iterates over the mesh
- Makes data available to the solver

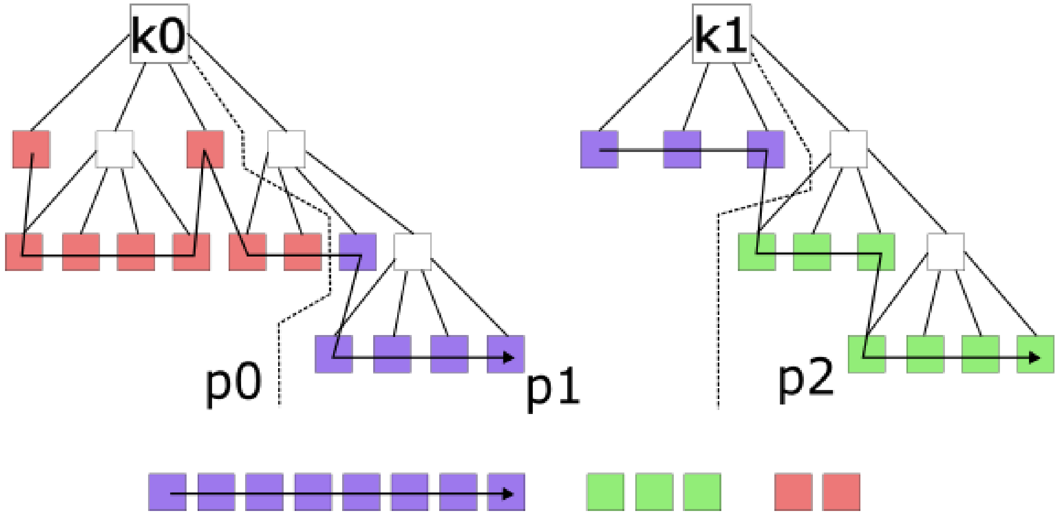


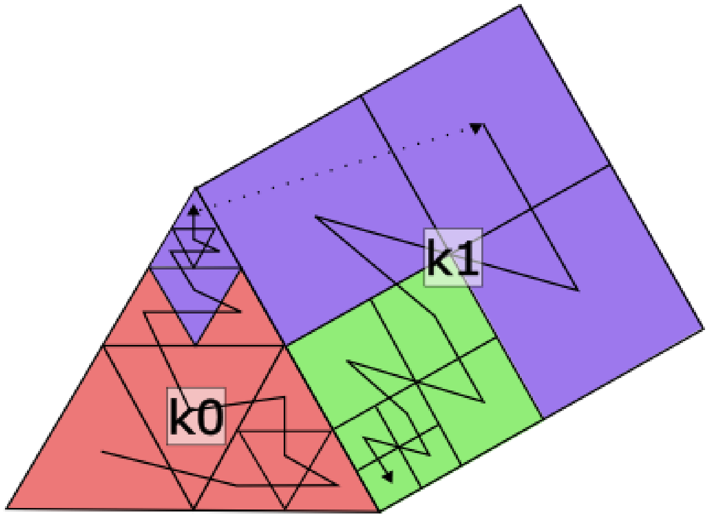
# Tree-based AMR



- Mesh as a tree-like structure
- Space-filling curve (SFC) orders elements
- Partitioning by splitting the curve
- Elements derived from their parents

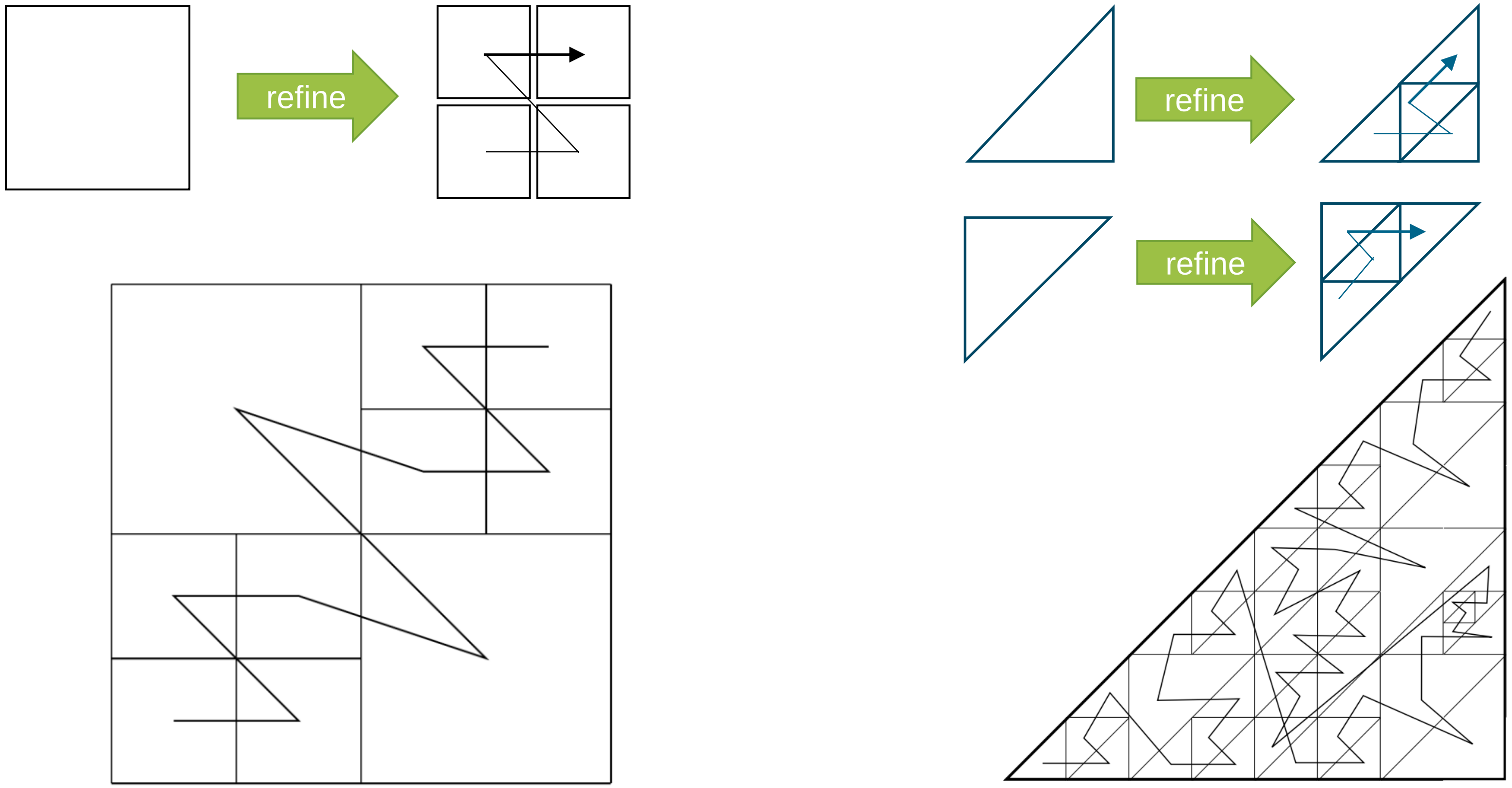
# From a tree to a forest



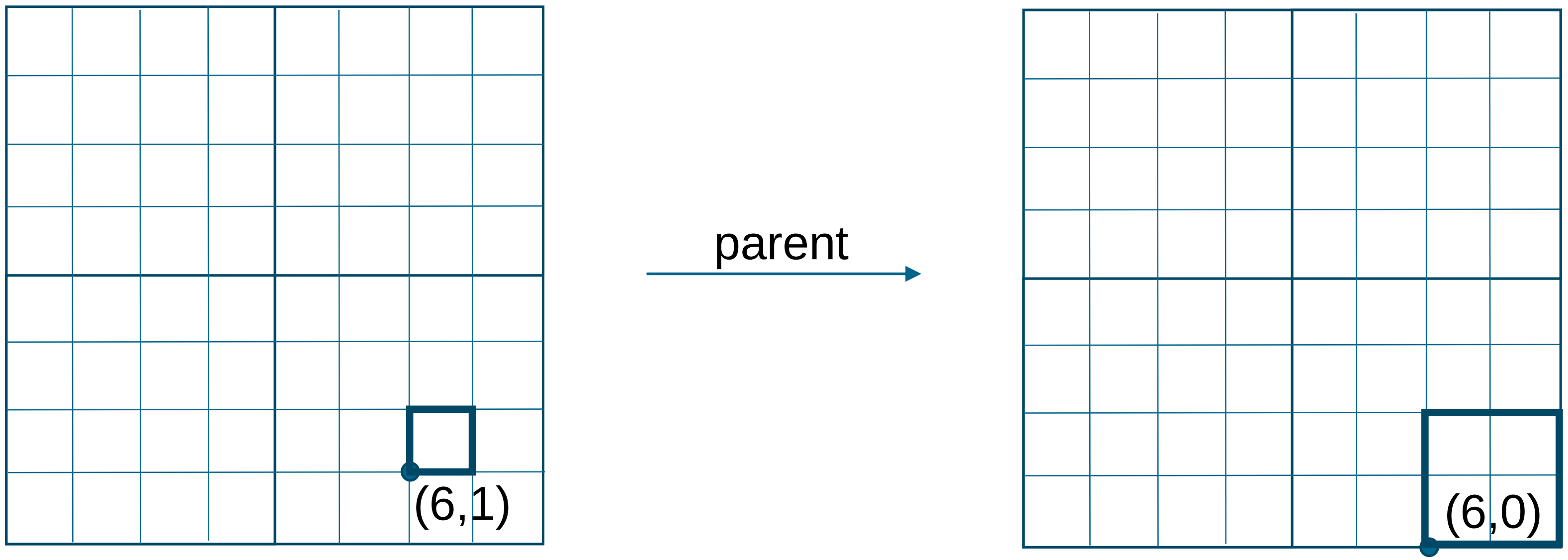

- Multiple trees form a forest
- Partitioning still easy
- Element shape still derived from parents

# Space-Filling Curves (SFC) via recursive refining




- SFC for different element shapes
  - 0D: Vertex
  - 1D: Line
  - 2D: Quadrilateral and triangle
  - 3D: Hexahedron, tetrahedron, prism and pyramid

