Computers & Geosciences 99 (2017) 159–170

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

HULK – Simple and fast generation of structured hexahedral meshes for improved subsurface simulations

Gunnar Jansen*, Reza Sohrabi, Stephen A. Miller

CHYN – Centre for Hydrogeology and Geothermics, Laboratory of Geothermics and Geodynamics, University of Neuchâtel, Switzerland

ARTICLE INFO

Keywords:
Hexahedral mesh
Binary space partitioning
Octree refinement
Geological model
FEM/FVM/FDM
STereoLithography

ABSTRACT

Short for Hexahedra from Unique Location in (K)convex Polyhedra – *HULK* is a simple and efficient algorithm to generate hexahedral meshes from generic STL files describing a geological model to be used in simulation tools based on the finite element, finite volume or finite difference methods. Using binary space partitioning of the input geometry and octree refinement on the grid, a successive increase in accuracy of the mesh is achieved. We present the theoretical basis as well as the implementation procedure with three geological models with varying complexity, providing the basis on which the algorithm is evaluated. *HULK* generates high accuracy discretizations with cell counts suitable for state-of-the-art subsurface simulators and provides a new method for hexahedral mesh generation in geological settings.

1. Introduction

A geological model should incorporate structural information and rock properties for any kind of subsurface simulation because simulation accuracy strongly depends on the relevant rock properties and their distribution in space. Therefore, reliable results can only be expected when well-constrained structural and lithological information is used in the simulation. Due to complexities in both the geological modeling and subsurface simulation, an integrated approach of modeling the geology and the physics of the subsurface (e.g. flow, deformation, etc.) is in many cases not available. Commercial software exists that provides integrated modeling and simulation, especially in the oil/gas industry, but often requires months of expert training and their usage might be unsuited for academic purposes. Unfortunately most researchers are only expert in a single field. Thus, they might be either experts in geological modeling or very good in simulators, but rarely both. Consequently, this often leads the situations where very detailed geological models are never used to simulate physics, while simulators with highly sophisticated physics are often only applied to simple and over-simplified geologies. We address this problem for simulators using hexahedral grids by proposing an efficient mesh generation method. The method is based on octree refinement and provides for direct transfer of structural geological information to the numerical simulator of the underlying physics.

1.1. Discretization of geological models

Most meshing algorithms focus on aligning the mesh exactly with the boundaries of the structural geological model. This is inevitable in some cases, such as the prototyping of engineering parts. In other cases, such as structural geological models, this requirement of a so-called *conforming* mesh is too strict and not always necessary because of the uncertainties associated with the geological model itself (Bárdossy and Fodor, 2001; Thore et al., 2002; Lindsay et al., 2012; Wellmann et al., 2014). A well-discretized geological model therefore does not necessarily follow the exact boundaries of the geological model, but more importantly enables the simulator to capture the most relevant aspects and to properly model the underlying physics.

Structural geological models can be built using several different methods if the amount of data (e.g. geological, geophysical or seismic surveys) is sufficient and are readily implemented in widely used commercial software packages *Geomodeller (Intrepid-Geophysics), GOCAD (Paradigm)* or *Petrel (Schlumberger)*. In cases where the geological information available is sparse, solely the interpretation of the geologist might be used to build a structural model using a computer aided design (CAD) program. Of course, with less data available, the accuracy of the structural model and thus ultimately of the subsurface simulation is limited.

