The Journal of Open Source Software

Dune-MMesh: The Dune Grid Module for Moving Interfaces

Samuel Burbulla[*1], **Andreas Dedner**[2], **Maximilian Hörl**[1], **and Christian Rohde**[1]

1 University of Stuttgart **2** University of Warwick

DOI: 10.21105/joss.03959

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @krober10nd
- @vijaysm

Submitted: 25 November 2021
Published: 29 November 2021

Summary

Dune-MMesh is an implementation of the Dune (Bastian et al., 2021) grid interface that is tailored for numerical applications with possibly moving physical interfaces. The implementation based on CGAL triangulations (The CGAL Project, 2020) supports two and three dimensional meshes and can export a predefined set of facets as a separate interface grid. In spatial dimension two, arbitrary movement of vertices is enhanced with a re-meshing algorithm that implements non-hierarchical adaptation procedures. Various examples based on the python bindings of the discretization module dune-fem (Dedner et al., 2020) have been implemented that demonstrate the versatile applicability of Dune-MMesh.

Statement of need

In many technical applications, in particular in the field of fluid dynamics, comparably thin physical interfaces can have a large impact on the overall behavior of a modeled system. Interfaces occur as separating layer between fluid phases in multiphase flows, in fluid-structure interaction, fluid-solid phase change and even fractures are modeled by lower-dimensional surfaces.

The grid implementation Dune-MMesh aims at providing numerical capabilities for grid based methods to model interface-driven processes within the Dune framework. Essentially, it consists of two things: First, a triangulation based on CGAL where a set of facets is considered as interface. And, second, the possibility to re-mesh the triangulation when necessary. The representation of some grid facets as interface makes Dune-MMesh a useful tool for the implementation of mixed-dimensional models. The inevitable non-hierarchical adaptation complements the existing grid implementations within the Dune framework and allows for unprecedented flexibility of grid adaptation.

CGAL Wrapper

In its core, Dune-MMesh is a wrapper of CGAL Triangulations in $\mathbb{R}^d, d = 2, 3$, that implements the Dune grid interface. A CGAL triangulation is a set of simplicial cells and vertices where each cell gives access to its $d+1$ incident vertices and cells. Facets are not explicitly represented: a facet is given by the pair of a cell c and an index i and has two implicit representations. For $d = 3$, edges are represented by triples of a cell c and two indices i and j that indicate the two vertices of the edge.

[*] corresponding author

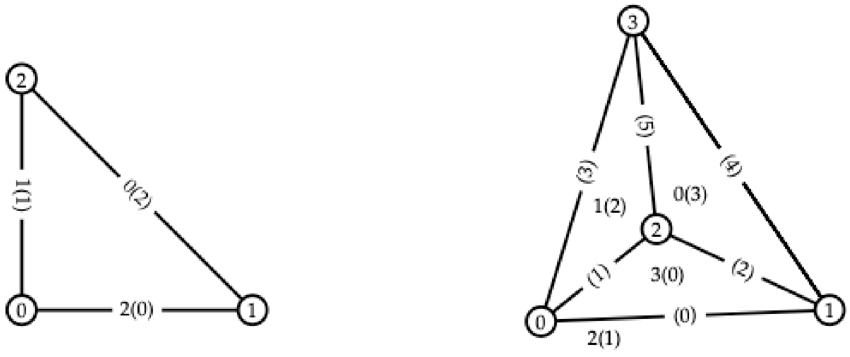

Figure 1: CGAL representation of cells and differing Dune numbering in brackets.

In order to match the Dune grid reference cell numbering we apply an index mapping, cf. Figure 1. Dune intersections can directly be represented by CGAL's cell-index representations of facets which are already equipped with an orientation. The index and id sets of the Dune grid interface are realized by consecutive numbering of cells and vertices. Various iterators of CGAL triangulations can directly be used to construct the Dune grid range generators. Additional (non-standard Dune) iterators have been added, e.g. iterating over incident cells of a vertex.

Interface Grid

Consider a domain $\Omega \subset \mathbb{R}^d, d \in \{2,3\}$, that includes a $(d-1)$-dimensional interface $\Gamma \subset \Omega$, as depicted in Figure 2. We assume the domain is triangulated conforming to the interface Γ.

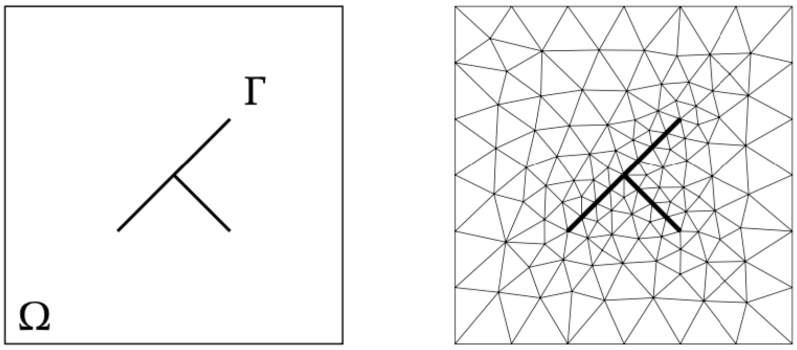

Figure 2: A domain with a T-shaped interface and an example for a conforming triangulation.

