Removing Hanging Faces from Tree-Based Adaptive Meshes for Numerical Simulations – an implementation in t8code

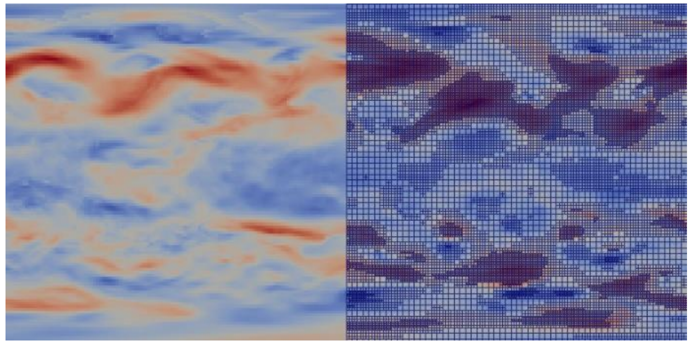
SIAM Annual Meeting – Undergraduate Research, July 2022

Florian Becker (German Aerospace Center | High-Performance Computing),
Gregor Gassner (University of Cologne | Numerical Simulation),
Johannes Holke (German Aerospace Center | High-Performance Computing)

DLR

Adaptive Mesh Refinement (AMR) in Numerical Applications with t8code

numerical application with AMR



Why AMR?

- Many mathematical applications are based on mesh structures
- The element size correlates with the accuracy (but also with the computational costs)
- AMR as a compromise of high accuracy and computational efficiency

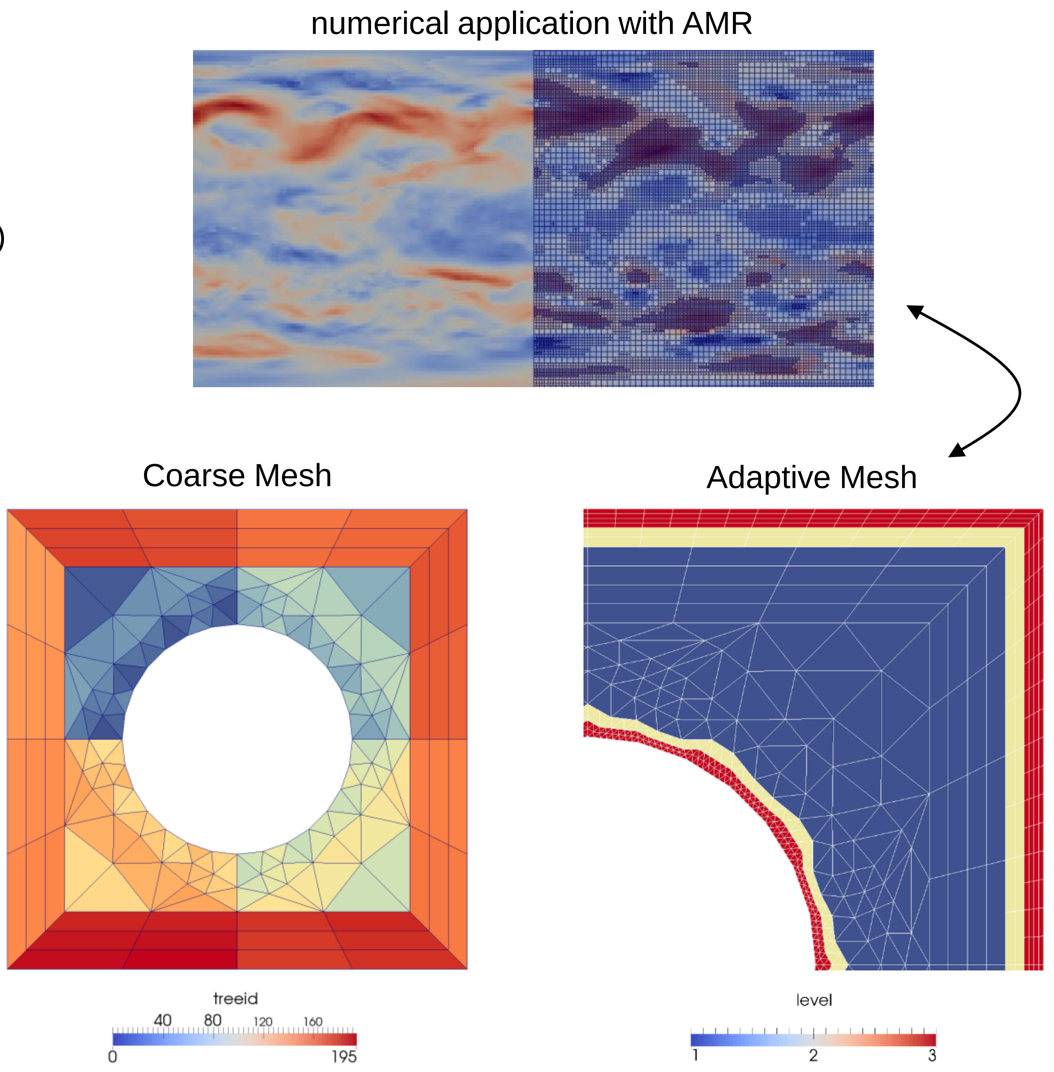
Adaptive Mesh Refinement (AMR) in Numerical Applications with t8code

Why AMR?

- Many mathematical applications are based on mesh structures
- The element size correlates with the accuracy (but also with the computational costs)
- AMR as a compromise of high accuracy and computational efficiency

t8code ("tetcode"):

- A open-source C/C++ library for parallel AMR [4]
- Tree-based hybrid meshes
- Adaptive refinement of each tree
- A modular Space-filling Curve Index (SFC Index) for element enumeration

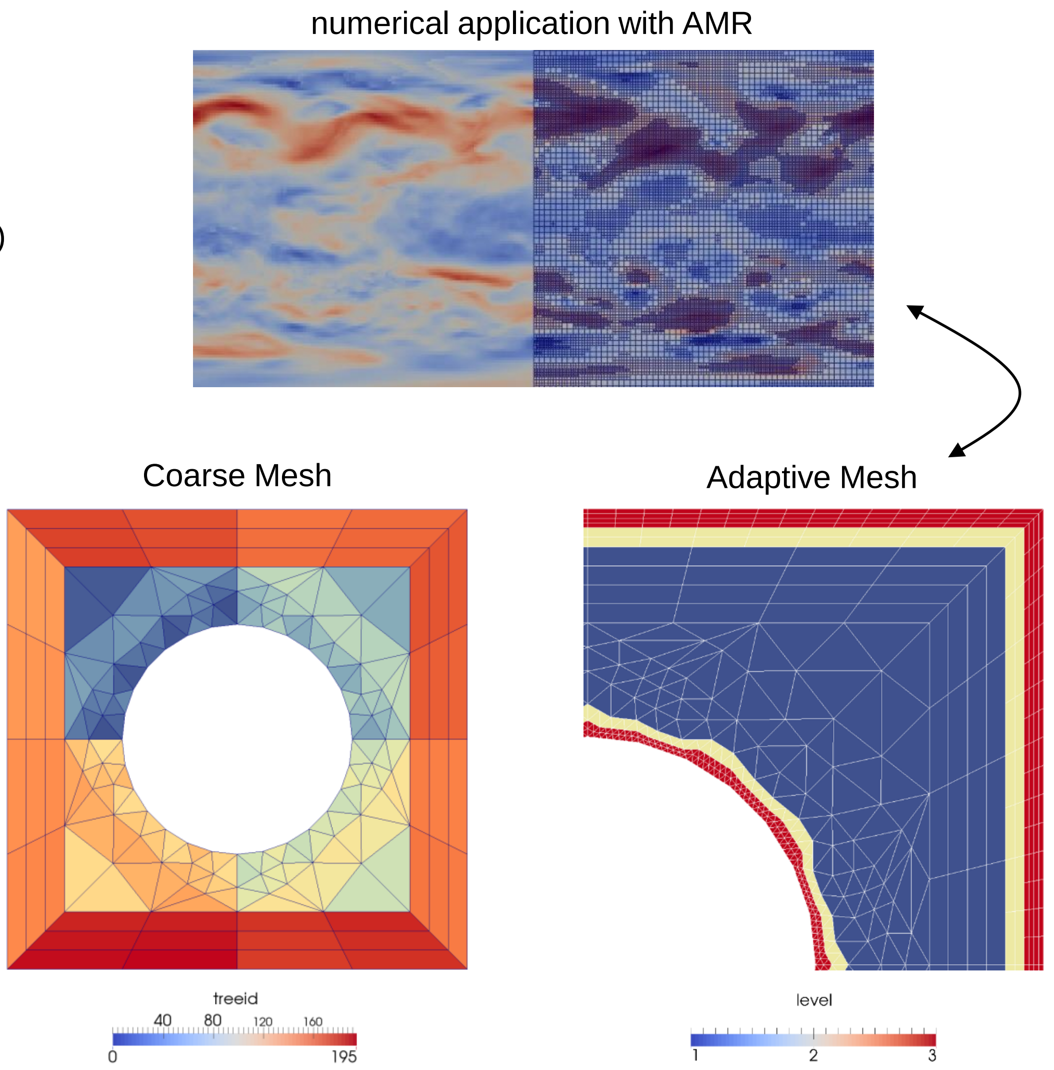


Adaptive Mesh Refinement (AMR) in Numerical Applications with t8code

Why AMR?
- Many mathematical applications are based on mesh structures
- The element size correlates with the accuracy (but also with the computational costs)
- AMR as a compromise of high accuracy and computational efficiency