The type of discretization requested by the wide variety of subsurface simulators varies largely due to the different underlying numerical methods that impose different restrictions on the geological model's discretization. Finite element simulators are often based on triangular

* Corresponding author.
E-mail addresses: gunnar.jansen@unine.ch (G. Jansen), reza.sohrabi@unine.ch (R. Sohrabi), stephen.miller@unine.ch (S.A. Miller).

http://dx.doi.org/10.1016/j.cageo.2016.11.011
Received 11 April 2016; Received in revised form 18 July 2016; Accepted 22 November 2016
Available online 27 November 2016
0098-3004/

or tetrahedral meshes, while many finite volume based tools require quadrilateral or hexahedral meshes (Pruess et al., 1999; Hammond et al., 2012; Jasak et al., 2007; Trefry and Muffels, 2007; Li et al., 2009). Automatic high quality mesh generation software is available for triangular and tetrahedral elements using Delaunay triangulation (Ho-Le, 1988; Chan and Anastasiou, 1997; Du and Wang, 2006). However, many simulators exist, especially in the field of computational fluid dynamics in general and in high performance computing, that are based on quadrilateral discretizations. Unfortunately, where a quadrilateral/hexahedral mesh is required or beneficial, a general and stable automatic solution method for the generation of a conforming mesh is not available and a field of ongoing research (Botella et al., 2014; Yu et al., 2015; Owen and Shelton). Significant progress was made in recent years and also applied to geological models (e.g. Xing et al., 2009; Zehner et al., 2015; Botella et al., 2016). However, these methods share a certain complexity in terms of automatism and are rather time-consuming (Stupazzini, 2004; Casarotti et al., 2008).

The methods previously described (including most methods from tetrahedral meshing) have their “bottom-up” approach in common. First, two dimensional surfaces based on the vertices on the geological formation boundaries are meshed by quadrilateral elements. The model is then divided into different meshable regions, which are meshed separately by one of the methods in hexahedral meshing such as sweeping or the advancing front technique (Scott et al., 2005; Löhner, 1996). These separated volumes are connected in a last step to a complete discretization.

As outlined previously, a conforming discretization as targeted by these algorithms is not completely necessary. Octree-based meshing algorithms are generally considered very efficient if a good but non-conforming discretization is targeted. Octree methods were introduced in Yerry and Shephard (1984) and further enhanced by various authors (e.g. Schneiders, 1996, 1997; Shephard and Georges, 1991; Maréchal, 2001, 2009; Tu and O'Hallaron, 2004). In general, octree based discretizations have good mesh quality, are adaptive and support automatic generation (Xing et al., 2009). The Los Alamos Grid Tool Box (LaGriT) is specifically designed to generate meshes for geological applications and also supports meshing procedures based on octree refinement. Using the octree refinement method, Miller et al. (2007) used LaGriT to discretize a geological model that demonstrated good performance.

1.2. Outline and scope

The purpose of this work is to present a new and simple-to-use tool to reduce the gap between numerical simulators based on structured or octree grids and geological modeling. Short for Hexahedra from Unique Location in (K)convex Polyhedra – *HULK* is a simple and efficient algorithm to generate hexahedral meshes from geological models. We achieve rapid mesh generation by binary space partitioning and the octree refinement. While the method of octree refinement is not enhanced in this work, we present an alternative approach for handling all geological formations at the same time, classifying their position in space efficiently. We therefore utilize a “top-down” approach in our method, in contrast to most of the meshing procedures mentioned previously.

In the following, we focus on the underlying theory of the implemented algorithm. We first describe the discretization procedure, followed by the theory of binary space partitioning that is essential for method efficiency. We briefly introduce the methods to evaluate its performance and provide insight into the implementation. The *HULK* algorithm is then applied to a series of test cases, including an irregularly shaped two component system, a sedimentary basin, and a complex example from the Swiss Jura Mountain range. We then evaluate the method's overall performance (e.g. different input mesh sizes for the same model), and conclude with some perspectives for future research.

2. Theory

A fundamental target of geological modeling is to identify the boundaries between geological formations in the subsurface. Consequently, the output of any geological modeler is in most cases a discretized (triangulated) boundary volume representation of the formations. This discretized boundary representation can be used as the input for the meshing preprocessor of the subsurface simulator. We use this representation to generate a binary space partitioning and successively refine a hexahedral background mesh as further described in this section.