Dune-MMesh features a second implementation of the Dune grid interface that represents the interface triangulation. Here, a codim-0 entity of the interface grid is represented by a CGAL cell-index pair. The interface grid also supports networks, cf. Figure 3, and it is possible to convert bulk intersections to interface grid cells and vice versa.

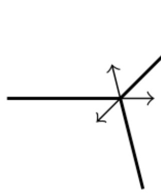

Figure 3: Outer normals at junctions.

Moving Mesh

Most interface driven-problems have time-dependent interfaces $\Gamma = \Gamma(t)$. Therefore, Dune-MMesh features capabilities of moving and re-meshing in spatial dimension two.

Moving Vertices

We assume that movement is given by a shift of interface vertices (or all grid vertices), cf. Figure 4 (left).

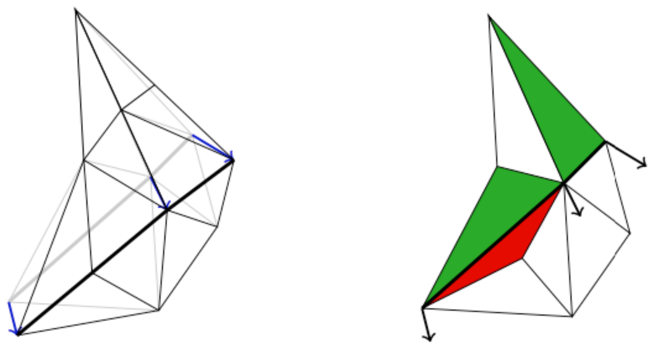

Figure 4: Left: Moving the interface. Right: Marking cells for refinement (green) or coarsening (red).

To prevent degeneration of the triangulation, i.e. cells have non-positive volume, Dune-MMesh is equipped with re-meshing routines.

Adaptation

Adaption in Dune is usually hierarchical by definition and the adaptation procedure is performed in two stages:

1. Mark: Grid cells are marked for coarsening or refinement.
2. Adapt: The cells are modified due to their markers and discrete functions are restricted or prolongated.

In Dune-MMesh, due to the moving mesh, non-hierarchic adaptation is inavoidable. However, we will try to follow the general Dune approach and separate the adaptation into two stages.

1. Mark

Dune-MMesh provides utility functions to mark cells either in expectation of a movement of vertices or regarding to their current geometrical properties, cf. Figure 4 (right). However, one can also use a proprietary procedure marking cells manually, or one can insert and remove vertices directly.

2. Adapt

After marking cells an adapt routine performs the actual adaptation process. The adaptation is performed by insertion and removal of points.

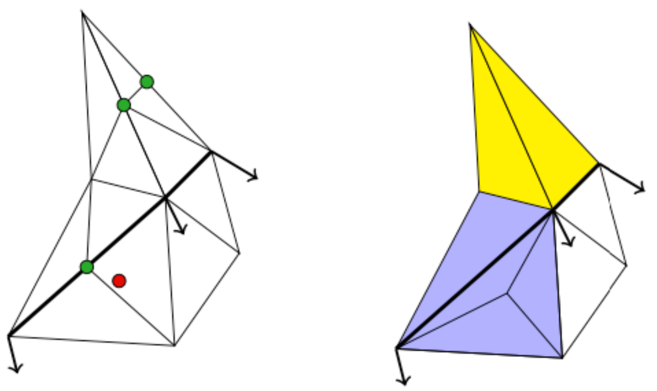

Figure 5: Left: Inserting and removing points. Right: Connected components.

In each cell that is marked for refinement we bisect the longest edge, cf. Figure 5 (left). In all cells marked for coarsening, the least important vertex is removed. When a vertex is removed, the resulting star-shaped hole is re-triangulated with respect to the interface.

For the purpose of projection, we introduce *connected components*, see Figure 5 (right), and implement a generalized callback adaptation in dune-fem.

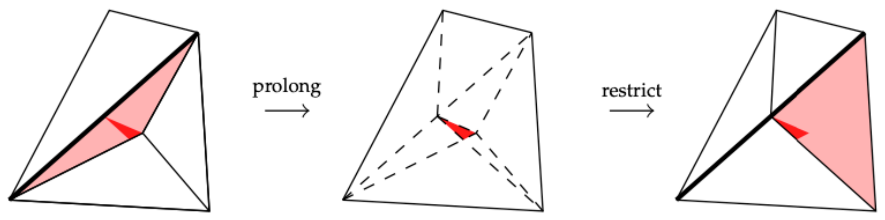

Figure 6: Non-hierarchic projection with cut-set triangulation.

A conservative projection of discrete functions can be performed by intermediate prolongation and restriction on the cut-set cells, cf. Figure 6. We use a similar concept on the interface grid that enables projection of discrete functions on the interface.

Trace and skeleton

Dune-MMesh exports both traces of bulk discrete functions on the interface and skeleton representations of interface discrete functions on bulk edges.