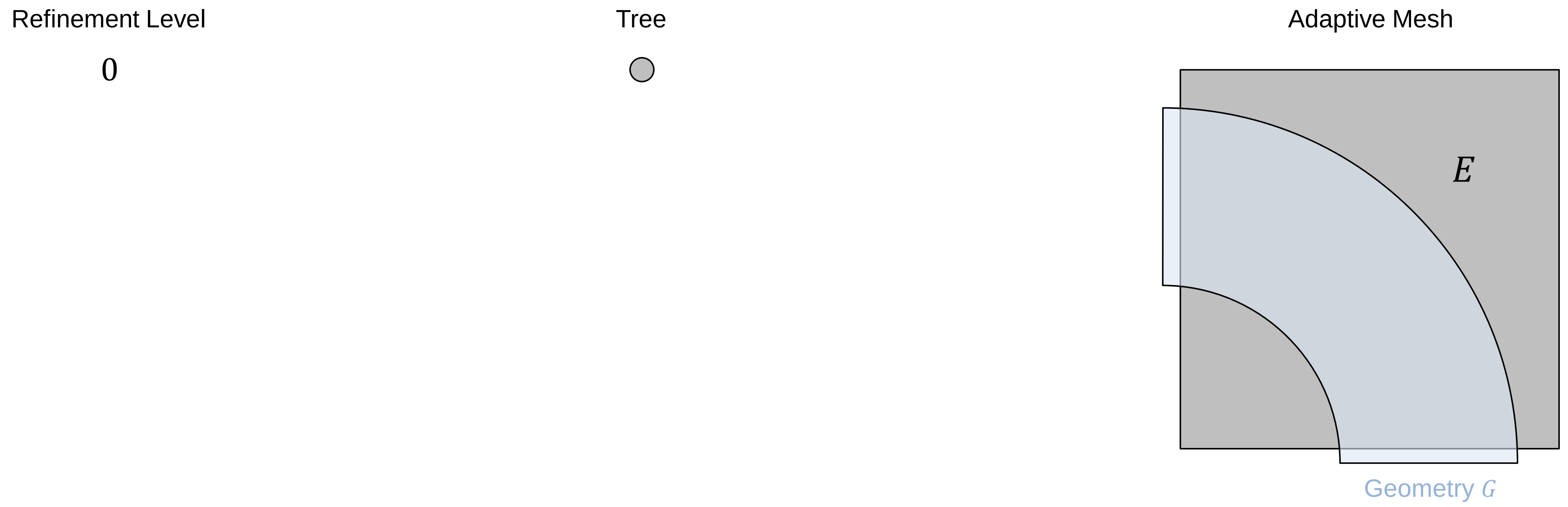
t8code ("tetcode"):
- A open-source C/C++ library for parallel AMR [4]
- Tree-based hybrid meshes
- Adaptive refinement of each tree
- A modular Space-filling Curve Index (SFC Index) for element enumeration

How does tree-based AMR work?



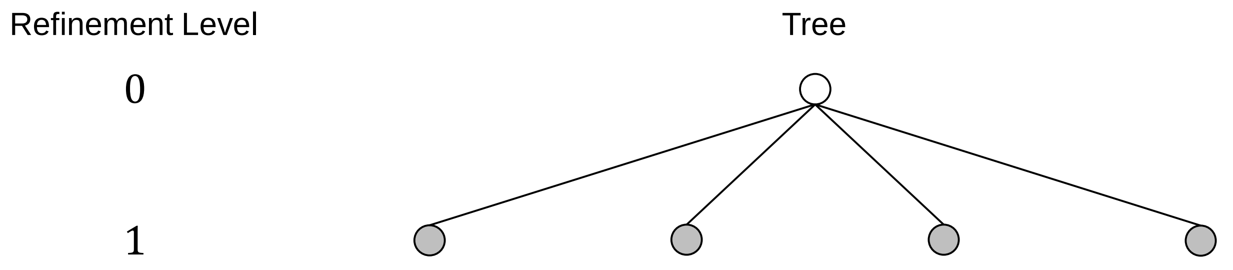
Tree-based AMR

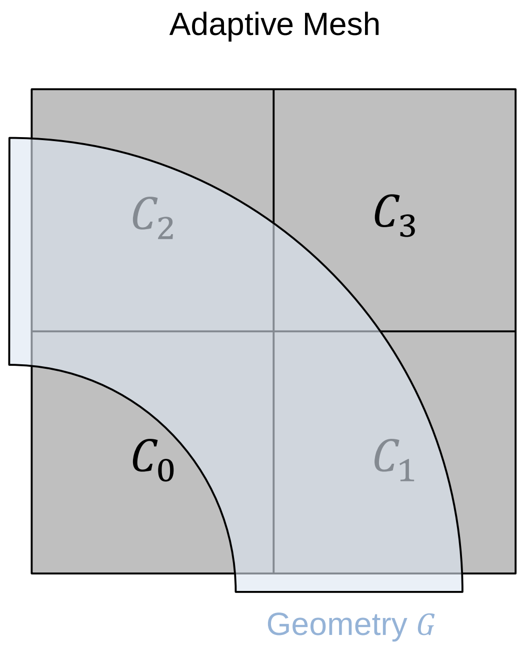
- **Refinement criterion**: refine an element E in the mesh if $E \cap G \neq \emptyset$
- **Refinement:** the replacement of an element E by its children: $E \to \{C_0, C_1, C_2, C_3\}$ with $\cup_i C_i = E$



Tree-based AMR

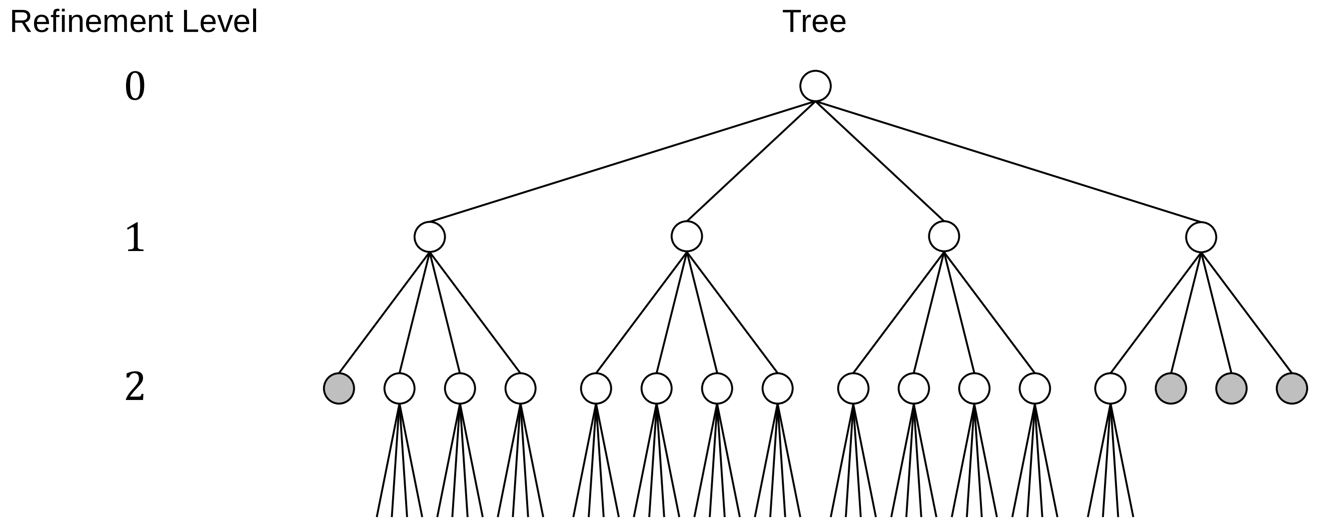
- **Refinement criterion**: refine an element E in the mesh if $E \cap G \neq \emptyset$
- **Refinement:** the replacement of an element E by its children: $E \rightarrow \{C_0, C_1, C_2, C_3\}$ with $\cup_i C_i = E$