2.1. Octree-based meshing

The grid generation process in *HULK* is based on the octree refinement technique (Yerry and Shephard, 1984; Schneiders, 1996, 1997; Shephard and Georges, 1991; Maréchal, 2001, 2009; Tu and O'Hallaron, 2004). Initially, a structured background mesh is generated circumscribing all of the volumes to be discretized. In the next step, each cell is associated with the corresponding geological formation. In *HULK* this cell-to-formation mapping is performed by binary space partitioning of the input volumes, discussed in detail below. Once the correct position within the input model is determined, cells can be flagged for refinement (or coarsening) based on their affiliation in a third step. Several refinement criteria can be used. Here we chose to flag all cells whose neighbors have non-matching material identifiers for refinement, and cells which have only matching-id neighbors for coarsening (Fig. 1). The refinement is then executed according to the flags that were previously set. If these two steps are repeated until a desired accuracy or number of mesh cells has been generated, it forces a cyclic refinement towards the boundaries of the geological formations (cf. Fig. 2). The actual implementation in *HULK* differs slightly from the general approach introduced here and is further explained in Section 3.

2.2. Binary space partitioning

A key element in the efficiency, and the main novelty of our approach, is the efficient mapping of a cell to the corresponding geological formation. Unlike many other meshing algorithms, all input geometries are treated in a single step. This requires an algorithm that is able to swiftly decide whether or not a point (such as the center of a cell) is contained in a geometry. This test is often called “collision detection” and is a well-studied field in computational geometry (Lin and Gottschalk, 1998; Ericson, 2004). A simple and common technique of collision-detection is ray-tracing, where multiple rays are sent out from the query point in arbitrary directions. Each ray is then tested for its number of intersections with the geometry in order to determine its

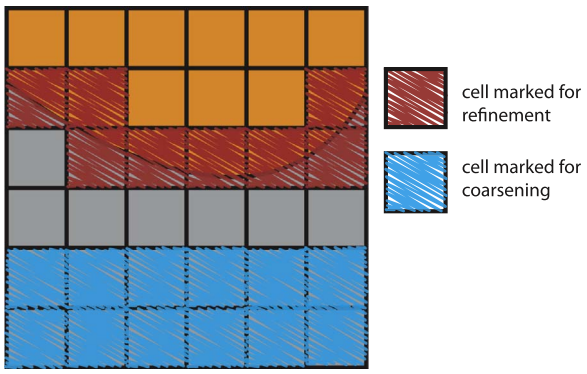

Fig. 1. A simple model with two materials. All cells which are at the interface of the two materials are flagged for refinement. In the lower part of the model, where many cells do not have neighbors close to the interface, the cells are marked for coarsening.

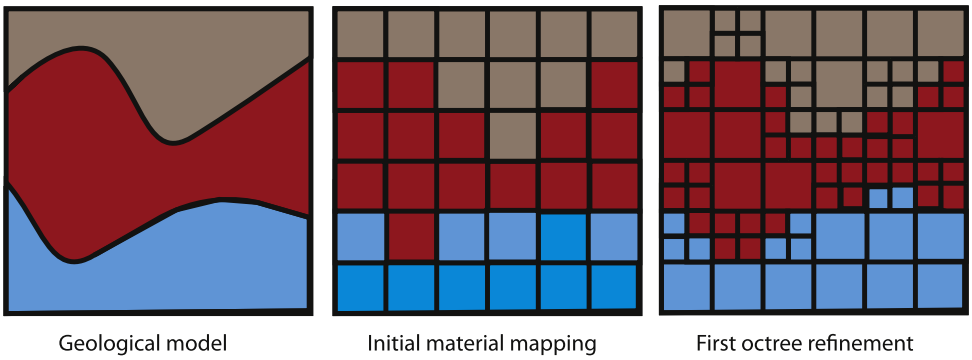

Fig. 2. Conceptual model of octree based mesh refinement in the context of geological models. Left: Geological model to be discretized. Middle: In a first step the material properties from the geological model are mapped onto the initial grid. Right: Cells are marked for refinement if any of its neighbors belong to another geological formation. This shows the result after the first cycle of octree refinement, and shows that the initial input are better resolved. With increasing number of refinement cycles, the approximation will become very close to the input geometry.

