t8code - scalable and modular adaptive mesh refinement

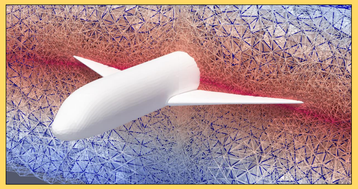
Johannes Holke [1], Carsten Burstedde [2], David Knapp [1], Lukas Dreyer [1], Sandro Elsweijer [1,2], Veli Ünlü [1],
Johannes Markert [1], Ioannis Lilikakis [1], Niklas Böing [1], Prasanna Ponnusamy [1], Chiara Marie Hergl [1], Achim Basermann [1]
[1] Institute for Software Technology, German Aerospace Center (DLR), Cologne, Germany
[2] University of Bonn, Germany

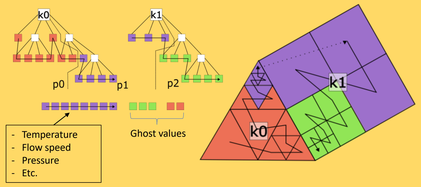
Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

t8code

t8code (www.dlr-amr.github.io/t8code) is an open source C/C++ library for scalable adaptive mesh refinement (AMR). We support parallel adaptive mesh and data management including the following tasks:

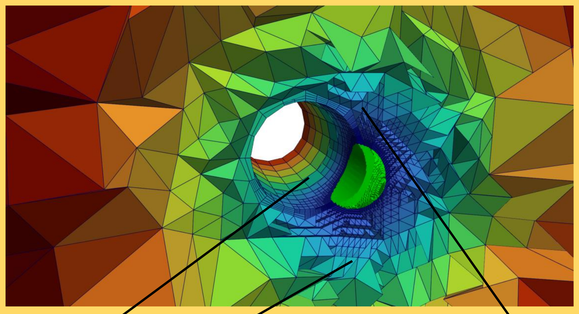
- Adapting meshes (refinement/coarsening)
- Load-balancing meshes and data
- Creating Ghost/Halo elements
- Exchanging data via Ghosts
- 2:1 balancing
- Interpolating between meshes
- Searching for features
- Curved and high-order meshes [6]

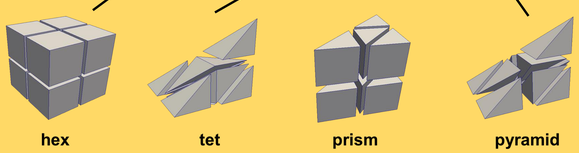

An airplane embedded in an adaptively refined tetrahedral mesh



The SFC data layout of t8code. An application code provides its data in array form.

t8code uses efficient **space-filling curves** (SFC) in a modular fashion such that **arbitrary element shapes** are supported. This is achieved by decoupling high-level (mesh global) from low-level (element local) algorithms.


A 3D hybrid curved mesh with hexahedra, pyramids and tetrahedra. Simulated is the flow of a bubble in liquid around a circular obstacle.

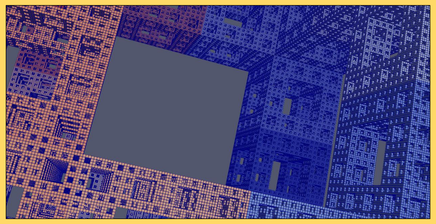


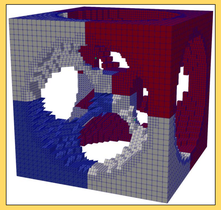
The subdivision of all standard 3D elements supported by t8code by default.
All use a morton-type SFC.

Non-standard SFC techniques

Cutting holes

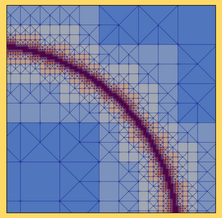
By deleting elements from the refinement tree, we support **holes in meshes**, for examples around embedded geometries. [4]





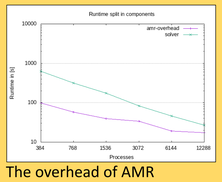
Subelements

Elements can be arbitrarily subdivided.
One application is **resolving hanging nodes** in non-conforming meshes. [5]



Performance

t8code scales up to at least **one million parallel processes** and **over one trillion mesh elements**. [1,2]
Simulations with a high-order discontinuous Galerkin solver show that the AMR overhead is between 10% and 25% while significantly reducing the DOFs and runtime.[3]

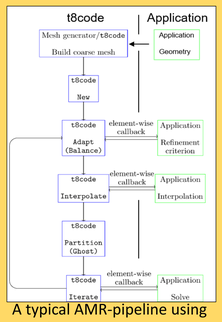


The overhead of AMR

	Runtime	Error	#DOFs
Uniform 3D	7057s	1.3e-3	16.777.216
Adaptive 3D	561s	1.5e-3	~1.920.000

