Windmillception

Exascale-Ready Adaptive Mesh Refinement

Sandro Elsweijer[12], Johannes Holke[1], David Knapp[1]

[1] Institute for Software Technology, German Aerospace Center (DLR), Cologne, Germany

[2] Institute for Numerical Simulation, University of Bonn, Bonn, Germany

Background and QR generated with png2mesh [3]

Windmill mesh generation

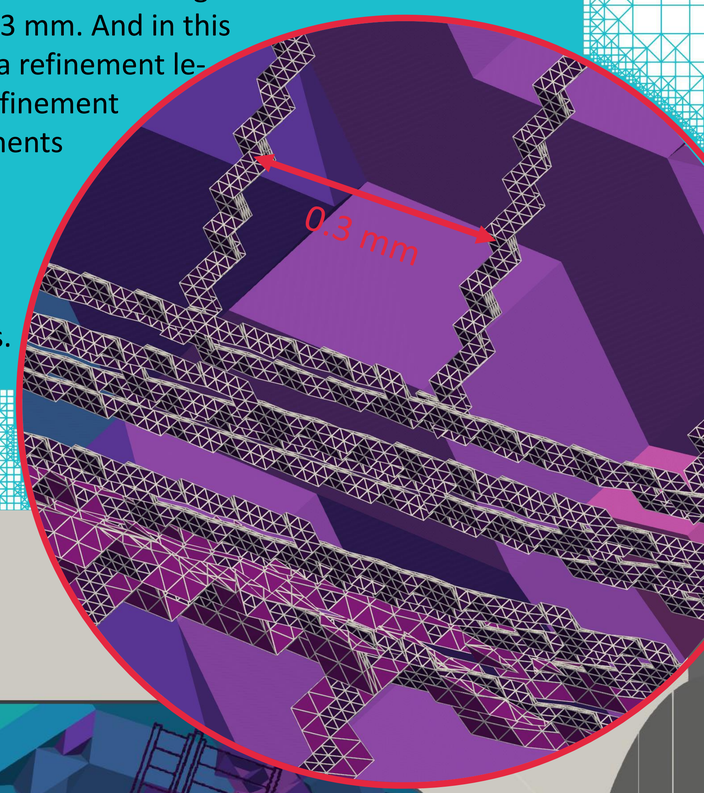
The depicted farfield of the given windmill was generated using Gmsh mesh generator and later on managed and post-processed and refined with t8code.

As a refinement criterion, a 1D mesh of a smaller version of the windmill itself is used. Every element intersecting an edge of the smaller windmill is refined iteratively, up to a final refinement level of 8. Starting with the Gmsh mesh consisting of ~300k coarse tetrahedra, the resulting mesh has ~10m tetrahedra and was refined, balanced and partitioned in 65 seconds on 160 processes.

Finally, the mesh is visualized in ParaView. While the unrefined cells (level 0), visualized in cyan, are clearly visible on the outside of the windmill, a peek through the door shows the different levels generated by the refinement with the smaller windmill inside the larger one.

Element sizes

While the coarsest elements are around 3.3 m large, the finest elements are by more than three magnitudes smaller and have a diameter of around 13 mm. And in this example we only used a refinement level of 8. Possible are refinement levels of 21 for 3D Elements and even more for 2D. This is achieved by the efficient management with refinement trees and space-filling curves.



Tree-based Adaptive Mesh Refinement

Tree-based Adaptive Mesh Refinement (AMR) is a powerful technique in computational science and engineering that enables efficient and accurate simulations on complex geometries. The t8code (“tetcode”) library [1] provides a high-level interface for implementing AMR in large-scale simulations, using the forest-of-trees approach.

In the forest-of-trees approach, multiple tree structures are used to represent different components of the computational domain, allowing for improved flexibility and scalability in the mesh resolution. t8code supports different element shapes, including hexahedra, tetrahedra, prisms, and pyramids and can be used for example for FVM or DG simulations.

Tested with over one trillion (~1.1e12) elements on around 50k processors [2], you can profit from the exceptional scalability and memory efficiency and make your first step into the exascale era.

Resolving of hanging nodes is coming soon [4]

References

[1] Holke, Johannes, Burstedde, Carsten, Knapp, David, Dreyer, Lukas, Elsweijer, Sandro, Uenlue, Veli, Markert, Johannes, Lilikakis, Ioannis, & Boeing, Niklas. (2022). t8code (1.0.0). doi: 10.5281/zenodo.7104580

[2] Holke, Johannes, Knapp, David, & Burstedde, Carsten. (2021). An Optimized, Parallel Computation of the Ghost Layer for Adaptive Hybrid Forest Meshes. SIAM Journal on Scientific Computing, 43 (6), 359-385. SIAM - Society for Industrial and Applied Mathematics. doi: 10.1137/20M1383033

[3] Holke, Johannes, & Knapp, David. (2023). Png2mesh. https://github.com/DLR-AMR/png2mesh

[4] Becker, Florian. (2021). Removing hanging faces from tree-based adaptive meshes for numerical simulations. Master thesis, Universität zu Köln. https://elib.dlr.de/187499/