relative position as "inside" or "outside" of the geometry. Unfortunately, the ray-tracing approach has non-optimal efficiency and is prone to errors due to a number of special cases that need to be treated accordingly (Smits, 2005). Another popular approach is binary space partitioning. Binary space partitioning trees (or BSP trees for short) are structures that can be used to recursively partition n-dimensional boundary representations into subspaces with respect to dividing hyperplanes. They are one of many spatial partitioning methods, but considered to be the most versatile and powerful in collision detection algorithms (Ericson, 2004). Originally, BSP trees were developed to address the "hidden surface problem" (Fuchs et al., 1980), but also the computer game engine community takes advantage of the tree structure due to their ability to perform fast collision tests even for a very large number of polygons. If the space to be partitioned is three dimensional, the dividing hyperplanes become regular two dimensional planes. The two partitions or half-spaces generated by a dividing plane are usually called positive and negative half-spaces. We use the convention that the positive half-space lies in front of the dividing plane and the negative half-space behind the dividing plane. In two dimensions the specification of front and back-sides is ambiguous, but as our target geometries are three dimensional the definition is useful. Henceforth, the front side is the side of the geometry's discretized elements whose normal points outward from the geometry.

2.2.1. Generation of a binary space partitioning tree

A quick and easy separation of inside and outside the volume ("collision test") is accomplished with BSP trees. Unlike other types of BSP trees, in our method, the dividing planes must be identical with the input geometry's polygons. This selection of the dividing planes is also known as auto-partitioning. Furthermore, all polygons have to be selected exactly once as dividing planes in order to correctly represent the geometry. Note that no actual geometry is stored in the nodes – merely the information pointing to the next nodes in the tree. The end-nodes of a tree partitioning are often called leaves. The leaves store whether the half-space in front/behind the last dividing plane is within the geometry (solid) or outside (empty).

The generation of a BSP tree is illustrated in Fig. 3 for a simple dart shaped geometry in 2D. The initial geometry is composed of four vertices (A, B, C, and D) connected by four edges ($\overline{AB}$, $\overline{BC}$, $\overline{CD}$ and $\overline{DA}$). Now, an edge is arbitrarily chosen to serve as the first dividing plane ①, in this example $\overline{DA}$. All remaining edges are classified according to ①. On the positive side of ①, edge $\overline{AB}$ is found and which is going to be the next dividing plane ② and a part of edge $\overline{BC}$. In order to continue the decomposition, edge $\overline{BC}$ has to be split into its parts on the positive sides respectively ($\overline{BC}^+ \rightarrow$ ③ & $\overline{BC}^- \rightarrow$ ④). The generation of the tree is continued with ②, which will produce a leaf on its positive side (pointing to empty space) and the third dividing plane on the negative side. ③ points to empty space on its positive side. Since all other

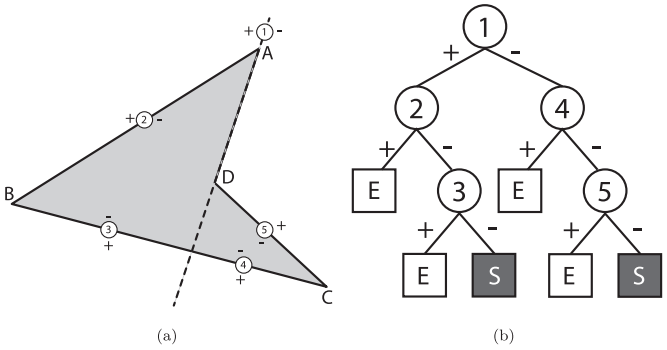

Fig. 3. A dart shaped geometry represented by a solid leaf BSP tree. (a) Geometry consisting of 4 vertices and 4 edges resulting in 5 dividing planes as the dividing plane along $\overline{DA}$ divides edge $\overline{BC}$ into two pieces. (b) The final corresponding BSP tree when all edges are chosen exactly once as dividing plane. Boxes with E signal empty (meaning outside) and dark boxes with S are solid (inside the geometry).