Interfacing with t8code

t8code applies the **Hollywood principle**:
"Don't call us, we'll call you!".
Whenever an application needs to interact with the mesh (adapting, interpolating, etc.), we offer **callback handlers**. The standard set of high-level algorithms for a simulation code is: **New, Adapt, Balance, Interpolate, Partition, Ghost, Iterate**



A typical AMR-pipeline using t8code

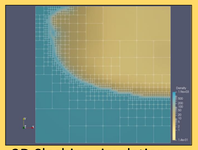
Selected Projects using t8code

ADAPTEX

t8code will be the meshing backend of a DG based exascale-ready CFD framework with focus on Earth System Modelling applications such as atmospheric chemistry, climate and flooding simulations.


Tracking an ash cloud of a volcanic eruption

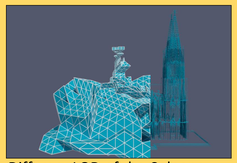
HYTAZER


2D Sloshing simulation

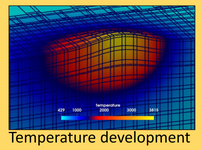
The goal is to develop solutions for the operation of **liquid hydrogen tanks** under safety standards for vehicles. We implement simulations of sloshing effects based on **high-fidelity CFD methods**, which require a **high resolution** between liquid and gaseous phase.

VISPLORE

VISPLORE uses t8code to enable the visualization of data at different **levels of detail.** Starting on a coarse level we **interpolate** the data on the elements and give the user a first look at its data. The initial visualization can then be adapted to the desired level of detail.


Different LOD of the Cologne Cathedral

PADME-AM

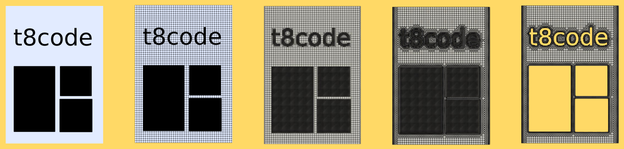

Temperature development during laser melting
©Frauenhofer SCAI

PADME-AM aims to simulate **Selective Laser Melting** efficiently by combining t8code and **Partition of Unity Methods** (PUM) developed by the **Fraunhofer SCAI** and the **University of Bonn**. Despite being a meshfree method, the PUM can profit from t8code's data storage and parallelization on thousands of processes.

Pipeline of the Postergrid

The mesh was generated using png2mesh: https://github.com/DLR-AMR/png2mesh

Step 1: Make the Poster | Step 2: Embed Poster in a Mesh | Step 3: Refine at dark pixels | Step 4: Resolve hanging nodes | Step 5: Remove elements



Can you find the space-filling curve in the poster?

References
[1] Holke J., Knapp D., Burstedde C. An Optimized, Parallel Computation of the Ghost Layer for Adaptive Hybrid Forest Meshes. SIAM Journal on Scientific Computing, pp. C359–C385, Jan. 2021. URL https://epubs.siam.org/doi/abs/10.1137/20M138303
[2] Burstedde C., Holke J. *A tetrahedral space-filling curve for nonconforming adaptive meshes.* SIAM Journal on Scientific Computing, vol. 38, C471–C503, 2016
[3] Dreyer L. *The local discontinuous galerkin method for the advection-diffusion equation on adaptive meshes*. Master's thesis, Rheinische Friedrich-Wilhems-Universität Bonn, Februar 2021. URL https://elib.dlr.de/143969/
[4] Lilikakis I., Algorithms for tree-based adaptive meshes with incomplete trees, Master's thesis, Universität zu Köln, 2022, URL https://elib.dlr.de/191968/
[5] Becker F., Removing hanging faces from tree-based adaptive meshes for numerical simulations, Universität zu Köln, 2021, URL https://elib.dlr.de/187499/
[6] Elsweijer S., Evaluation and generic application scenarios for curved hexahedral adaptive mesh refinement, Hochschule Bonn-Rhein-Sieg, 2022, URL https://elib.dlr.de/186561/
[7] Knapp D., A space-filling curve for pyramidal adaptive mesh refinement, Master's thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2020, URL https://elib.dlr.de/139134/
Colors created using Colorgorical:
[8] Gramazio C., Laidlaw D. Schloss K. Colorgorical: creating discriminable and preferable color palettes for information visualization, 2017, IEEE Transactions on Visualization and Computer Graphics

t8code @ CSE23: Holke J., Markert J., The Power of Modular Tree-Based AMR -- Resolving Hanging Nodes and Cutting Holes
t8code @ IMR23: Holke J. et al, t8code v1.0 – Modular Adaptive Mesh Refinement In The Exascale Era, Research Note
Elsweijer S. et al, Windmillception – Exascale-Ready Adaptive Mesh Reinement, Meshing Contest