# SFCs via bit operations

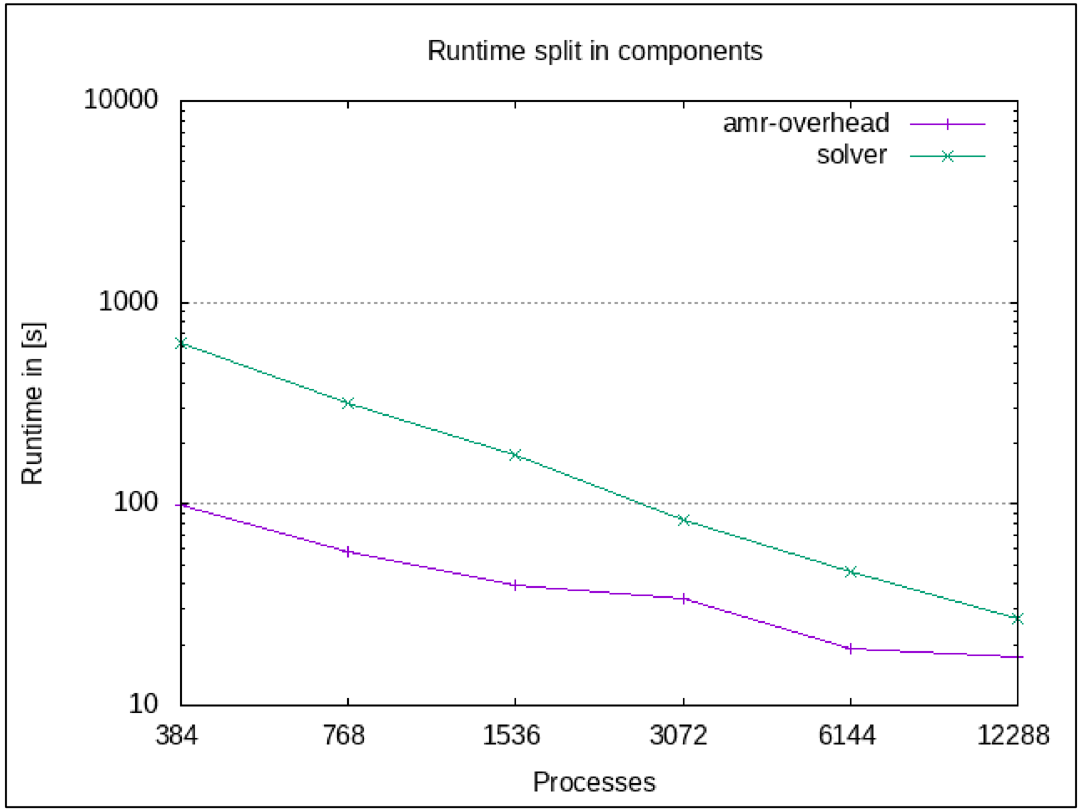



$x \quad = (110|0..0)_2$
$y \quad = (001|0..0)_2$

$x \quad = (11|00..0)_2$
$y \quad = (00|00..0)_2$

- Compute coordinates of parents, ancestors, siblings, descendants, face-neighbors, etc. by bit operation
- Super fast on CPUs

# AMR performance




Only 10 – 15% AMR overhead
But: Greatly dependent on how often the mesh is adapted

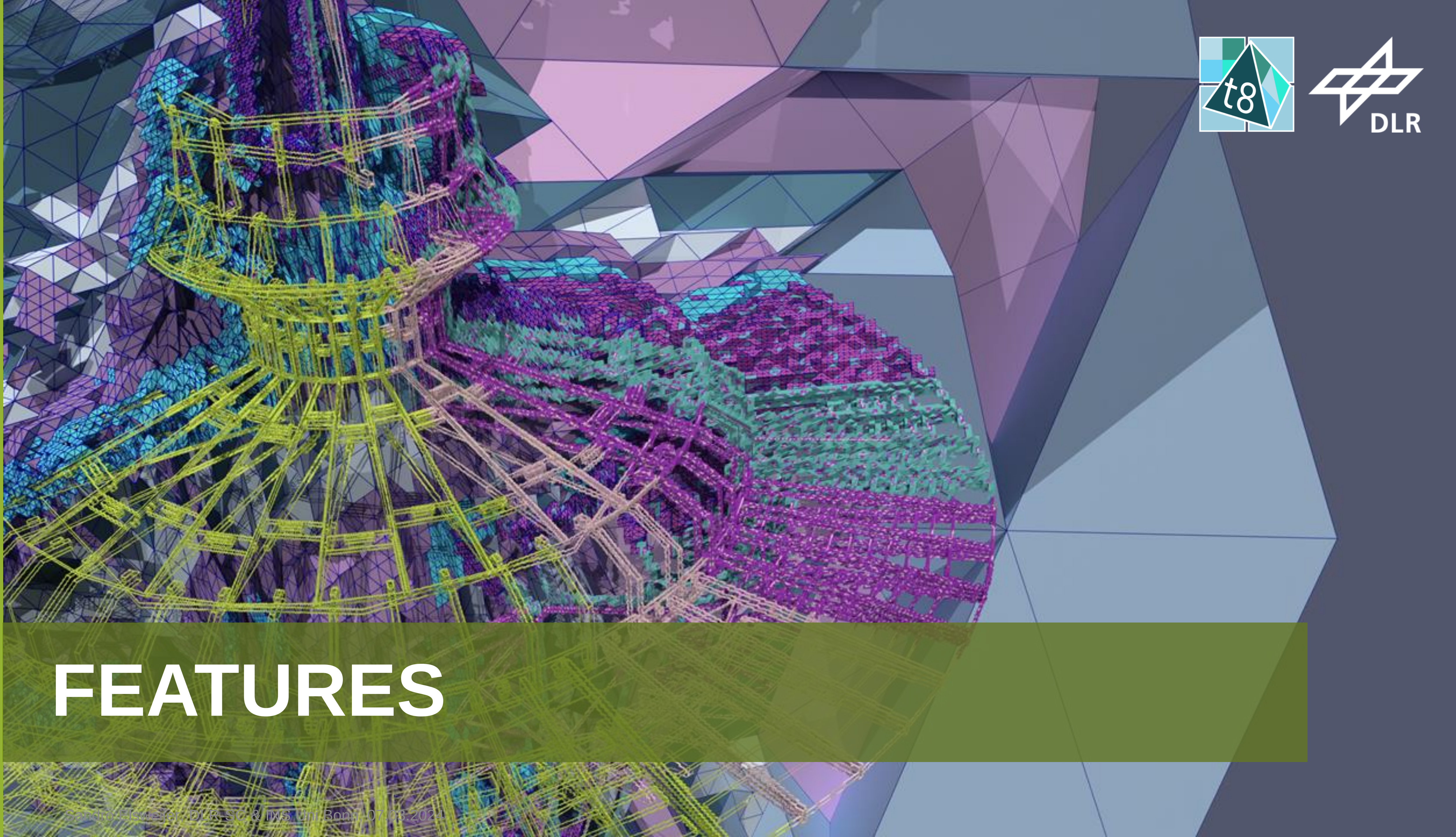
**uniform vs. adaptive**

| Advection-Diffusion DG Simulation | Runtime | Error | #DOFs |
|---|---|---|---|
| Uniform 3D | 7057s | 1.3e-3 | 16.777.216 |
| Adaptive 3D | 561s | 1.5e-3 | ~1.920.000 |

Factor 12 runtime improvement with about the same error.

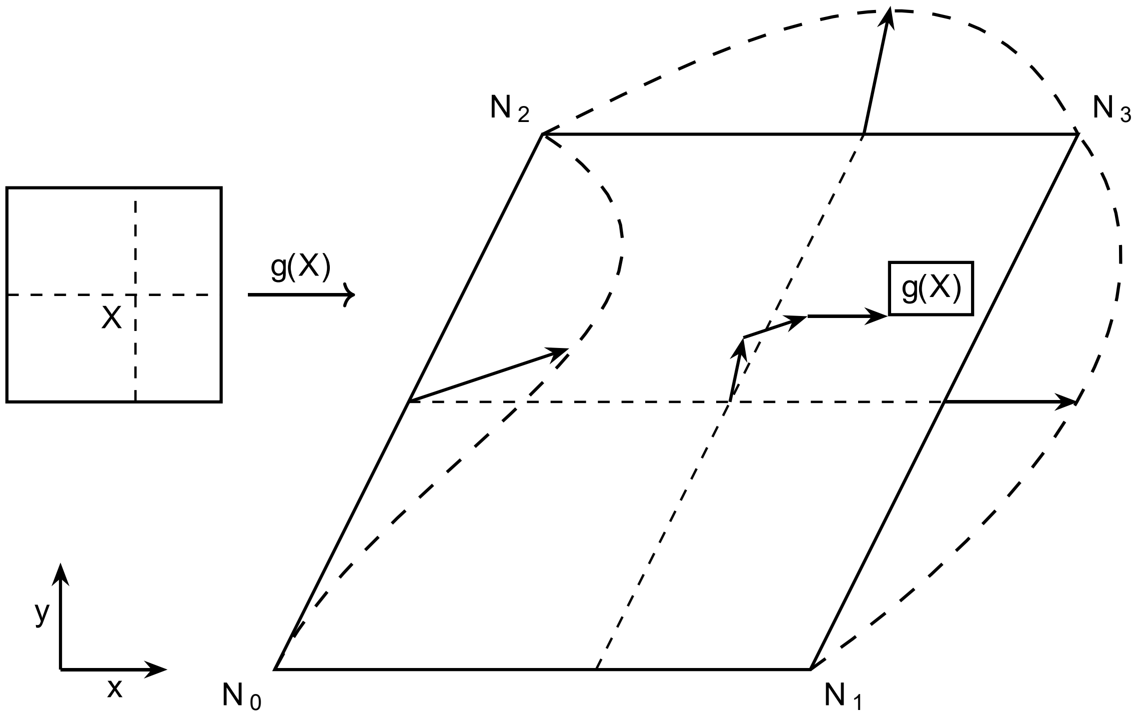
**1 trillion element mesh**

| #processes | #elements | #elements/ process | ghost | partition |
|---|---|---|---|---|
| 98,304 | ~ 1.1e12 | 11,184,811 | 1.43 s | 0.33 s |



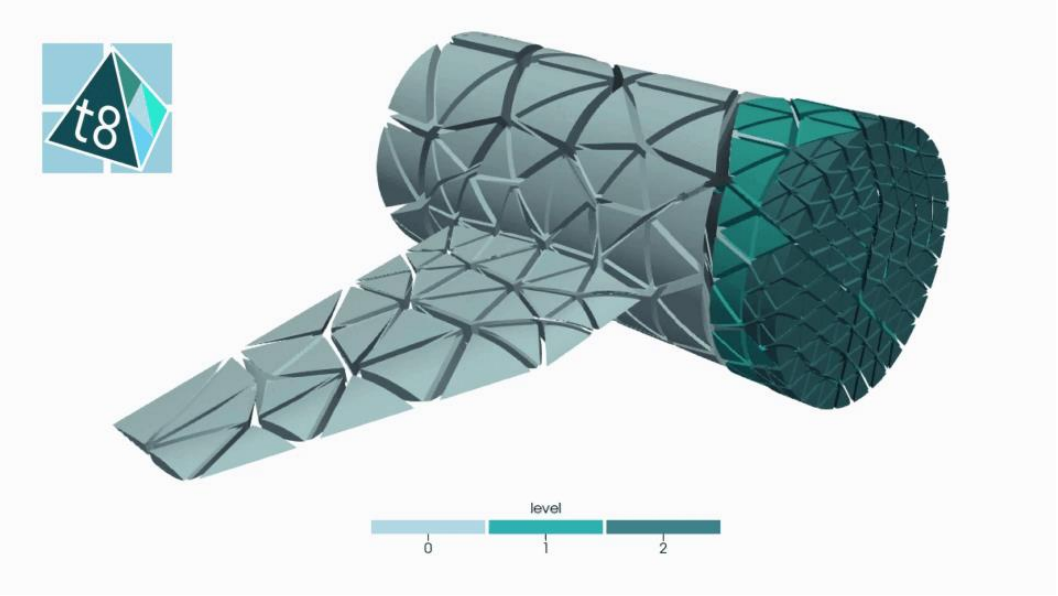
# FEATURES

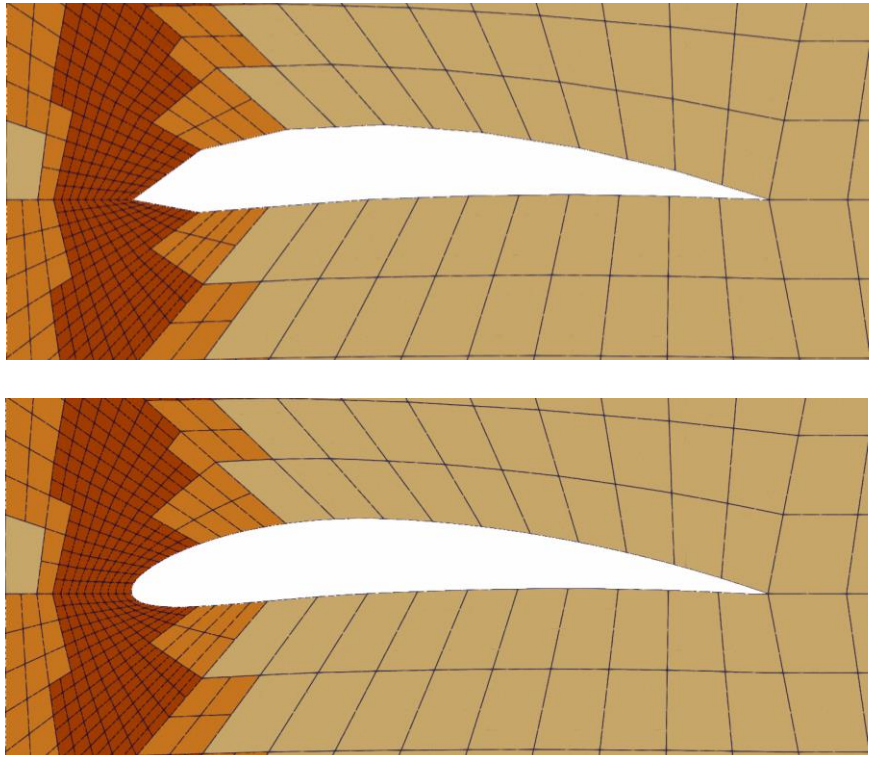
# Curved meshes (CAD enhanced)