dividing planes on the negative side of ③have already been considered, it can concluded that it points to solid volume inside the geometry. The tree construction is continued in the same manner until all polygons have been chosen once as dividing plane. With this the BSP tree generation is complete and can be used to query for a location of an arbitrary point with regard to the geometry. Fig. 4 shows a graphical representation of such queries for the cases that the query-point is

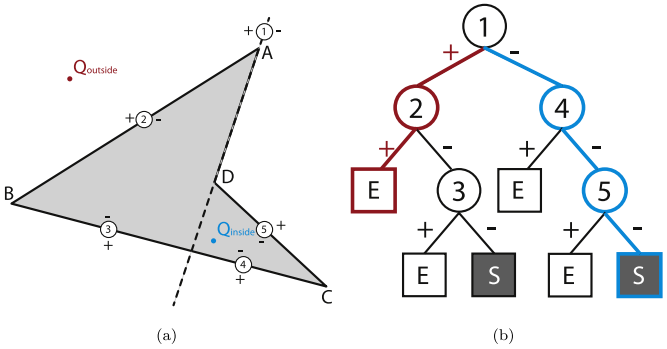

Fig. 4. Two query points Q_{inside} and $Q_{outside}$ are tested against a collision of the dart-shaped geometry to obtain a result for the relative position of the query point with respect to the geometry. (a) Geometry with the two query points – the first outside and the second inside the geometry. (b) The corresponding BSP tree query path is visualized for both query points. At each node, the query point is tested against the corresponding plane and sent down in the respective direction. The red path belongs to the outside point, whereas the blue path is assigned to the inside query point. It is evident that not all nodes of the geometry have to be tested for a collision query. This figure visualizes the optimal complexity of the BSP tree algorithm (c.f. Section 2.2.2). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

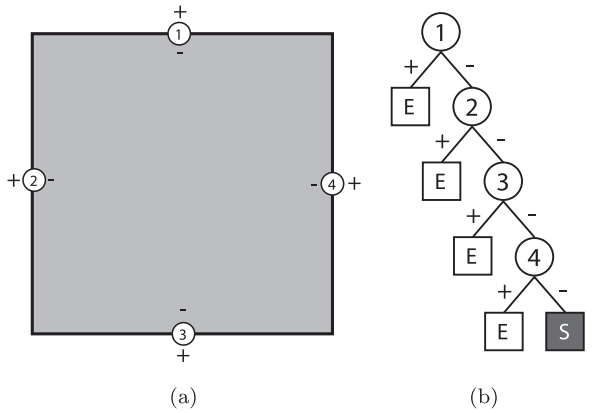

Fig. 5. The square geometry. This shows how auto-partitioning can lead to unfavorable BSP tree structures. No part of the tree can be neglected after each test. Real world geometries are not likely to suffer from this problem to the same degree, as irregularities in the domain boundaries will most likely provide suitable dividing planes.

located within and outside of the geometry. Finally note that, although the initial geometry in Fig. 3 is concave, the spatial regions corresponding to the leaves are always convex.

2.2.2. Complexity of binary space partitioning

Due to the hierarchy formed by a BSP tree with n objects in the tree, only on the order of $O(\log(n))$ objects need to be tested by a query (Ericson, 2004). This is due to the fact that typically half of the possibilities can be disregarded with each test. However, in poorly balanced trees the query can cost up to the order of $O(n)$. It is worth noting that the ray-tracing approach is also $O(n)$, yet having higher computational cost per element than the corresponding worst case BSP tree. Auto-partitioning of a square yields in fact a worst case BSP tree and is shown in Fig. 5a. Since none of the dividing planes actually divides the square, no possibilities can be disregarded during queries, leading to an $O(n)$ cost. With this in mind the BSP tree generation of our geological model benefits from complexities in the model as the number of self-intersections is greatly increased. On the other hand, it is clear that we cannot expect to generate perfectly balanced BSP trees from our input geometries with auto-partitioning. Thus a query cost in between $O(log(n))$ and $O(n)$ can be expected.