The trace is a discrete function on the interface grid that evaluates a given bulk discrete function. It can be restricted to both sides of the interface and might be used in UFL forms.

Analogously, the skeleton function is a discrete function that returns the interface's discrete function values on interface bulk facets.

Both `trace` and `skeleton` can be used to couple bulk and interface problems. Such couplings occur, e.g., in mixed-dimensional PDEs.

Coupled solve

We provide two helper functions to solve bulk and interface schemes in a coupled way.

The first method `iterativeSolve` uses an iterative solution strategy which alternately solves both schemes until the two norm between two iterates is below an objective tolerance.

The second helper function `monolithicSolve` solves bulk and interface scheme coupled monolithically. A newton method is implemented assembling the underlying jacobian matrix where the coupling jacobian blocks are evaluated by finite differences.

Examples

We implemented a few examples to display how Dune-MMesh can be used in different contexts. All examples can be found in `dune-mmesh/doc/examples` as IPython notebooks. Some numerical results of these examples are visualized in Figure 7, Figure 8 and Figure 9.

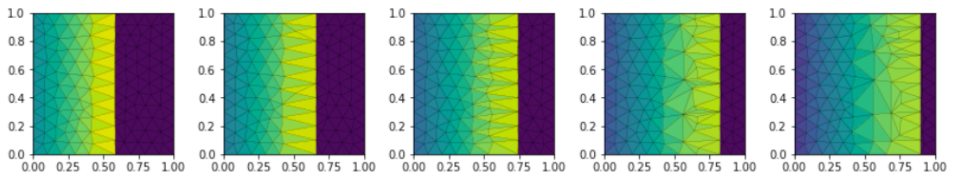

Figure 7: Finite volume moving mesh method to track a discontinuity (Chalons et al., 2018)

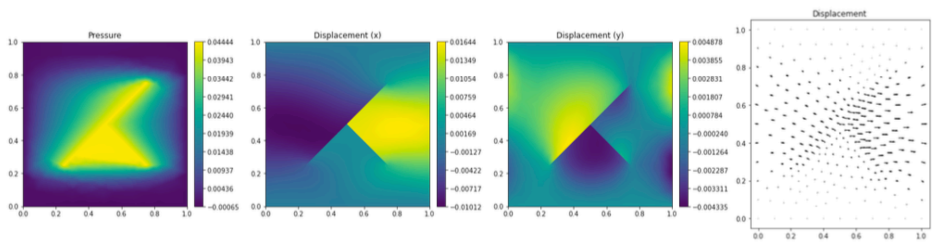

Figure 8: Mixed-dimensional model of poro-elasticity with a T-shaped fracture.

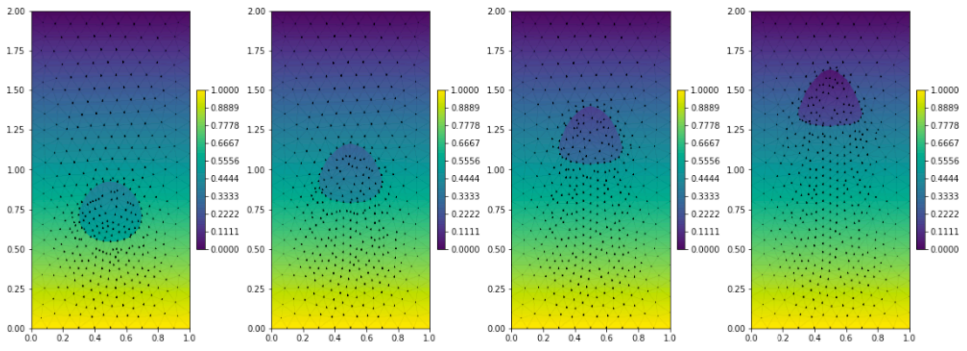

Figure 9: Two-phase Navier-Stokes equation (Gerstenberger et al., 2020).

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number 327154368 - SFB 1313.

We thank all contributors that improved Dune-MMesh via the GitLab repository, especially Timo Koch.

References

Bastian, P., Blatt, M., Dedner, A., Dreier, N.-A., Engwer, C., Fritze, R., Gräser, C., Grüninger, C., Kempf, D., Klöfkorn, R., Ohlberger, M., & Sander, O. (2021). The DUNE framework: Basic concepts and recent developments. *Computers & Mathematics with Applications*, *81*, 75–112.

Chalons, C., Magiera, J., Rohde, C., & Wiebe, M. (2018). A finite-volume tracking scheme for two-phase compressible flow. *Theory and Numerics and Applications of Hyperbolic Problems I*, 309–322.

Dedner, A., Nolte, M., & Klöfkorn, R. (2020). Python bindings for the DUNE-FEM module. *Zenodo*. https://doi.org/10.5281/zenodo.3706994

Gerstenberger, J., Burbulla, S., & Kröner, D. (2020). Discontinuous galerkin method for incompressible two-phase flows. *Finite Volumes for Complex Applications IX - Methods and Theoretical Aspects and Examples*, 675–683.

The CGAL Project. (2020). CGAL user and reference manual. *CGAL Editorial Board, 5.2 edition*.