- Recombines mesh and CAD shape
- Enables geometry-based mapping

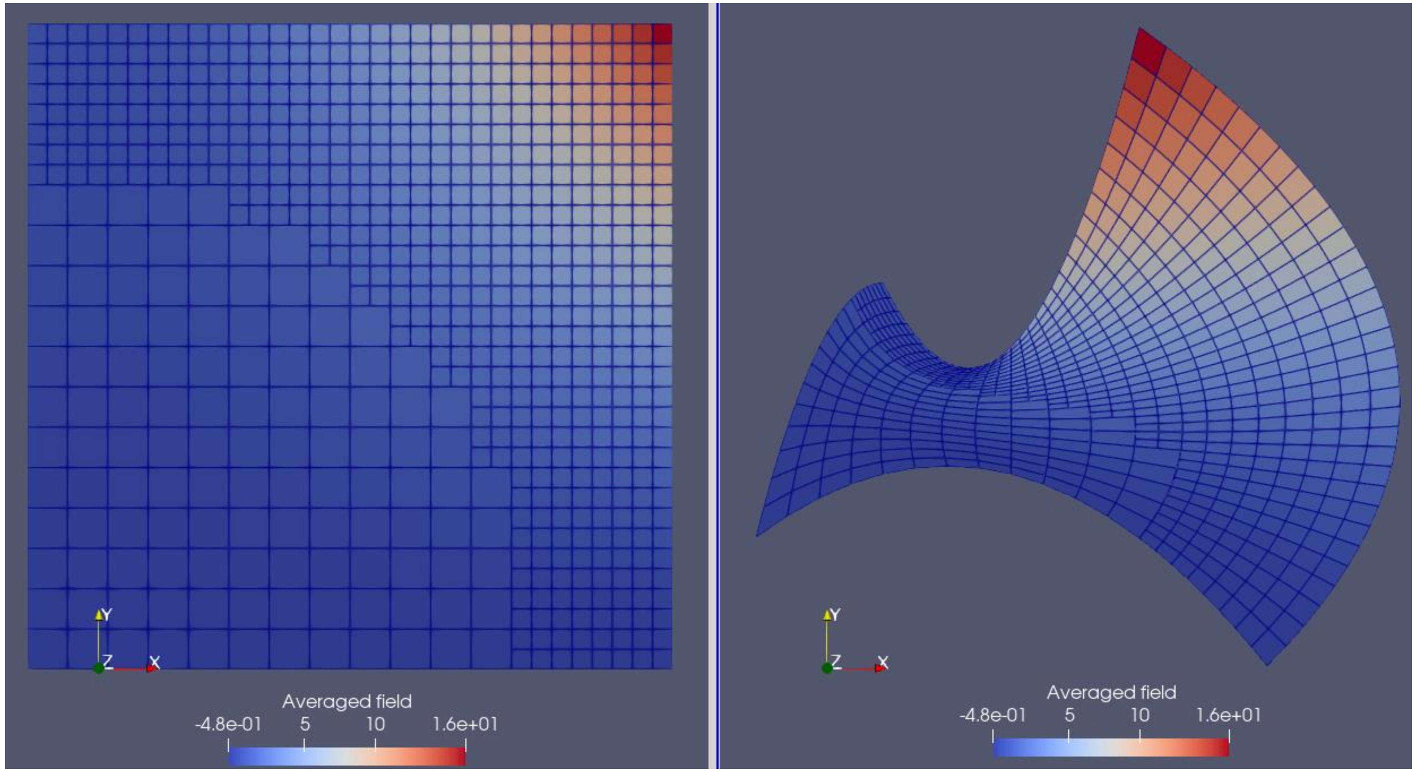
# Curved meshes (CAD enhanced)



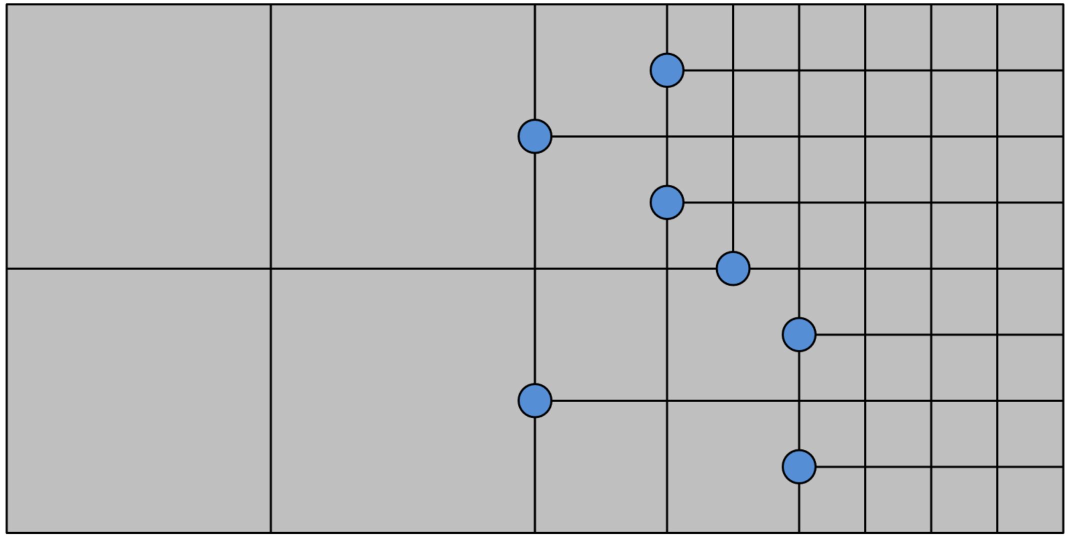
- Allows geometry-driven refinement and solving
- Approaches real geometry even on high refinement levels

# Curved meshes (Lagrangian) developed by Zoltan Csati (of CERFACS)




- Lagrangian high-order approach
- Compatible to typical curved meshes
- Fast and accurate

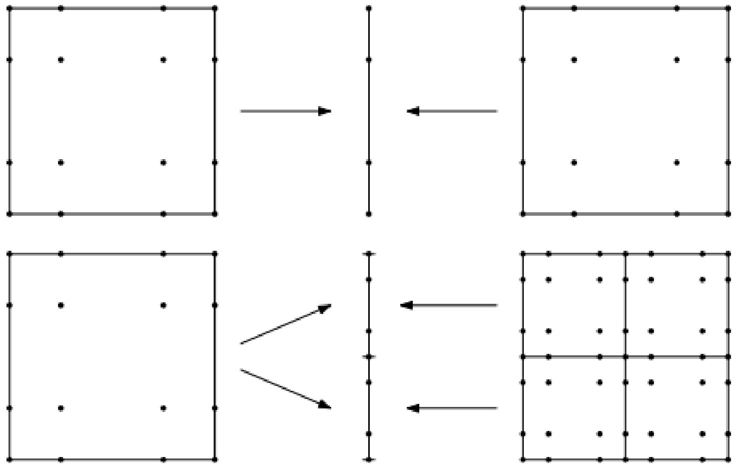
# Coping/resolving hanging nodes



Hanging nodes:

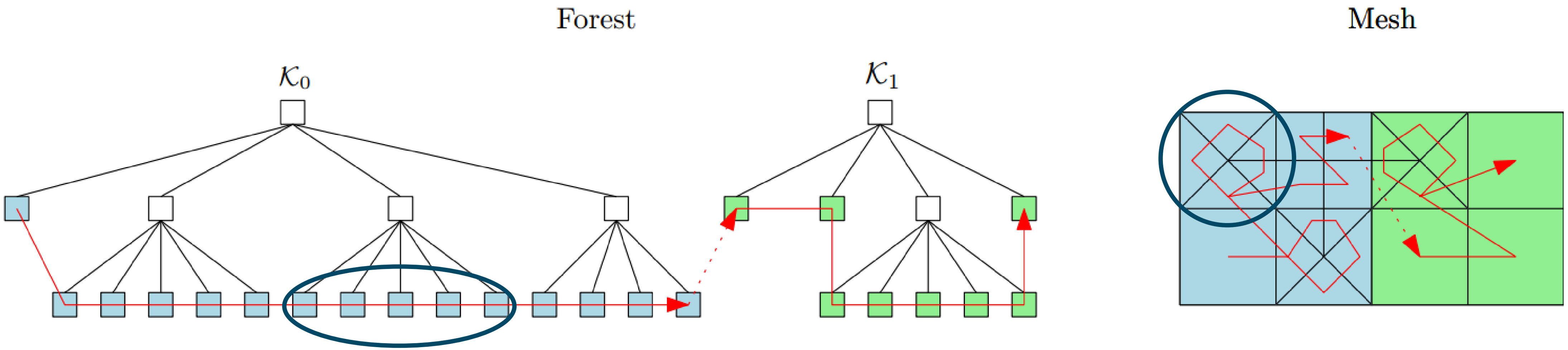
- Nodes introduced by refining
- “Hang” on the face/edge of the neighboring element
- Have to be treated by the solver