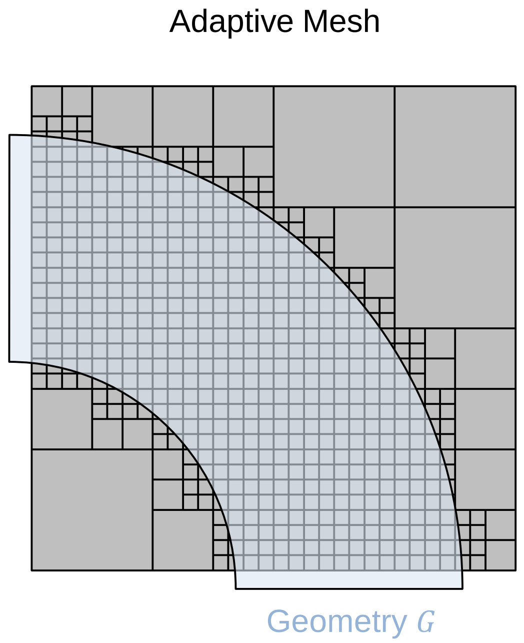




Tree-based AMR

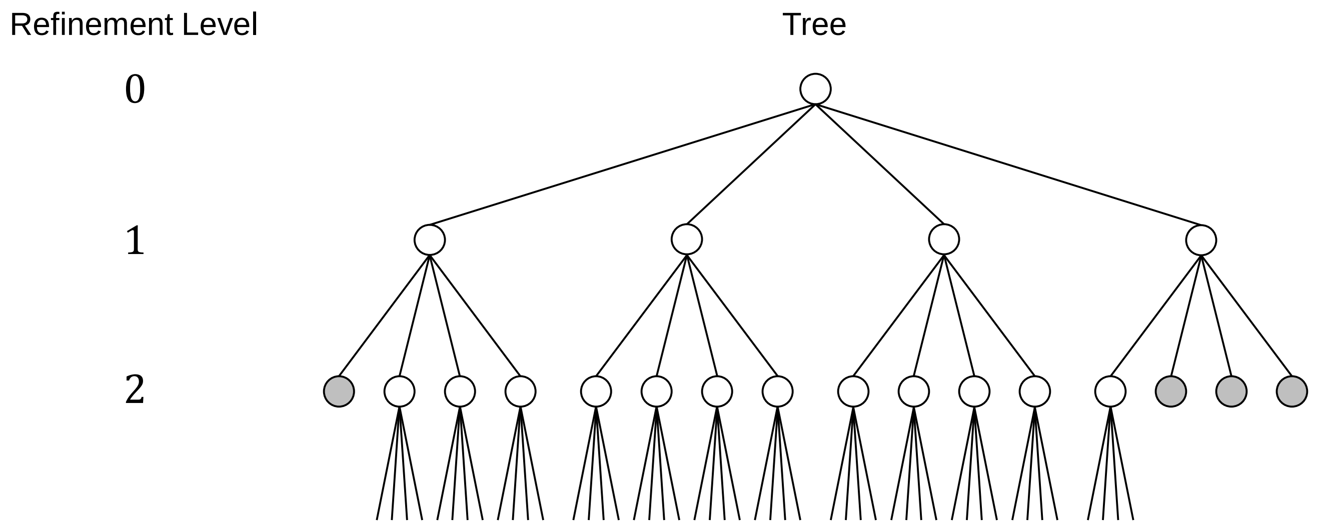
- **Refinement criterion**: refine an element E in the mesh if $E \cap G \neq \emptyset$
- **Refinement:** the replacement of an element E by its children: $E \rightarrow \{C_0, C_1, C_2, C_3\}$ with $\cup_i \, C_i = E$

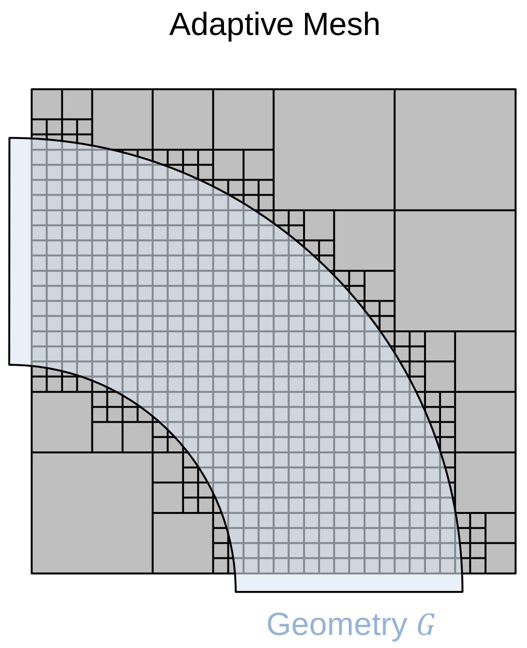




Tree-based AMR

- **Refinement criterion**: refine an element E in the mesh if $E \cap G \neq \emptyset$
- **Refinement:** the replacement of an element E by its children: $E \rightarrow \{C_0, C_1, C_2, C_3\}$ with $\cup_i \, C_i = E$

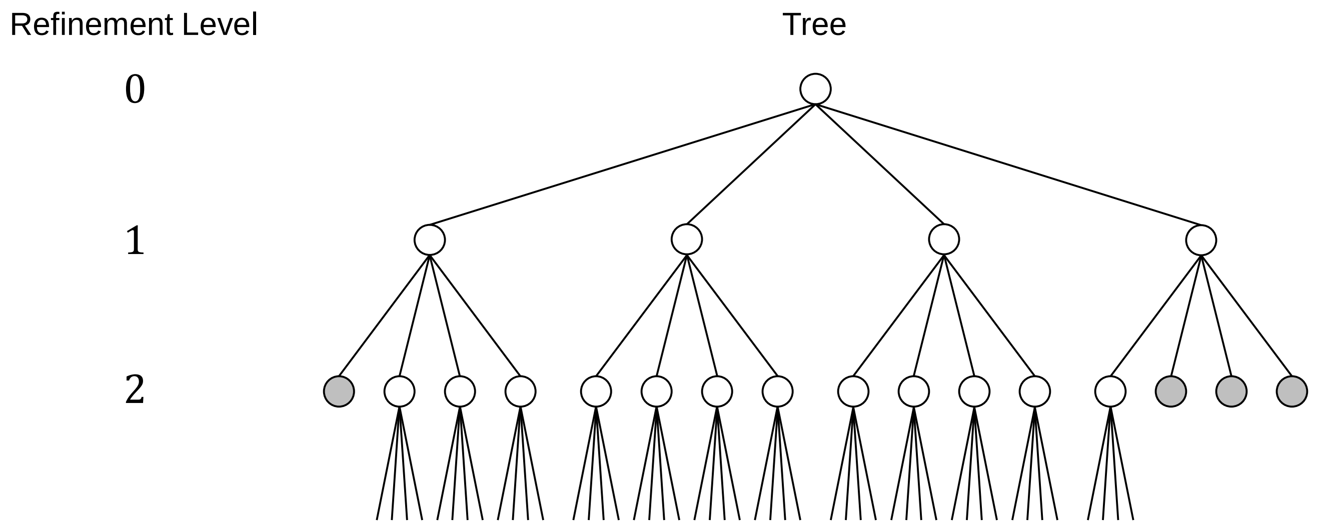


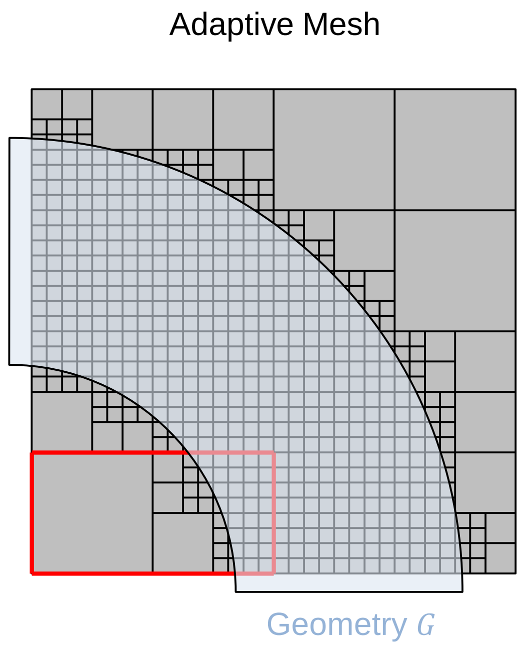


- **Hanging Faces:** the intersection of an elements vertex with an other elements face

Tree-based AMR

- **Refinement criterion**: refine an element E in the mesh if $E \cap G \neq \emptyset$
- **Refinement:** the replacement of an element E by its children: $E \rightarrow \{C_0, C_1, C_2, C_3\}$ with $\cup_i C_i = E$





- **Hanging Faces:** the intersection of an elements vertex with an other elements face