3. *HULK* – combining octree meshing and space partitioning

The theory is implemented into an efficient and rapid mesh generator. As our primary targets are finite element simulations on adaptively refined grids, we build the mesh generator based on the finite element library *deal.II* (Bangerth et al., 2015, 2007). This library has already previously been used for multiphase flow and transport simulations in the subsurface (Chueh et al., 2010, 2013). Using these subsurface simulators on adequate geological domains is a logical extension of the previous work.

In the following specific parts of the implementation are presented in depth. The pseudo-code for the currently implemented *HULK* procedure is given in Algorithm 1.

3.1. Preparation of the input data

There is currently no standard in the output format of geological modelers or CAD programs. Rather, a set of widely used formats has evolved. A well-known format is the STereoLithography (STL) format, which is available as an export format in almost any of the relevant programs from the geological modeling or CAD environments. The STL format is based on triangulated surfaces describing only the boundaries of the domain. Due to its wide availability, we use the STL output format as input to our mesh generator. STL files contain only an information header, the overall number triangles to be expected, and then a list of all individual triangles. Each triangle is defined by three vertices and a normal vector, and no information is given about the connectivity. While the data format is fairly simple, a number of issues may arise during their creation. Evidently, the input geometry must be an orientable surface for which a system for determining a normal vector can be consistently defined. In other words, it must not contain any holes, self-intersections or multiple shells. Furthermore, any triangle cannot share more than two vertices. This ensures that at least a minimal distance is maintained between all triangles. Fortunately, a number of STL repair methods exist that can detect and often fix these defects automatically (e.g. implemented in the open-source software *MeshLab* and the freeware *netfabb* and recently, Chong et al., 2007). Note that a BSP tree can be generated also if these defects are left unaccounted for, but it is clear that a valid "collision" query on the BSP tree is only possible for orientable geometries. In general, the quality of the input STL does not so much influence the final resulting mesh, but more the generation of the BSP tree structures.

3.2. Generation of the BSP trees

The BSP tree generation is performed once at the beginning of the program execution after the corresponding geometry has been read from a file. For each input geometry an individual BSP tree is generated. The pointers to the root nodes of the trees are stored in a shared vector to allow easy queries for all geometries. The current implementation does not cover the selection of a preferable splitting plane to guarantee the generation of a balanced BSP tree. Rather, the first polygon (triangle) from the STL is chosen as a root node of the tree, and generation is performed accordingly until all polygons of the input file are used once as splitting planes. Clearly, the time to generate the tree structure is directly related to the number of triangles in the geometry. But since it is performed previously to the actual mesh refinement, the overhead is kept small. During refinement the optimal complexity of the BSP tree queries minimizes the dependence on the tree size, largely contributing to the efficiency of *HULK*.

3.3. Refinement criteria

Flagging cells for coarsening or refinement is the main control on the cell size and count in the final discretization. A number of different criteria could be used. In *HULK* relatively basic criteria are implemented at the current state. As the name octree suggests, in the uniform case where all cells are refined, the total number of cells is increased by a factor of 8. Thus, choosing cells to be refined has to be performed cautiously to prevent an unpleasantly large number of cells. On the other hand, cells must be refined enough to resemble the input geometry as close as possible. To this end the following criteria based on each cell's geometric affiliation in terms of an material identifier (id) are used:

- Refinement: if $id \neq id_i$ for any i neighbors → refine
- Coarsening: if $id = id_i$ for all i neighbors → coarsen

Additionally, a suitable criterion to cease refinement has to be defined. Here, the generation stops once the distance d of the query point x is sufficiently close to the BSP tree.