# Coping/resolving hanging nodes



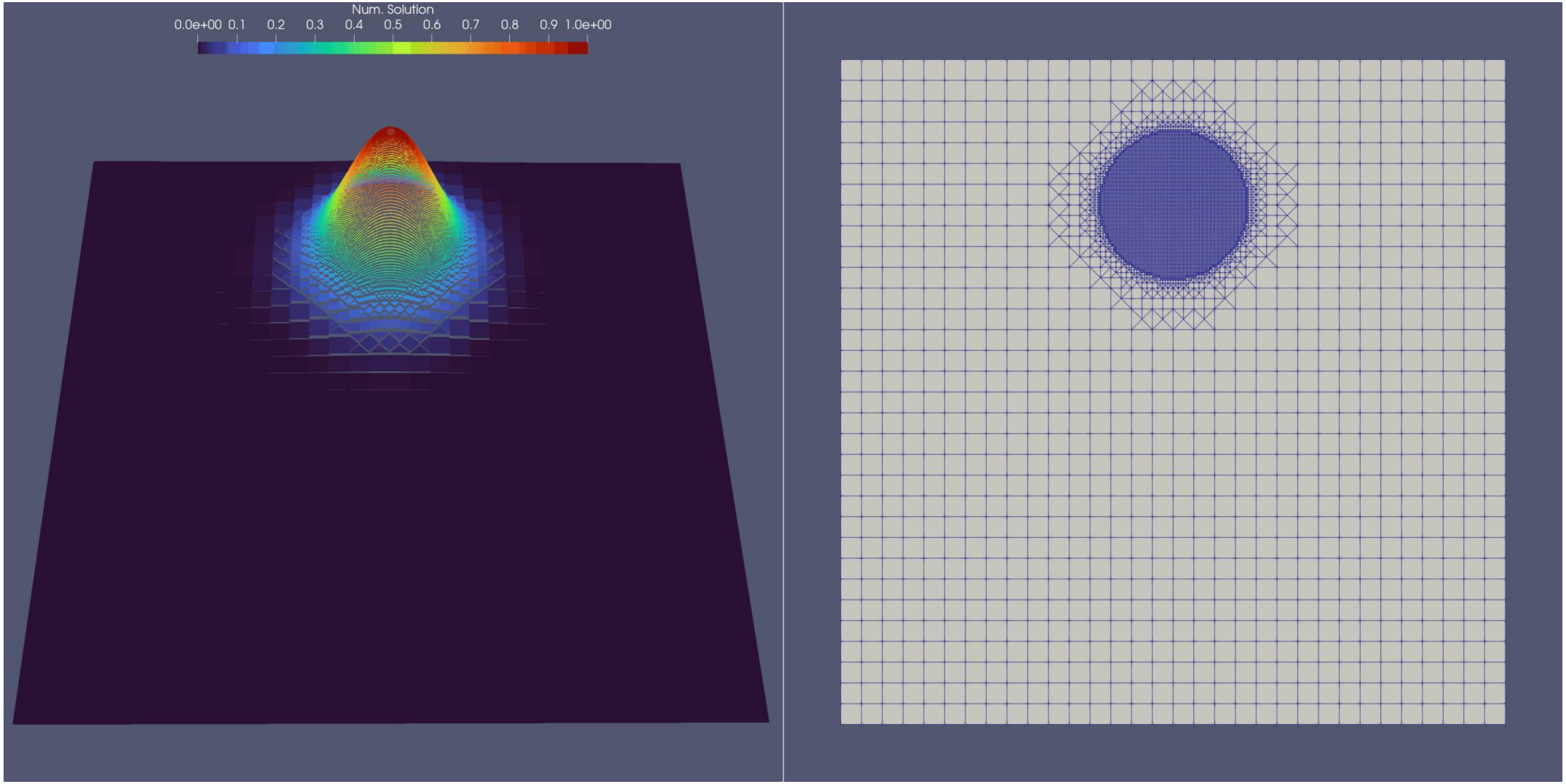
- Mortar Method resolves hanging nodes on the solver side
- Balance routine can ensure 2:1 level difference

# Subelements / resolving hanging nodes (in 2D)

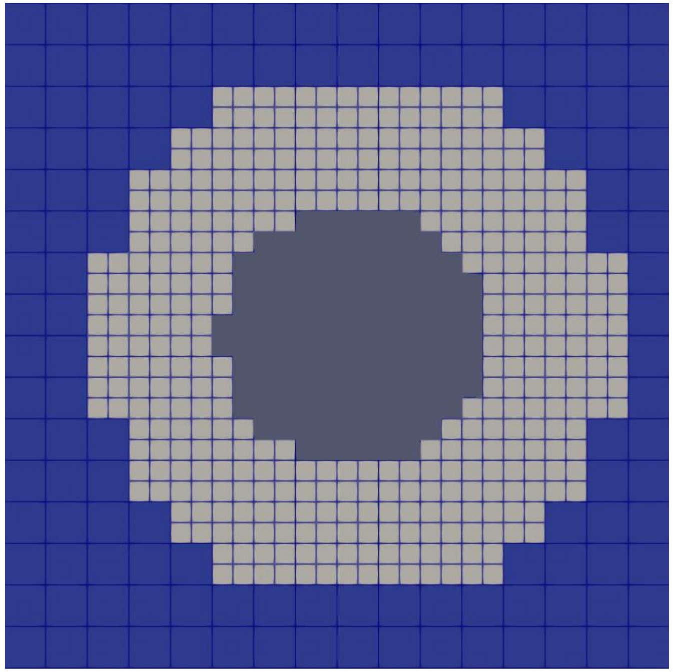


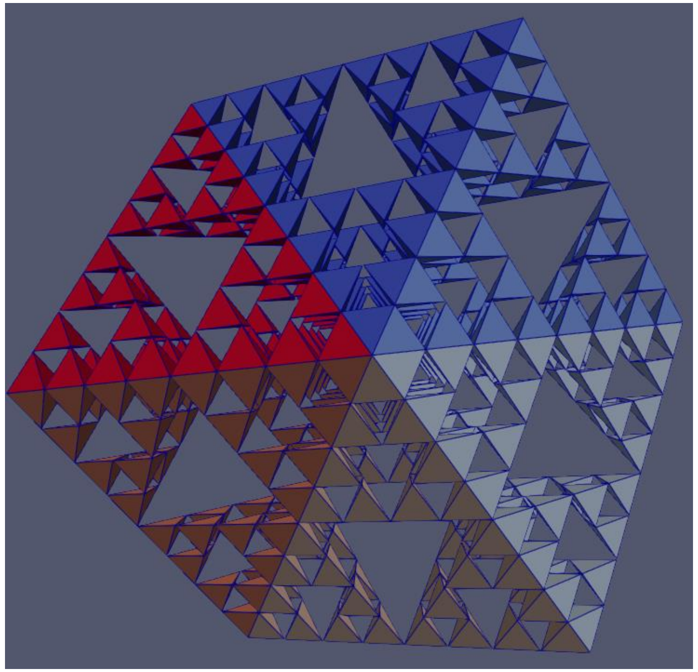

- Introduction of subelements
- Subelements can only be leaf elements
- Subelements behave like elements
  - They implement a subset of low-level algorithms
  - Iteration, ghost elements, etc.

# Subelements / resolving hanging nodes (in 2D)



# Deleting elements / incomplete trees

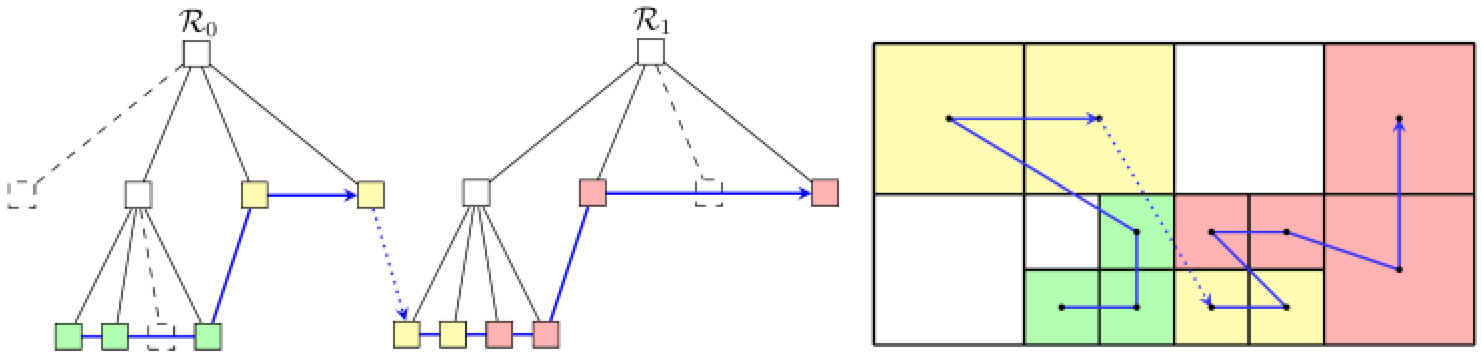




Deleting elements can be used for:

- Embedding obstacles in the mesh
- Deleting regions without interest
- Compressing data

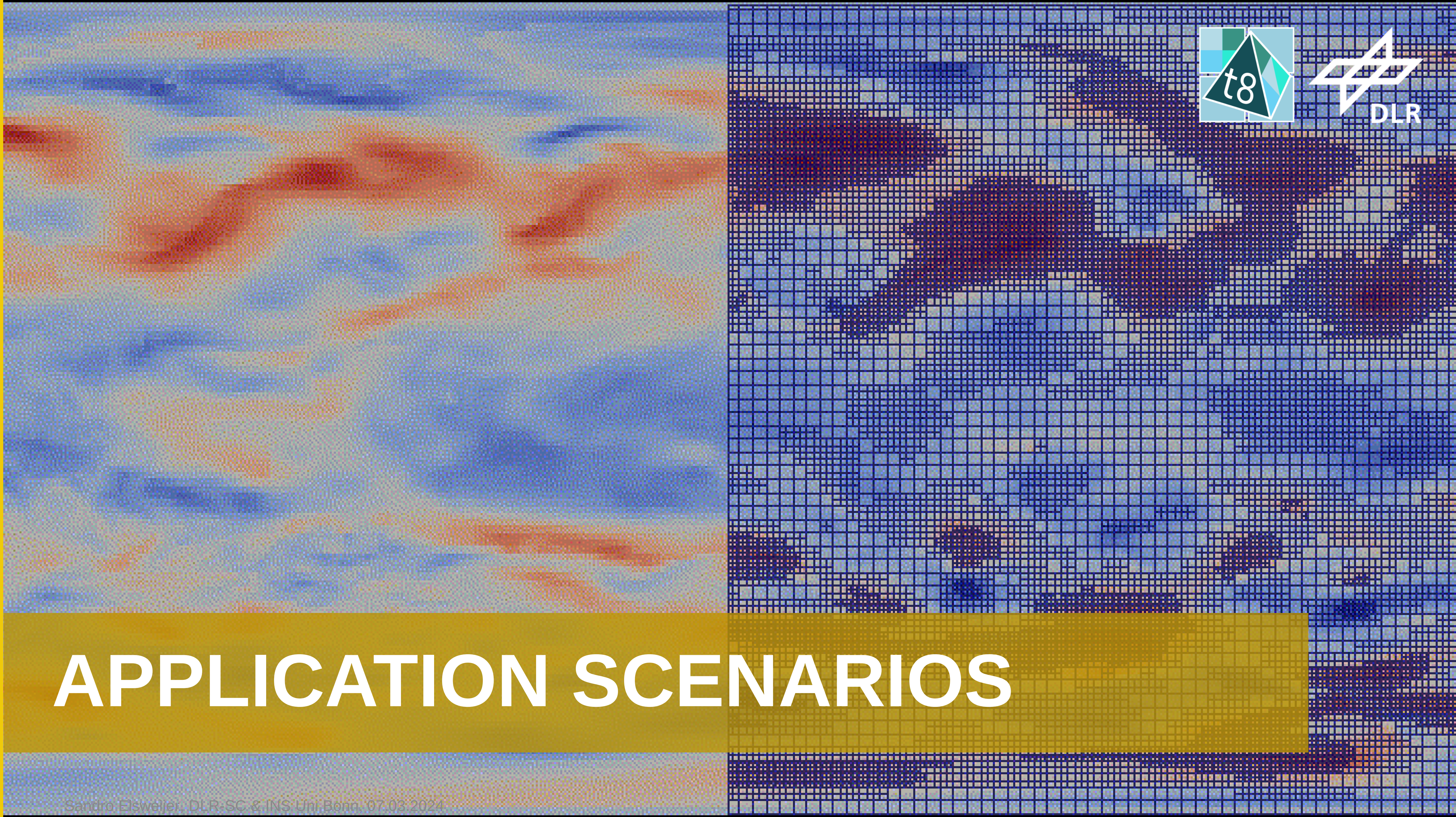
# Deleting elements / incomplete trees




(a)