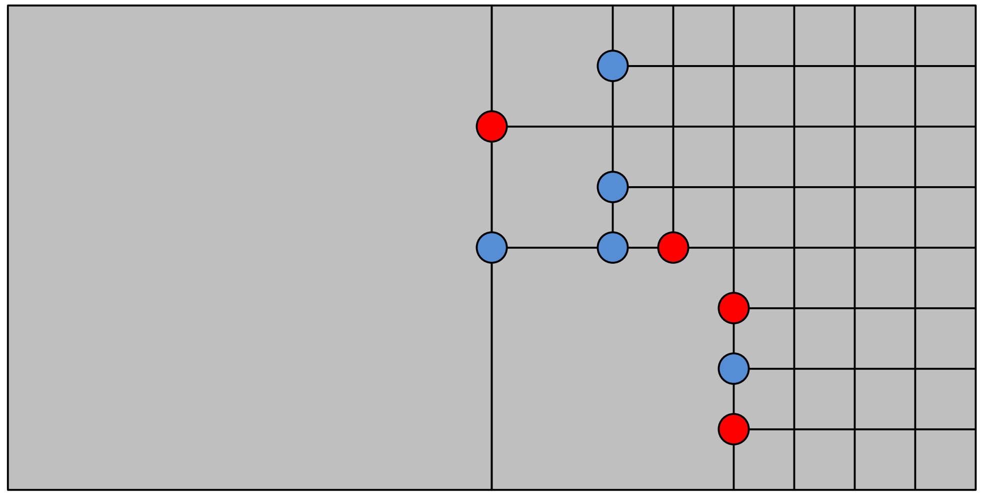
How can we make the mesh conformal after adaptation?

Removing Hanging Faces via Transition Cells of Triangular Subelements

Removing hanging faces is a two-step procedure*:

1. Remove red nodes: balance the mesh
2. Remove blue nodes: use transition cells of triangular subelements

Adapted Mesh



*originally a recursive one-step procedure, proposed by Schneiders [5]

Removing Hanging Faces via Transition Cells of Triangular Subelements

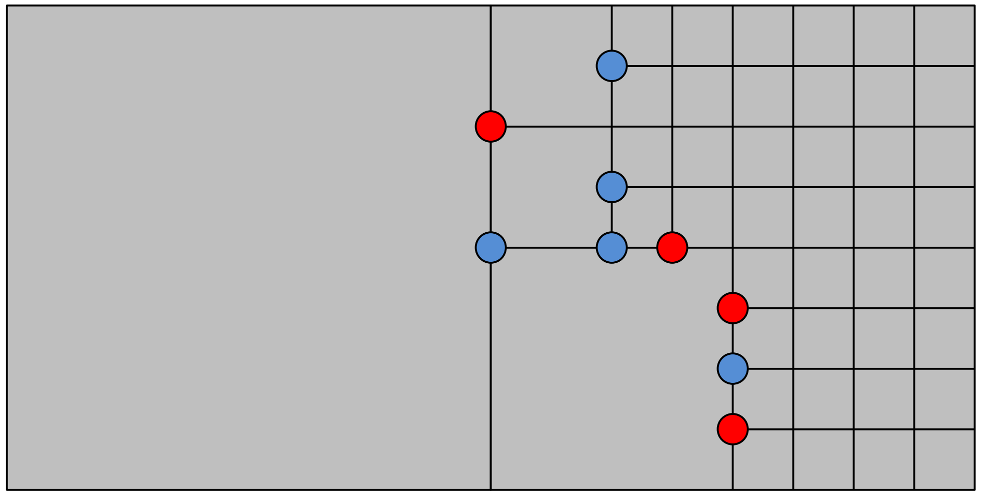
Removing hanging faces is a two-step procedure*:

1. Remove red nodes: balance the mesh
2. Remove blue nodes: use transition cells of triangular subelements

1. Balancing:

In balanced meshes, the maximum level difference of neighboring elements is 1.

Adapted Mesh



*originally a recursive one-step procedure, proposed by Schneiders [5]

Removing Hanging Faces via Transition Cells of Triangular Subelements

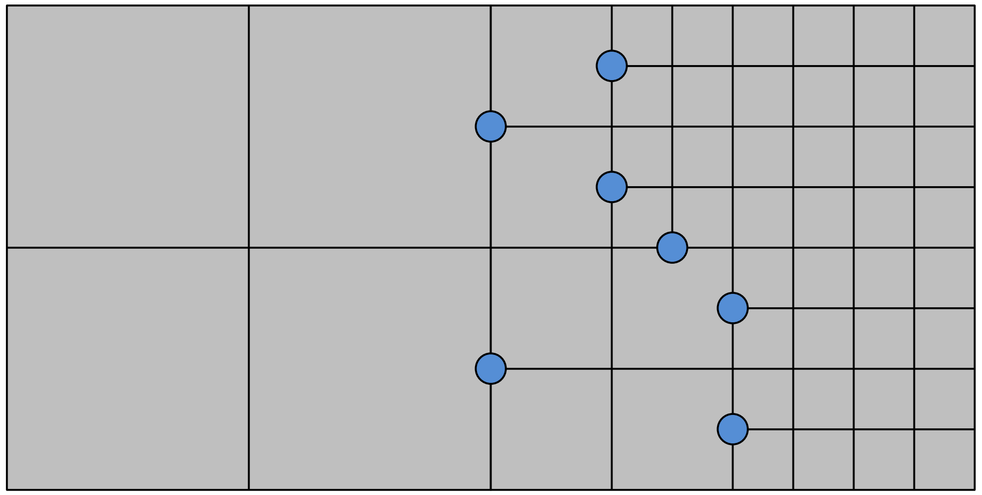
Removing hanging faces is a two-step procedure*:

1. Remove red nodes: balance the mesh
2. Remove blue nodes: use transition cells of triangular subelements

1. Balancing:

In balanced meshes, the maximum level difference of neighboring elements is 1.

Balanced Mesh



*originally a recursive one-step procedure, proposed by Schneiders [5]

Removing Hanging Faces via Transition Cells of Triangular Subelements

Removing hanging faces is a two-step procedure*:

1. Remove red nodes: balance the mesh
2. Remove blue nodes: use transition cells of triangular subelements

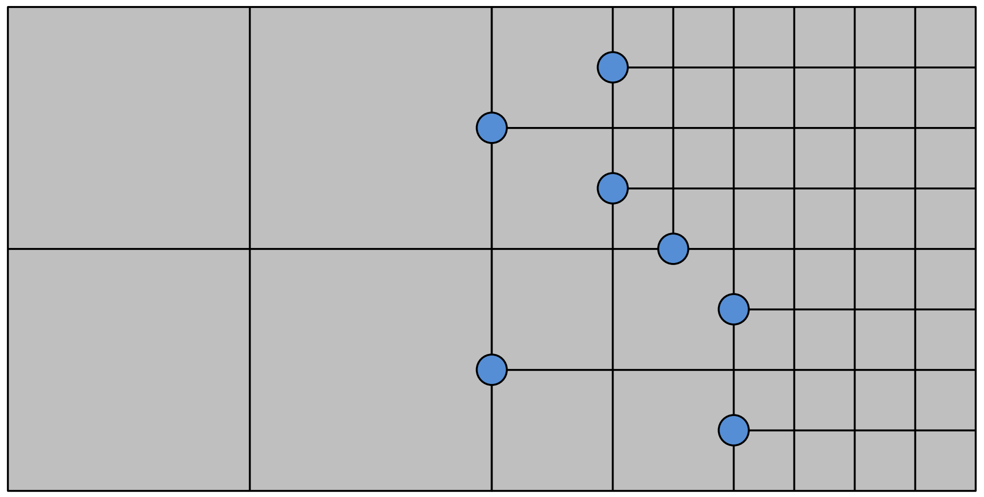
1. Balancing:

In balanced meshes, the maximum level difference of neighboring elements is 1.

2. Transitioning:

A transitioned mesh is a mesh that is conformal due to the insertion of **transition cells** T - families of **triangular subelements** $\{S_0, S_1, \ldots, S_n\}$.

Balanced Mesh



*originally a recursive one-step procedure, proposed by Schneiders [5]

Removing Hanging Faces via Transition Cells of Triangular Subelements

Removing hanging faces is a two-step procedure*:

1. Remove red nodes: balance the mesh
2. Remove blue nodes: use transition cells of triangular subelements

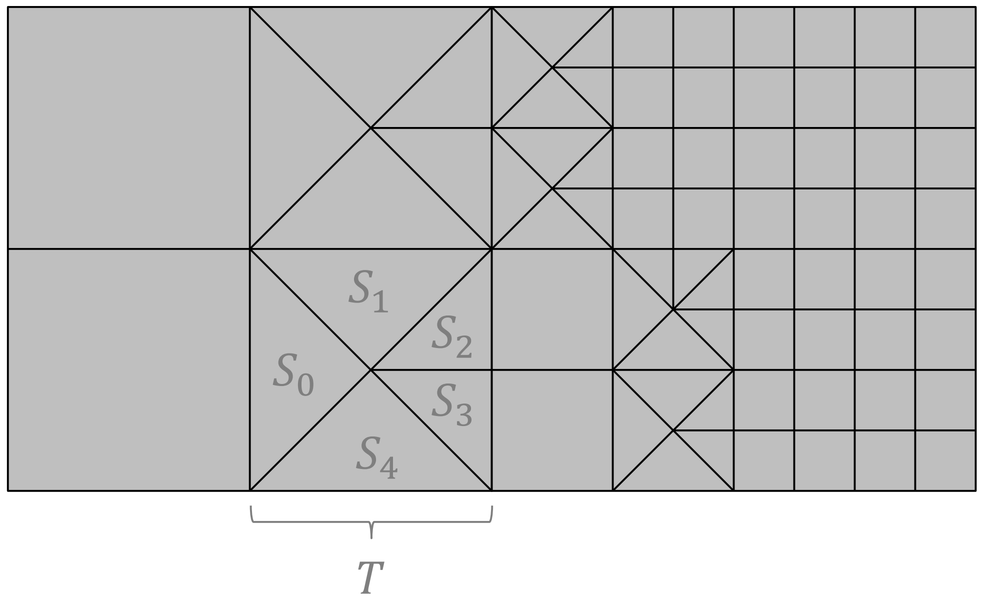
1. Balancing:

In balanced meshes, the maximum level difference of neighboring elements is 1.

2. Transitioning:

A transitioned mesh is a mesh that is conformal due to the insertion of **transition cells** T - families of **triangular subelements** $\{S_0, S_1, \ldots, S_n\}$.

Transitioned Mesh



*originally a recursive one-step procedure, proposed by Schneiders [5]

Removing Hanging Faces via Transition Cells of Triangular Subelements

Removing hanging faces is a two-step procedure*:

1. Remove red nodes: balance the mesh
2. Remove blue nodes: use transition cells of triangular subelements

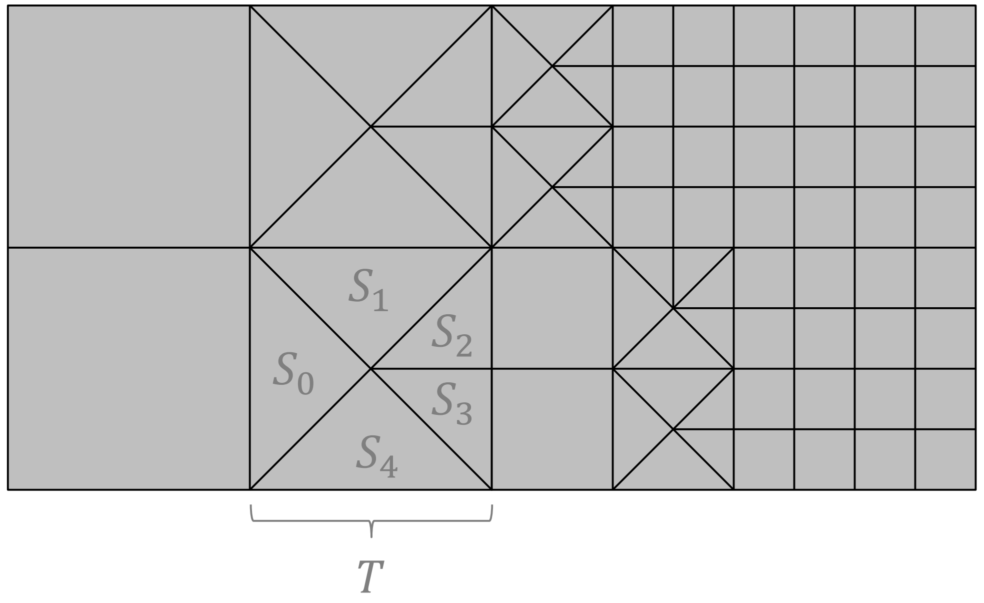
1. Balancing:

In balanced meshes, the maximum level difference of neighboring elements is 1.

2. Transitioning:

A transitioned mesh is a mesh that is conformal due to the insertion of **transition cells** T **-** families of **triangular subelements** $\{S_0, S_1, \dots, S_n\}$.

Transitioned Mesh

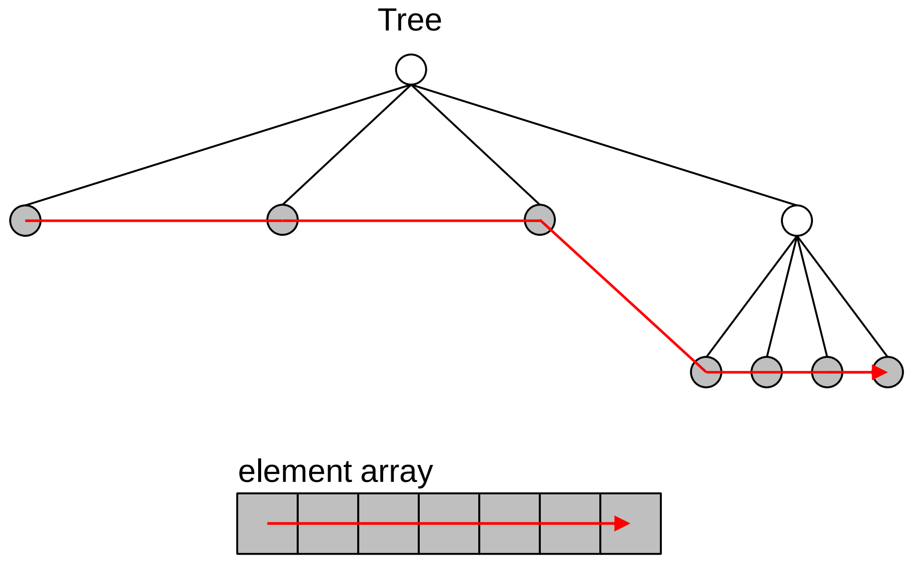


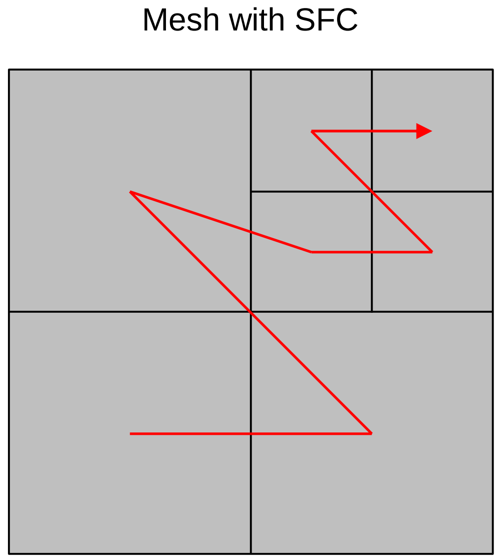
How can we find neighbors in transitioned meshes? Lets find a suitable SFC Index!

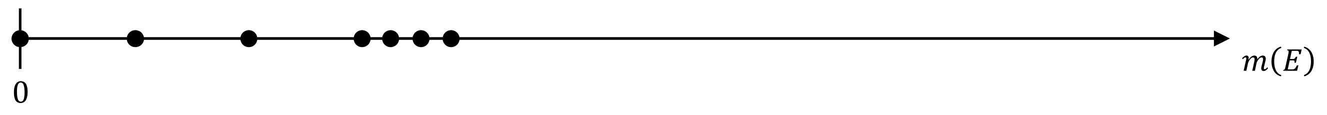
*originally a recursive one-step procedure, proposed by Schneiders [5]

A Space-filling Curve (SFC) Index for transitioned meshes

- t8code uses the **Morton-Index** $m(E) \in \mathbb{N}_0$ to enumerate its elements