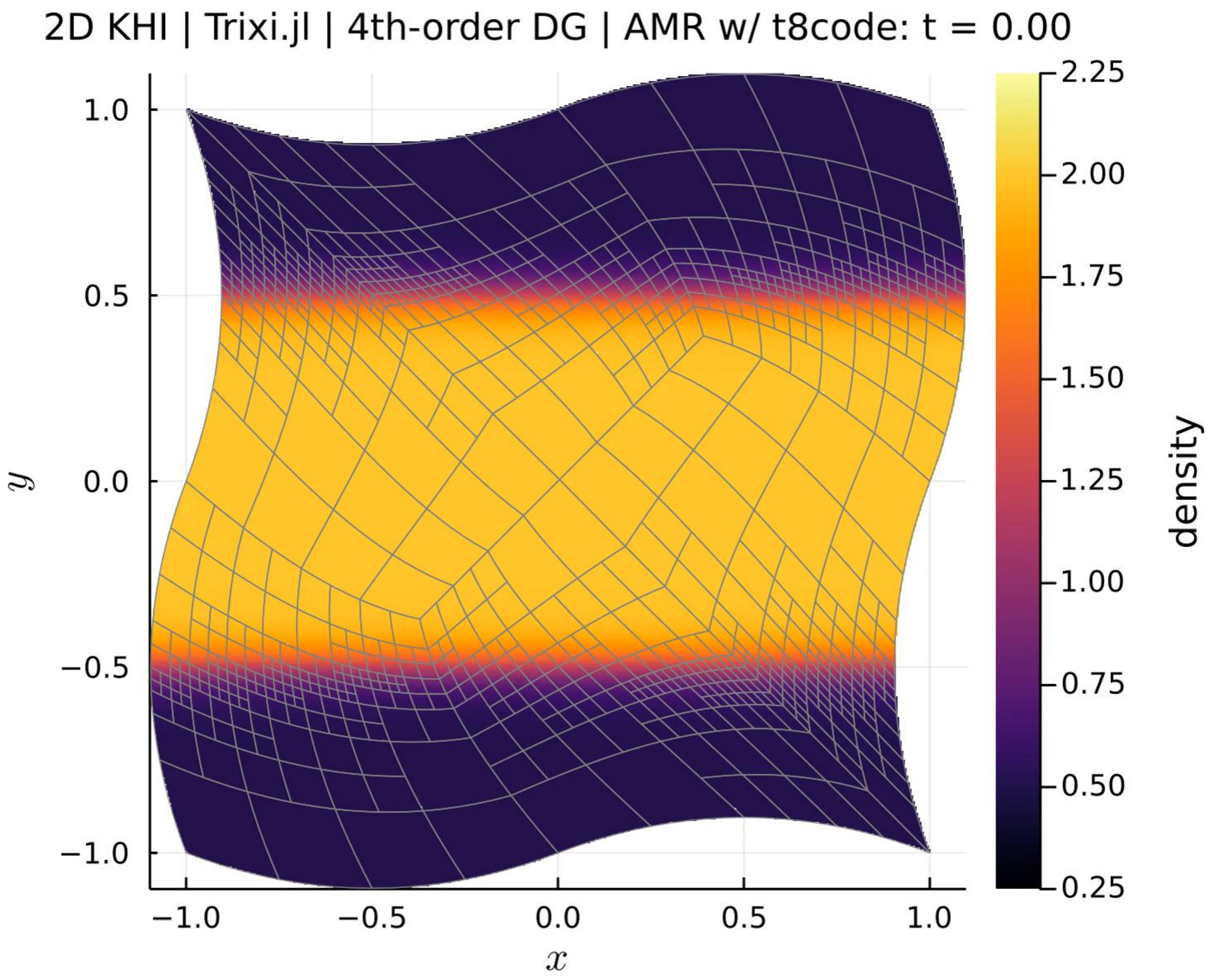
(b)

- No „virtual elements“ of weight 0 or similar constructs
- No memory needed for unused elements

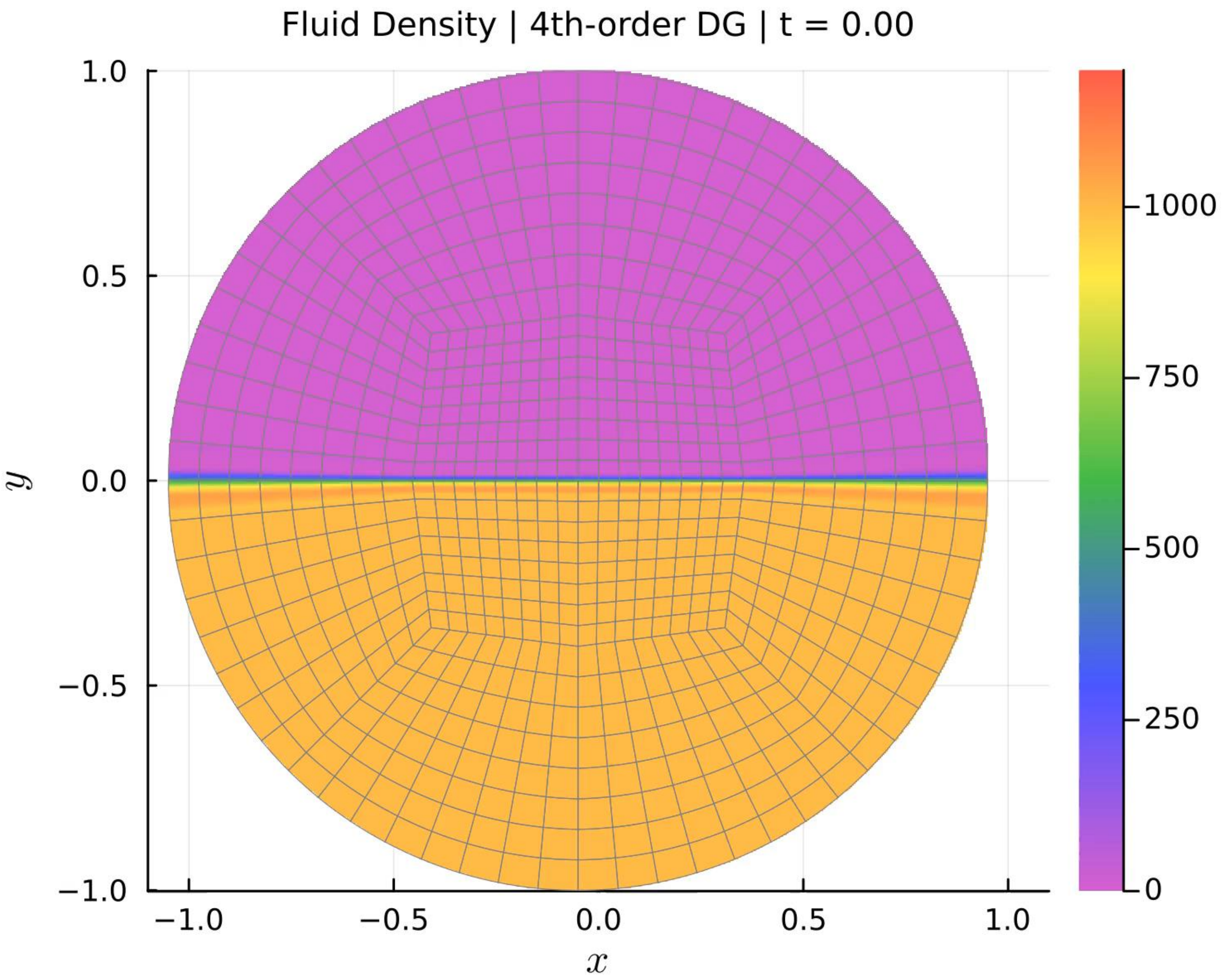


# APPLICATION SCENARIOS

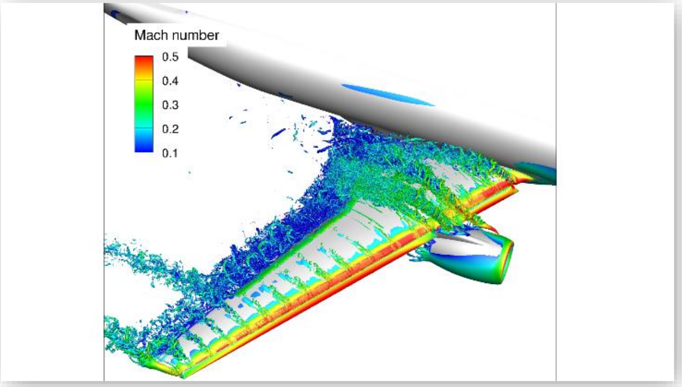
# Use cases – Kelvin-Helmholtz instability

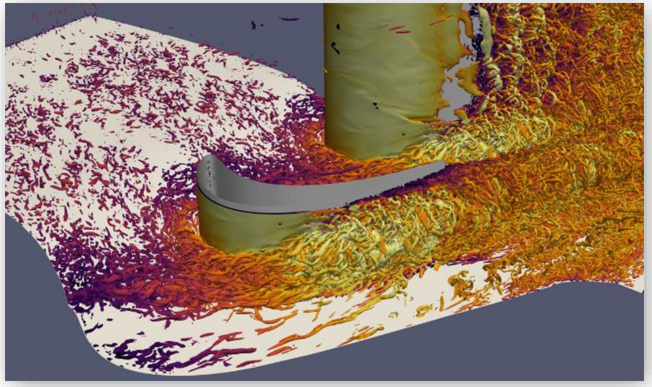


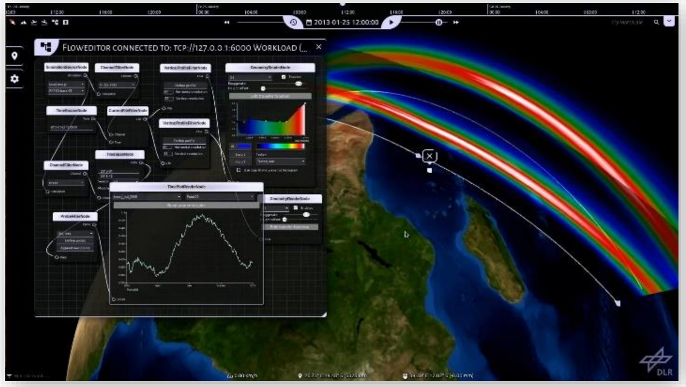
# Use cases – Tank sloshing



# Data visualization





Simulations produce large datasets

- Visual data analyzation of large datasets difficult
- Local workstations do not suffice
- Access to HPC systems limited / expensive