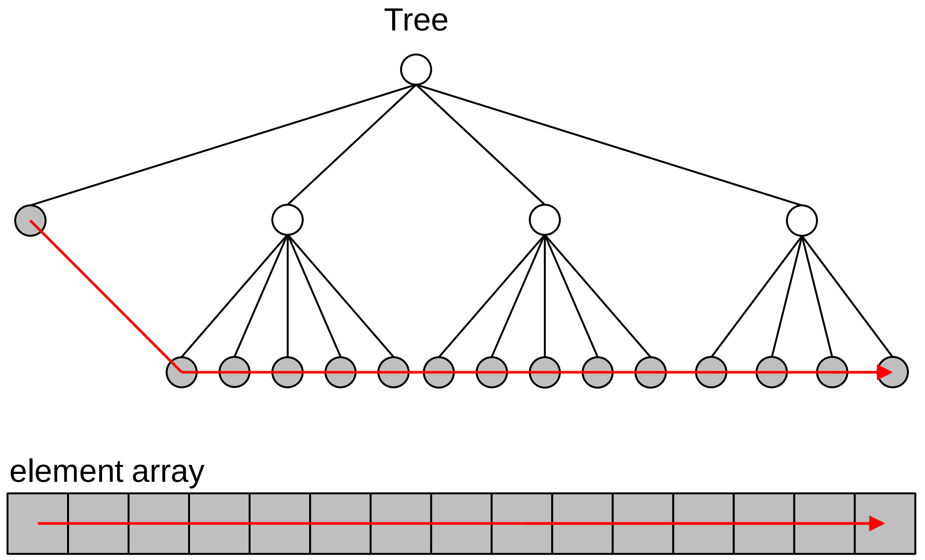


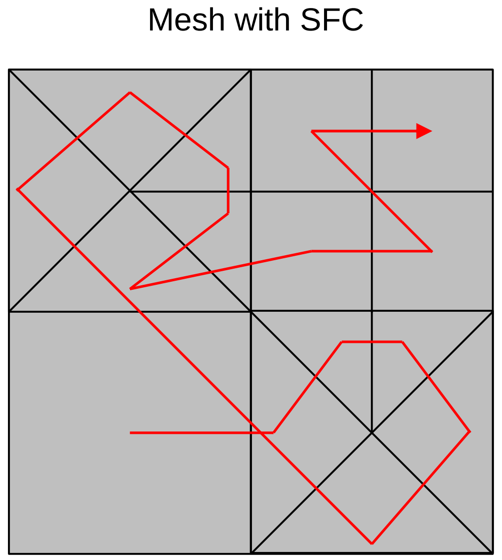


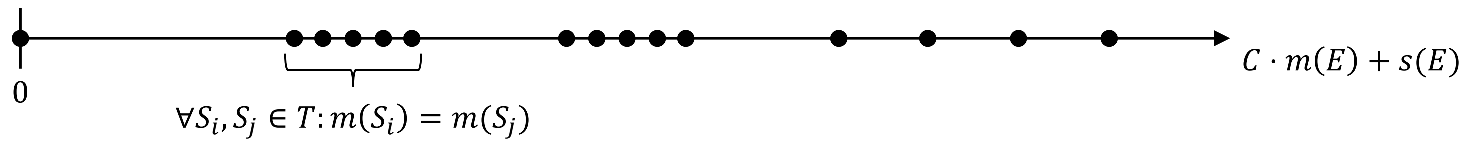


A Space-filling Curve (SFC) Index for transitioned meshes

- In transitioned meshes, we extend the Morton-Index by adding the **Subelement-Index** $s(E) \in \mathbb{N}_0$

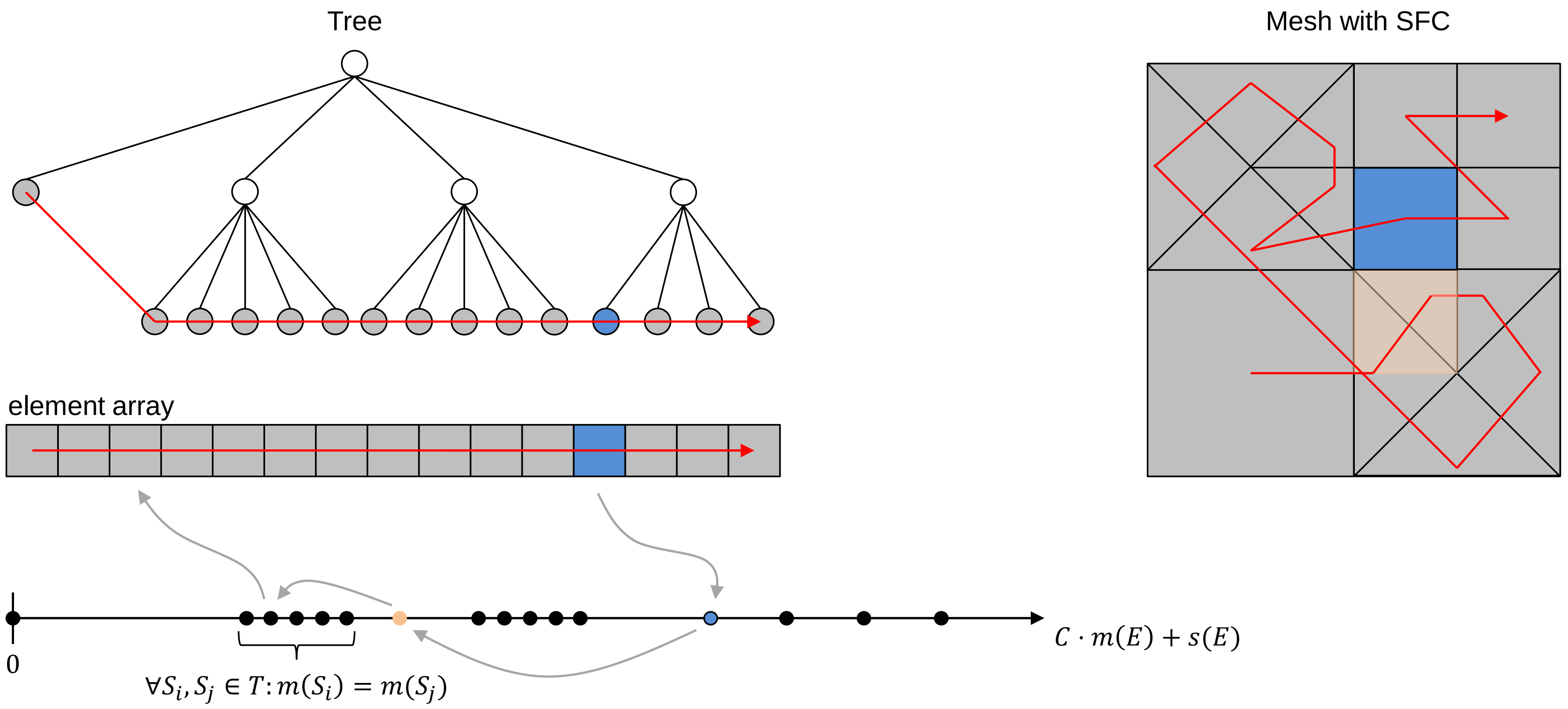






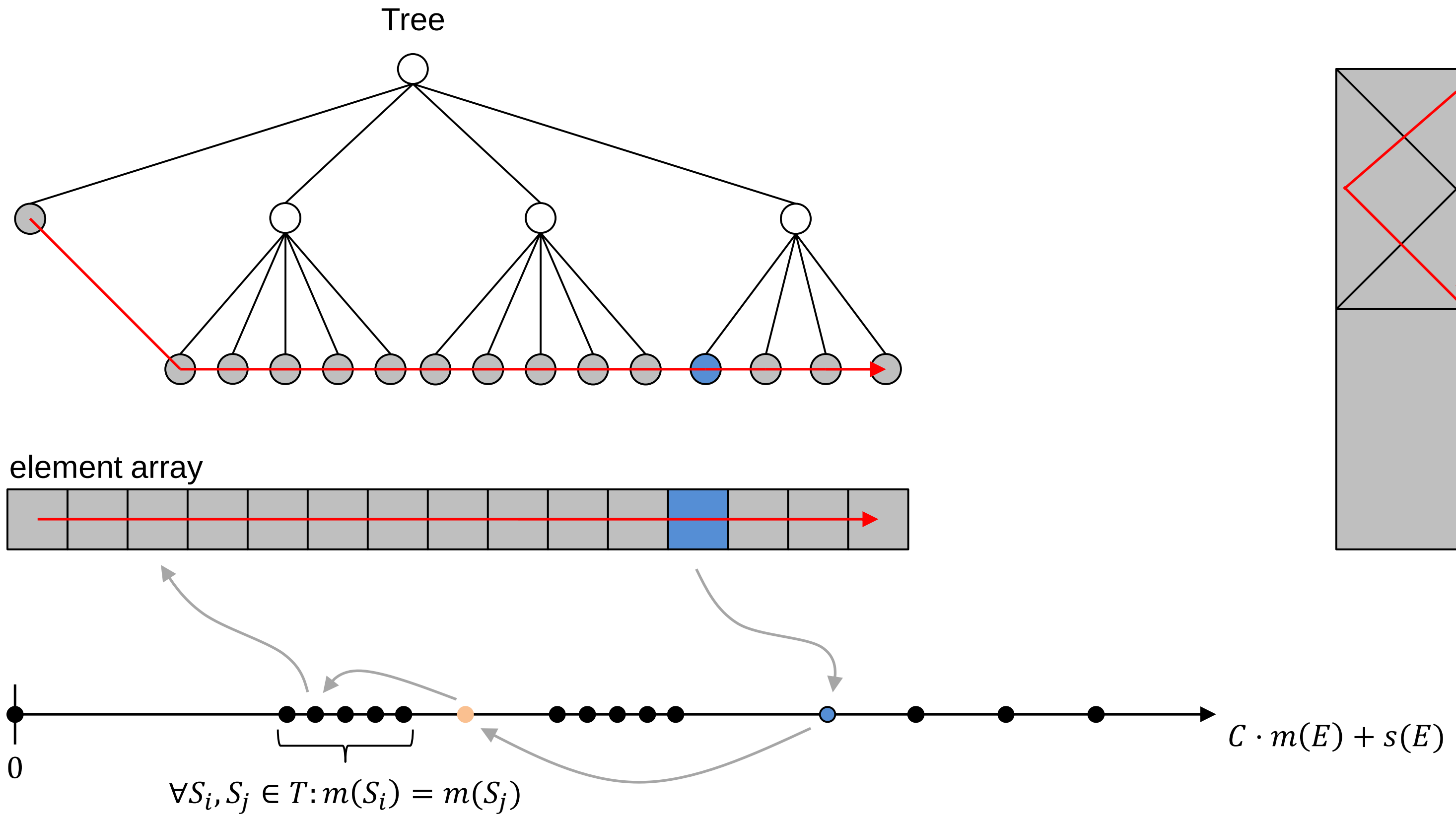
A Space-filling Curve (SFC) Index for transitioned meshes

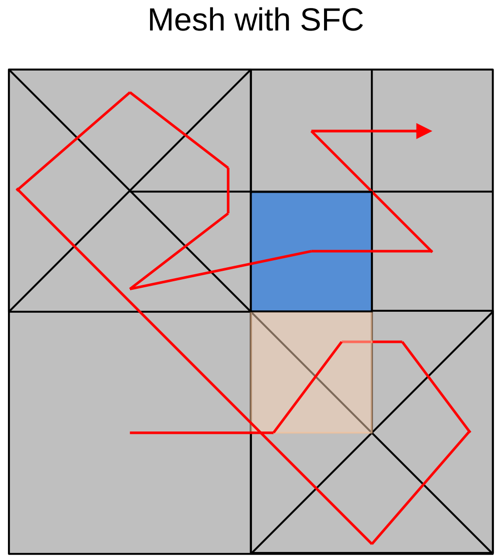
- In transitioned meshes, we extend the Morton-Index by adding the **Subelement-Index** $s(E) \in \mathbb{N}_0$



A Space-filling Curve (SFC) Index for transitioned meshes

- In transitioned meshes, we extend the Morton-Index by adding the **Subelement-Index** $s(E) \in \mathbb{N}_0$



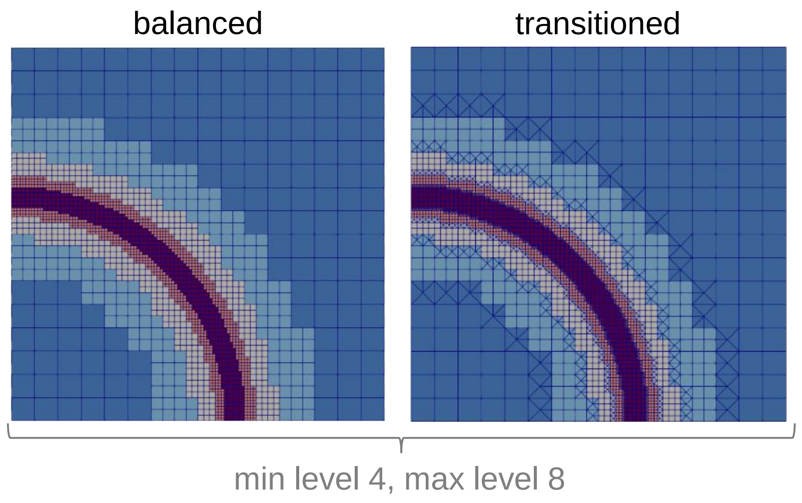


But is this approach efficient?

Benchmarks

Testcase – comparing balanced and transitioned meshes:

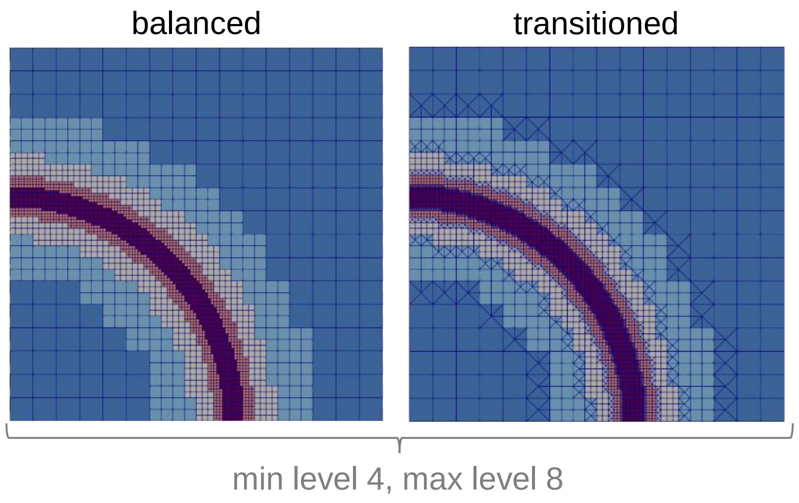
- Construct a balanced mesh and its transitioned version
- Compare the LFN runtime of these meshes



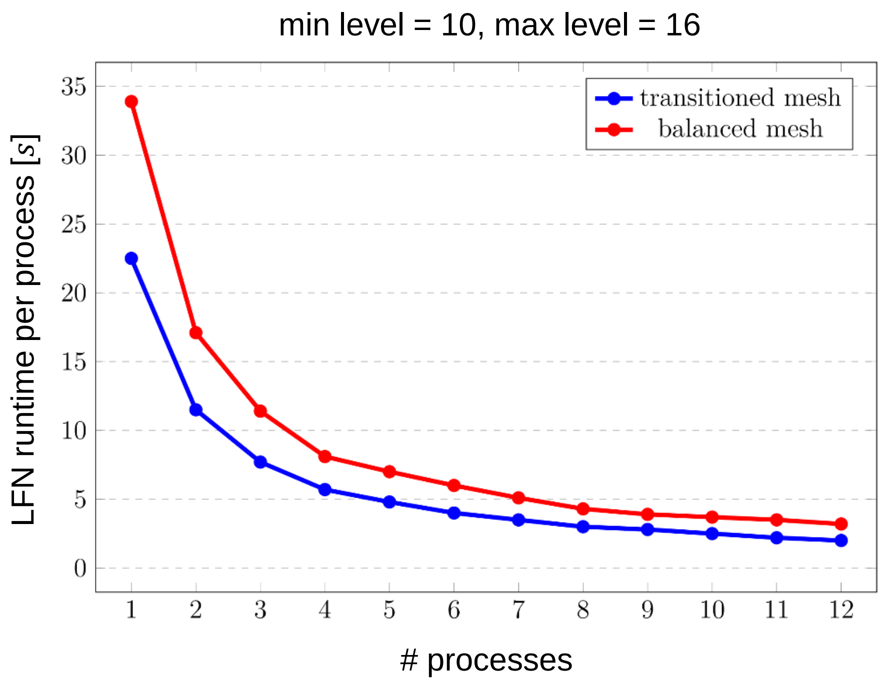
Benchmarks

Testcase – comparing balanced and transitioned meshes:

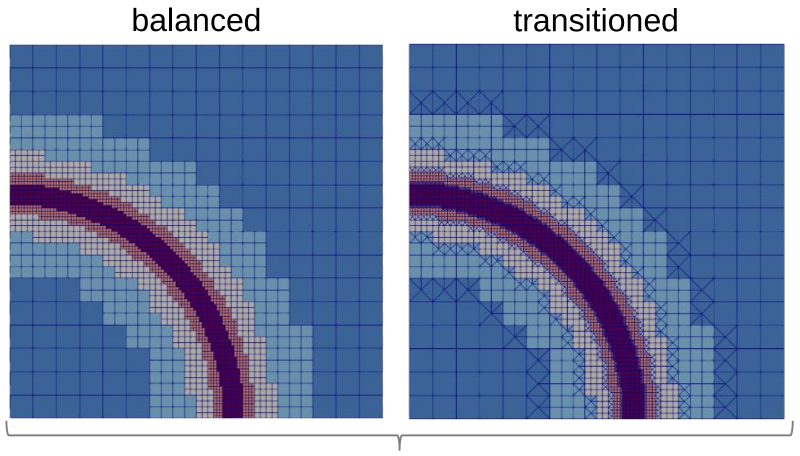
- Construct a balanced mesh and its transitioned version
- Compare the LFN runtime of these meshes



Results:



Benchmarks

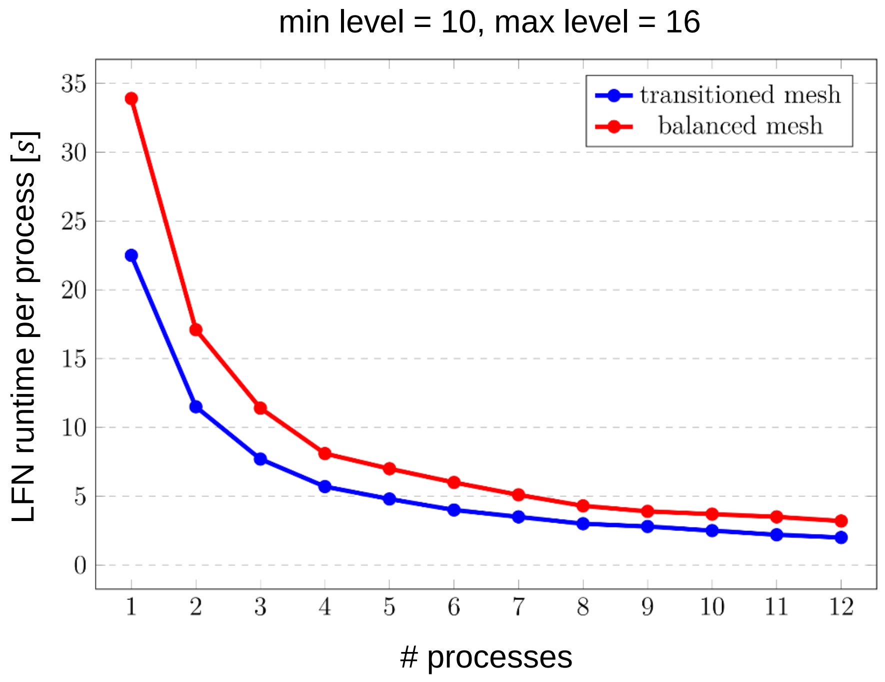


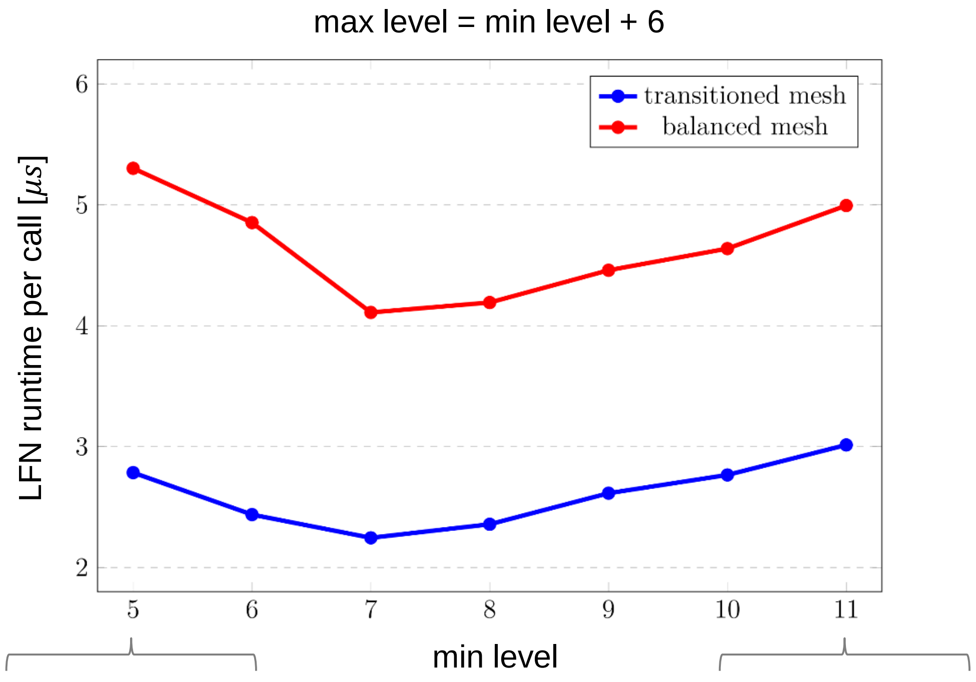
min level 4, max level 8

Testcase – comparing balanced and transitioned meshes:

- Construct a balanced mesh and its transitioned version
- Compare the LFN runtime of these meshes

Results:

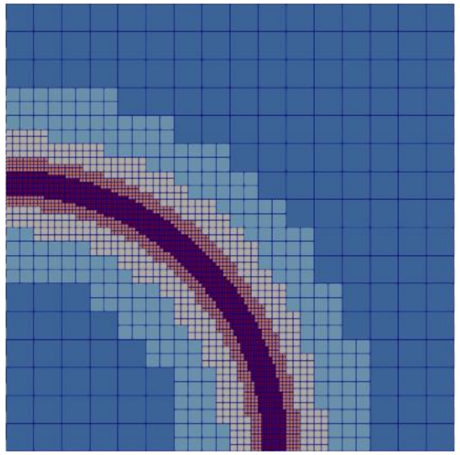


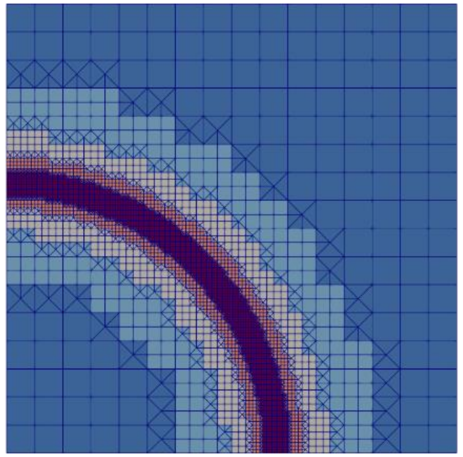


≈50k elements
≈200k LFN calls

≈7m elements
≈ 28m LFN calls

Conclusion and Outlook

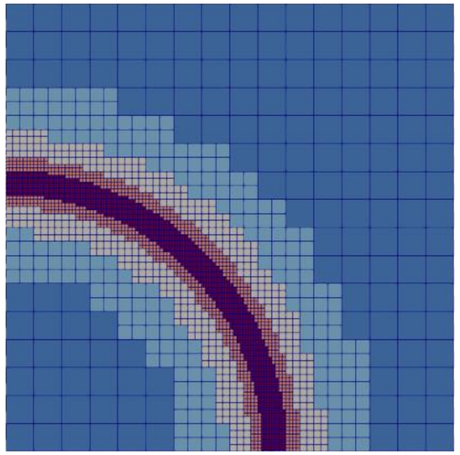


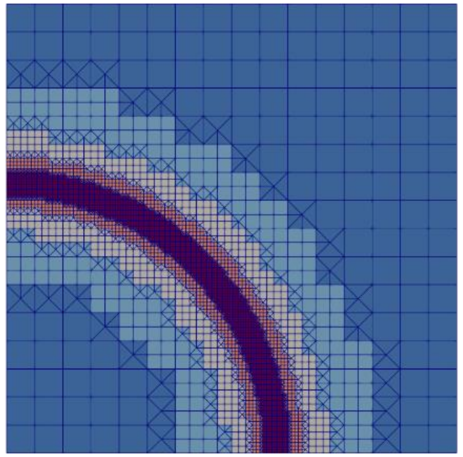


Conclusion:

- The efficiency of the binary search, based on the Morton Index, is not negatively affected by post processing it via the Subelement Index
- The neighbor finding algorithm could even be accelerated by exploiting the conformal property of the transitioned mesh

Conclusion and Outlook





Conclusion:

- The efficiency of the binary search, based on the Morton Index, is not negatively affected by post processing it via the Subelement Index
- The neighbor finding algorithm could even be accelerated by exploiting the conformal property of the transitioned mesh

Outlook/TODOs:

- Extension to 3D?
- Large scale runtime tests
- Merging the new software features into the main code base of t8code

Thank you

Special thanks to Johannes Holke and Gregor Gassner for making this project possible.

Questions?

References

1. Florian Becker. ”Removing hanging faces from tree-based adaptive meshes for numerical simulations”. Master thesis. Universität zu Köln. 2021. https://elib.dlr.de/146849/1/RemovingHangingFacesFromTreeBasedAMR.pdf.
2. Johannes Holke. "Exascale-ready adaptive mesh refinement and applications in Earth system modelling“. 19th Workshop on high performance computing in meteorology. 2021. https://elib.dlr.de/144163/
3. Johannes Holke. “Scalable algorithms for parallel tree-based adaptive mesh refinement with general element types”. PhD thesis. Rheinische Friedrich-Wilhelms Universität Bonn. 2018. https://bonndoc.ulb.uni-bonn.de/xmlui/handle/20.500.11811/7661.
4. Johannes Holke, Carsten Burstedde, et al. Github - holke/t8code: Parallel algorithms and data structures for tree-based amr with arbitrary element shapes.
   https://github.com/holke/t8code.git.
5. Robert Schneiders. "Refining quadrilateral and hexahedral element meshes“. 1996.