→ Use AMR in a ParaView Plugin

# Data visualization

**Example: Planetary convection**

- Simulation with
  - 794 timesteps
  - 20 M cells
  - 949 GB of data

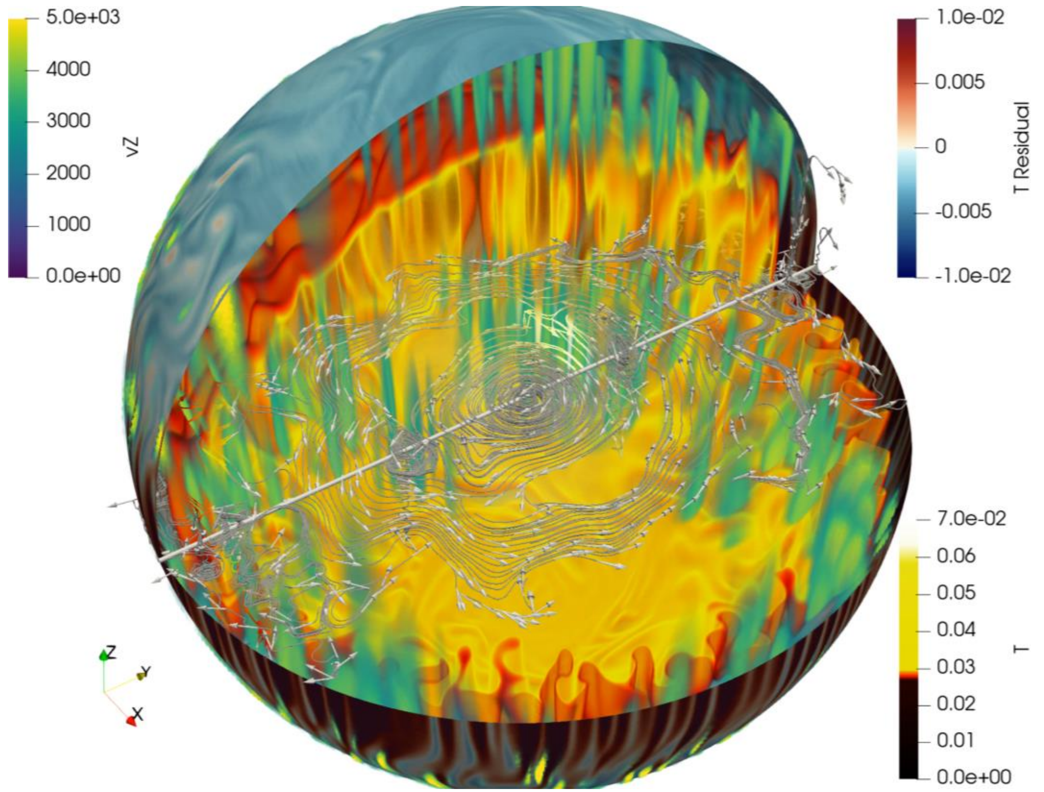
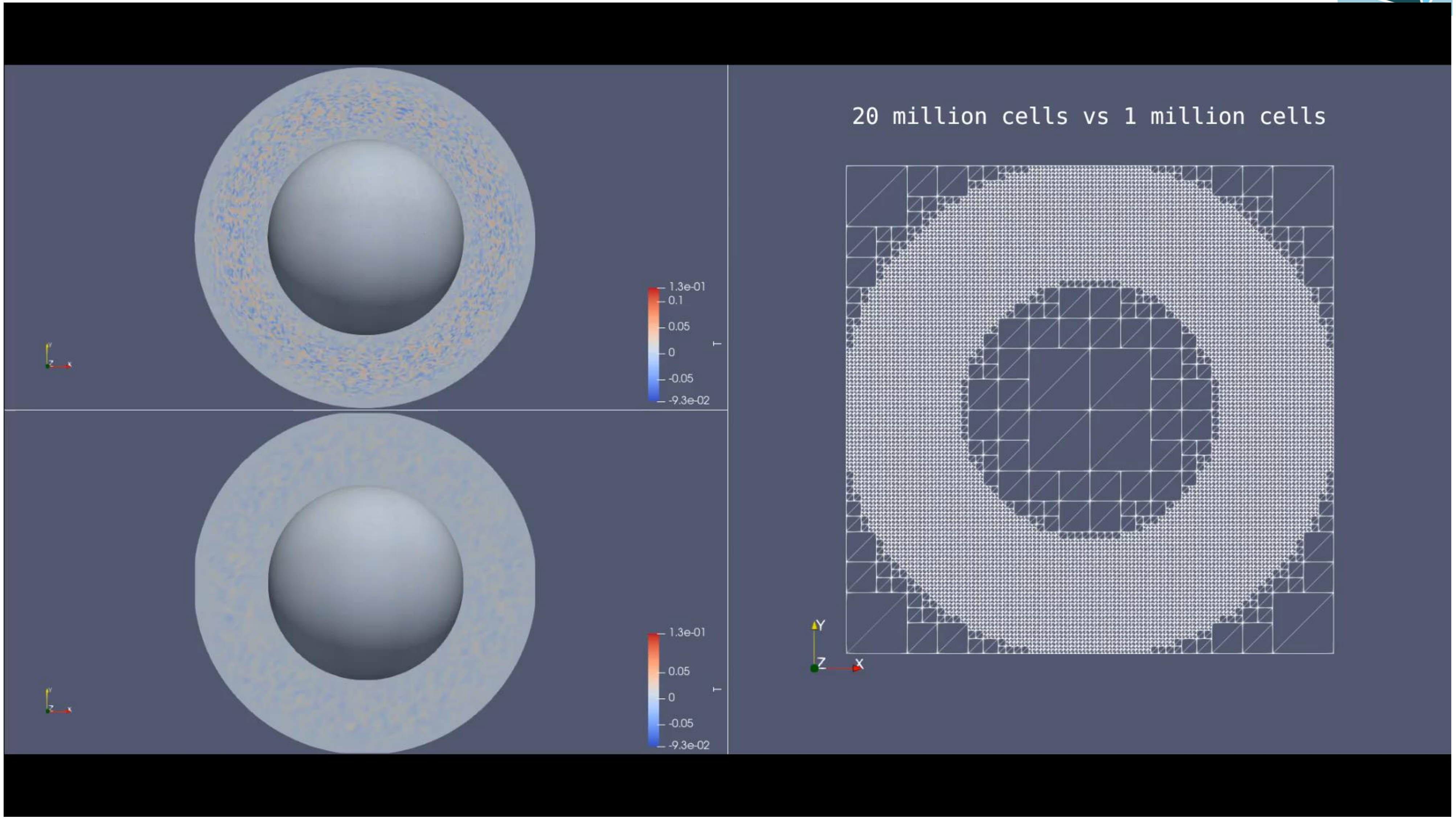
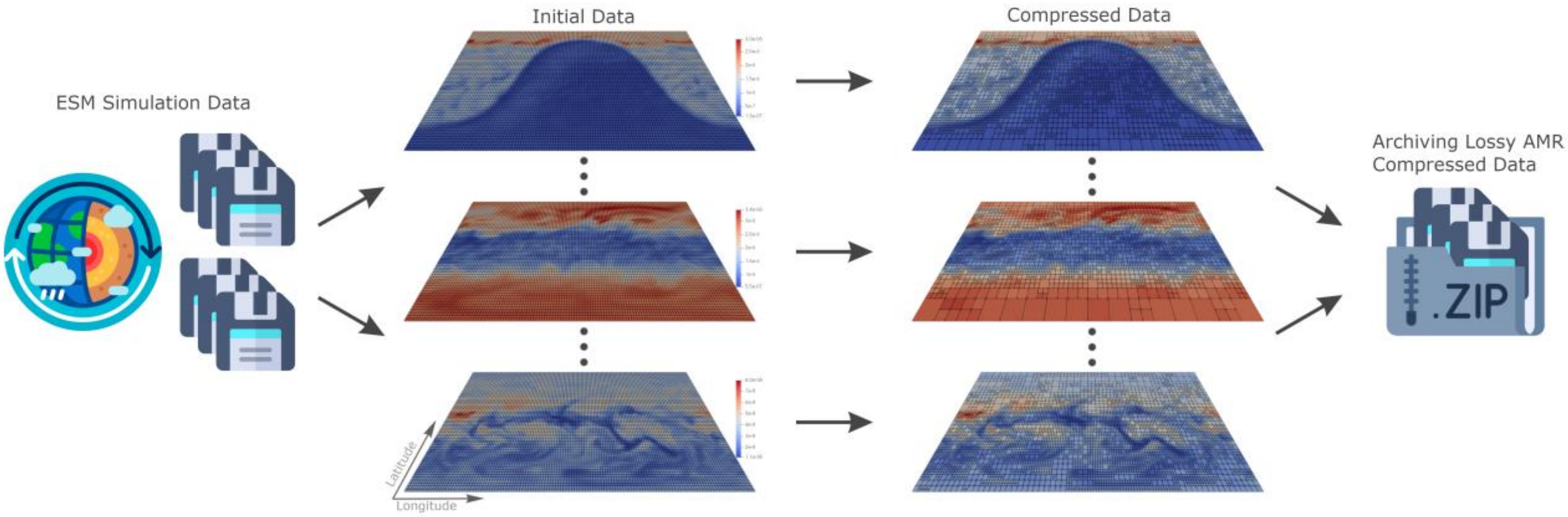



Image by DLR institute for planetary research

# Data visualization (VISPLORE)



# Data compression



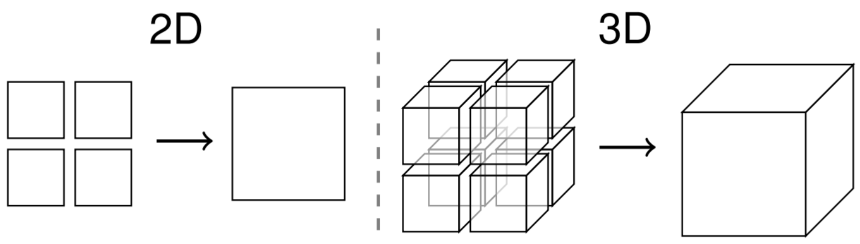

Same problem as before:

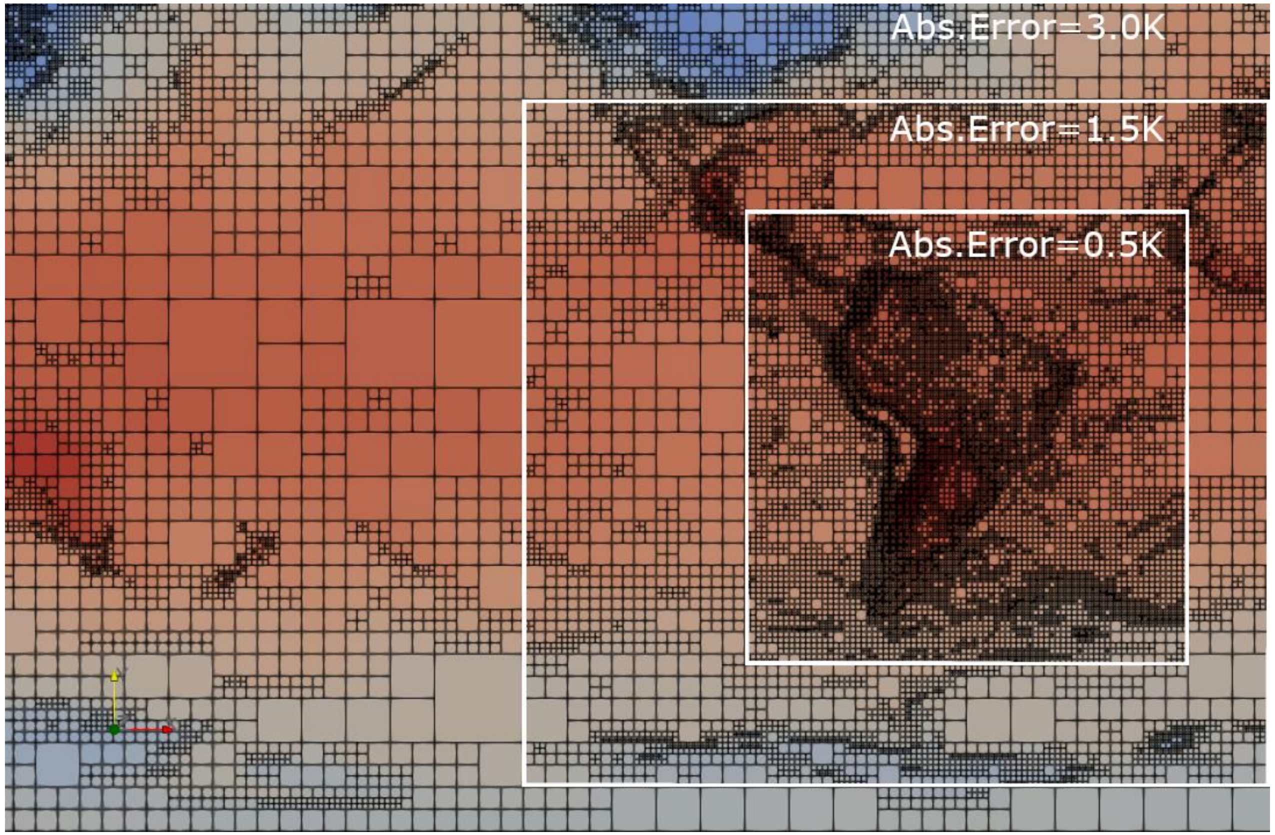
- Increasing (simulation) dataset sizes
- Use AMR for lossy data compression

# Data compression

AMR permits:

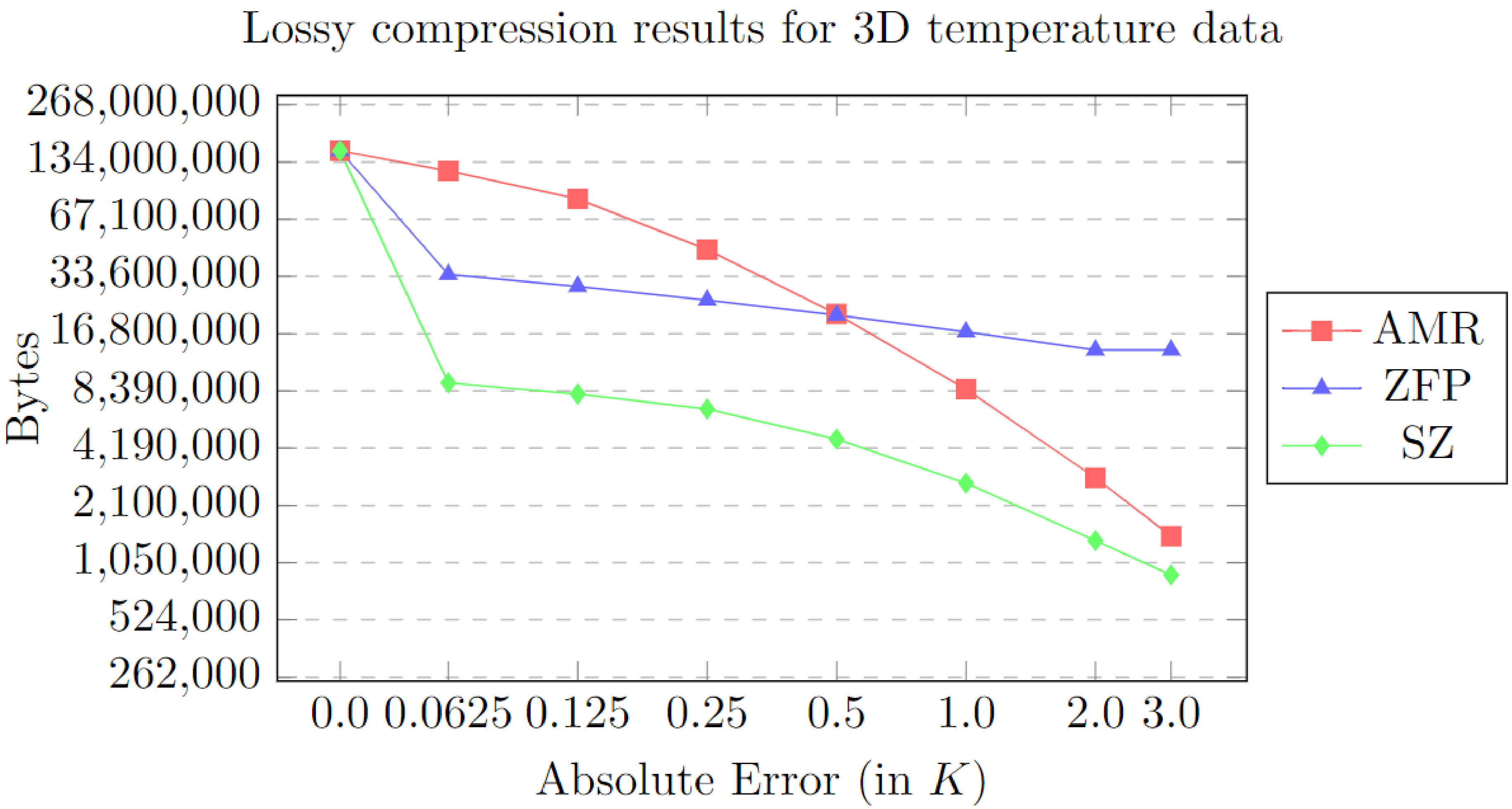
- Absolute/Relative error criteria
- Domain exclusion
- Nested error domains




Compression of ERA5 temperature data with different nested error domains specifying different absolute errors (units: Kelvin)

# Compression Results




Compression of 3D ERA5 temperature data; Comparison with different lossy compressors (i.e. SZ, ZFP)

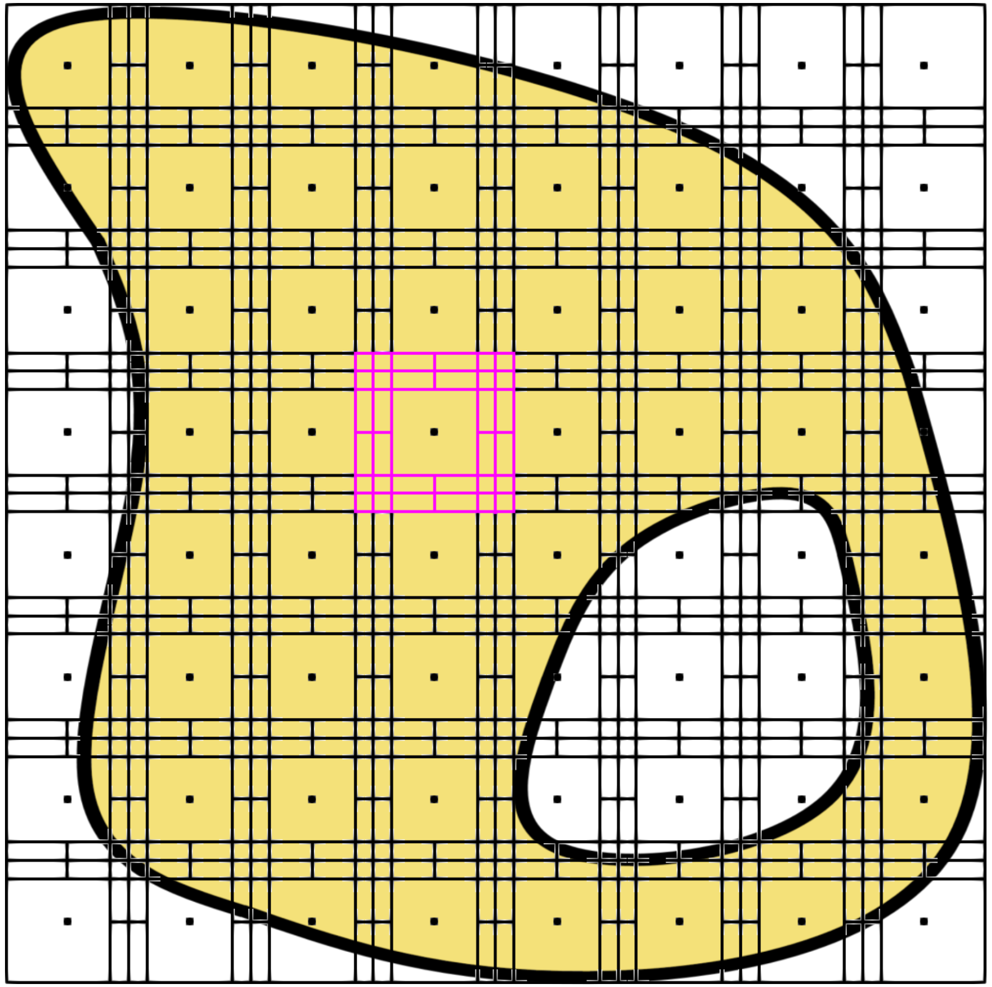
# Partition of Unity Method (PUM) in PUMA

Institute for Numerical Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

DLR

Fraunhofer SCAI

- Meshfree multiscale method
- Discretization via overlapping patches



Images kindly provided by the Fraunhofer Gesellschaft

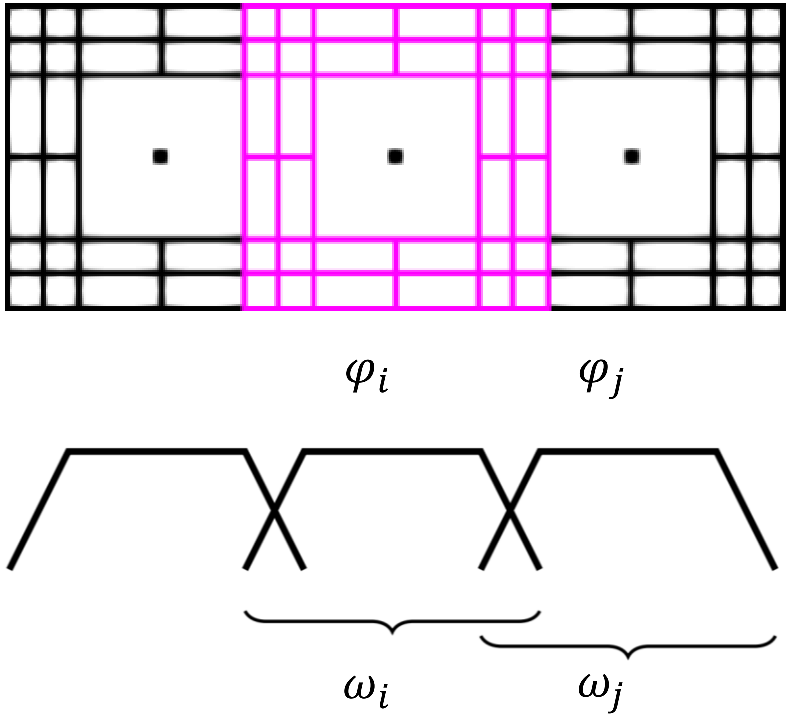
# Partition of Unity Method (PUM) in PUMA

Institute for Numerical Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

DLR

Fraunhofer SCAI

- Meshfree multiscale method
- Discretization via overlapping patches
- Partition of unity (PU) $\{\varphi_i\}$ with
  $$\omega_i \coloneqq \mathrm{supp}(\varphi i)$$
- Smooth splicing of local spaces
  $$V^{\mathrm{PU}} := \sum_{i=1}^{N} \varphi_i V_i(\omega_i)$$
- Approximation by $V_i(\omega_i)$, functions $\varphi_i$ just „glue“




Images kindly provided by the Fraunhofer Gesellschaft

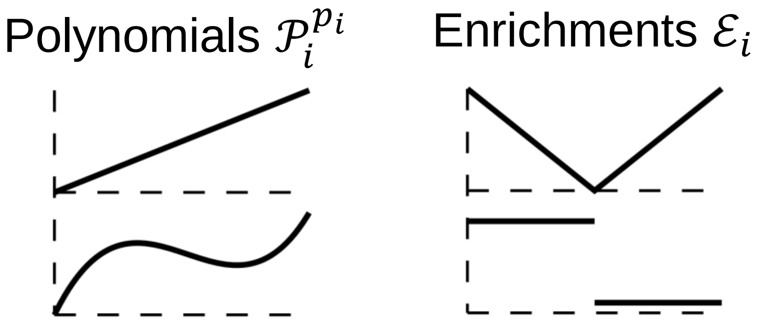
# Partition of Unity Method (PUM) in PUMA

Institute for Numerical Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

t8 DLR Fraunhofer SCAI

$$V^{\mathrm{PU}} := \sum_{i=1}^{N} \varphi_i V_i(\omega_i) = \sum_{i=1}^{N} \varphi_i(\mathcal{P}^{p_i} + \mathcal{E}_i)$$

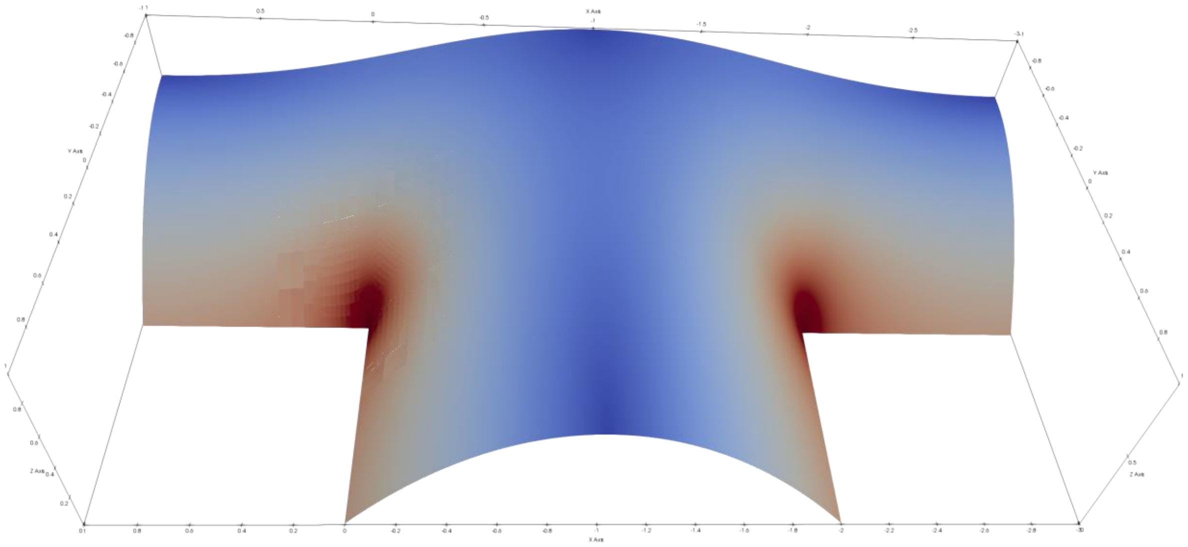
- Smooth polynomials $\mathcal{P}_i^{p_i}$
- Enrichments $\mathcal{E}_i$ depend on physics

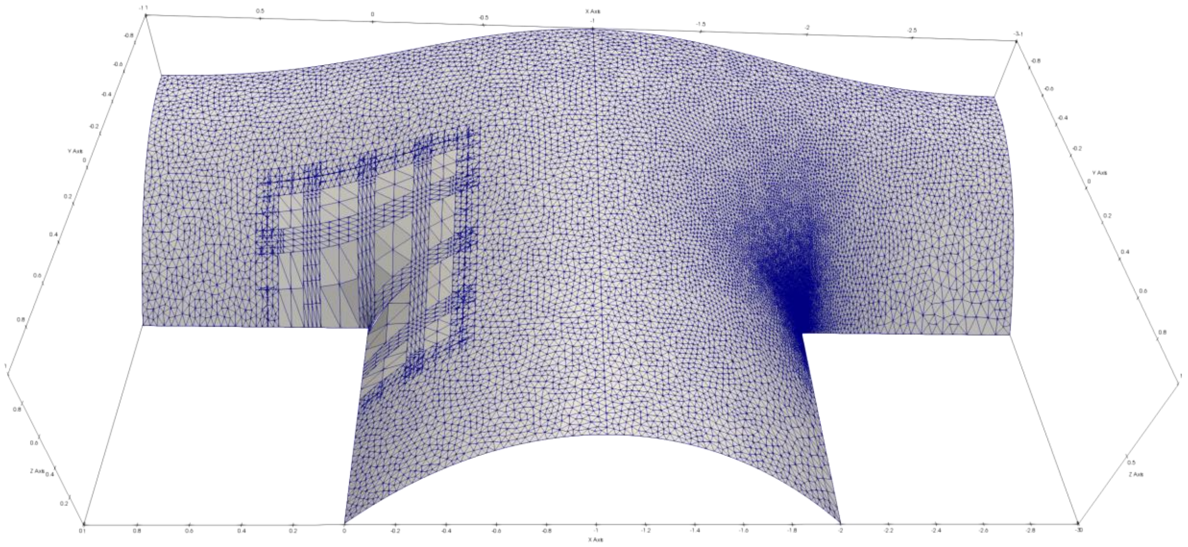



Images kindly provided by the Fraunhofer Gesellschaft

# Partition of Unity Method (PUM) in PUMA

## Example: PUM vs FEM



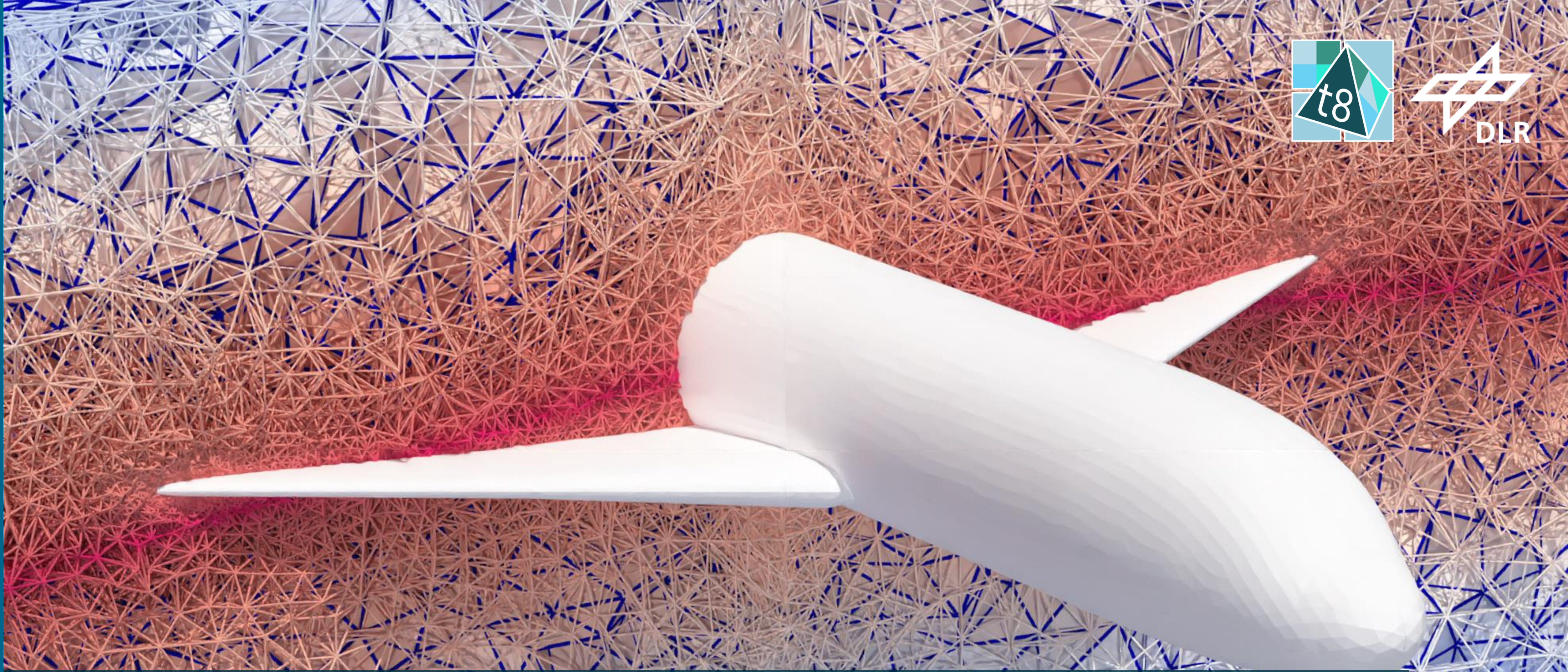


**100** PUMA higher order enrichment functions

**50,000** locally refined finite elements

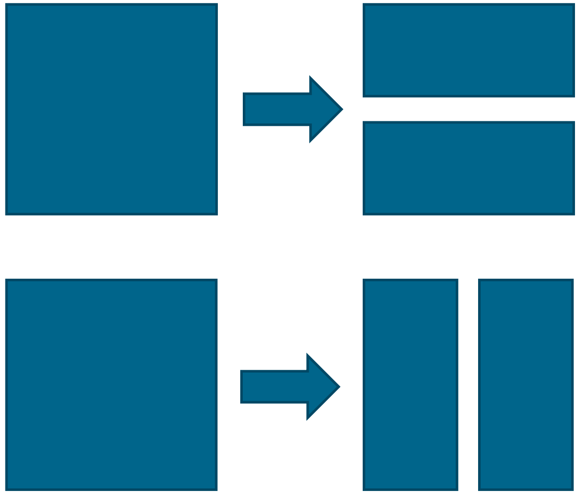


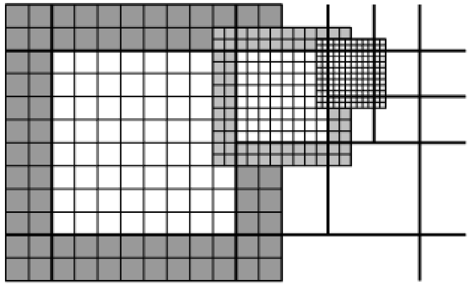
Images kindly provided by the Fraunhofer Gesellschaft



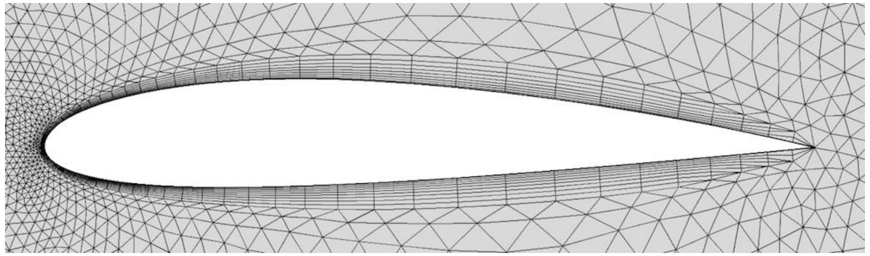
# FUTURE PLANS

# Subelements

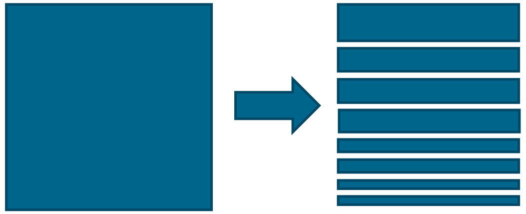




Donna Calhoun et. al.



https://www.comsol.fr/blogs/your-guide-to-meshing-techniques-for-efficient-cfd-modeling/



Anisotropic refinement

Uniform subgrids for GPUs

Boundary layers