- Evaluate distance: if $d(x) \leq \epsilon$ → clear refine flag

Here ϵ is the only actual tuning parameter in *HULK*. From experience, good results are achieved when $\epsilon \approx 0.005 \cdot \max(L_x, L_y, L_z)$ where L_i are the model extents in the coordinate directions.

In order to assure numerical accuracy, the maximum level difference in refinement between adjacent cells may not be greater than one.

This implies that also cells that are not directly associated to the interface might be flagged for refinement in a second internal step. Furthermore, due to the implemented criteria, the number of cells in the initial mesh has to be chosen appropriately to the smallest geometric feature that is to be resolved in the final grid.

Algorithm 1. Pseudo-code for the algorithm used in HULK using octree refinement to resolve complex geometries in hexahedral cells.

```
for all input geometries do
    read input data from file;
    generate BSP tree;
end
while cycle < maxCycles and number of cells < MaxCells do
    for all cells do
        for all BSP trees do
            if is inside current BSP tree then
                set id = corresponding material identifier;
                break loop;
            else
                continue with BSP tree for next input geometry;
            end
        end
        if (neighbor's id ≠ cell's id) for any neighbor then
            flag cell for refinement;
        end
        if (neighbor's ID = cell's ID) for all neighbors then
            flag cell for coarsening;
        end
        if cell's distance to cell's corresponding BSP tree
        ≤ϵ·max(L_x, L_y, L_z) then
            remove cell's flag for refinement;
        end
    end
    execute refinement;
end
```

4. Methods for mesh quality analysis

Measuring the quality of the generated discretized domain is an important part of a meshing algorithm. Standard procedures include the measurement of element skewness, aspect ratio and distortion. These quantities are mainly used to determine the quality based on the degree of conformity and to identify degenerate elements. Since we do not seek a conforming discretization of the input geometries and the discretization is based on regular octree meshes, some techniques of defining mesh quality no longer apply; for example element skewness and aspect ratio become constant and attack optimal value in this case. In order to give a measure of the degree of conformity nonetheless, we compute the "mean distance" between the triangulated input and our hexahedral approximation as well as the volumetric space filling.

Measuring the distance or similarity of two discretized surfaces is not straightforward. Many of the more sophisticated theories developed are used in the field of automatic face recognition (Abate et al., 2007). A simpler approach is used here which computes the minimum distance between the cell centers of cells at the interfaces and the BSP tree of the corresponding geological formation. From this, the mean distance is calculated. Ideally, the BSP tree – cell center distance distribution should have its maximum at zero. However, since we compare the cell centers we expect a shift of the mean distance towards the half the size of the largest element on the boundary. In any case, with progressing refinement of the mesh the mean distance is expected to decrease continuously.

Further, the accuracy in volume of the mesh is considered. To this end, the percentage deviation between the volume of the input V_0 and the final discretized mesh V is calculated:

$$Deviation_{vol}(\%) = \frac{V}{V_0}\cdot 100\% - 100\% \tag{1}$$

The deviation by itself is not a good indicator for the mesh quality, as no information about the shape of the geometries is incorporated and in fact could lead to very poor discretizations. However, in combination with the mean distance it becomes a viable indicator for mesh accuracy.

The quality of octree-based discretization in general is known to be good (e.g. Xing et al., 2009), but the accuracy of the numerical method depends on the overall distribution of the cell sizes. Assuming the numerical accuracy is ideal if all cells are of the same size, the utilization of octree refinement can increase computational efficiency while maintaining the same numerical accuracy only if the cell size distribution is only mildly altered. In other words, smooth cell size distributions with a single peak should always be preferred. Again, this is not a direct measurement of mesh quality, but serves as an important indicator for the expectable accuracy of the applied numerical simulators.

5. Results

The implemented algorithm is validated against three test geological scenarios. The first is a 3D model of an artificial dike intrusion into a single constituent host rock to demonstrate the general performance of the algorithm. We then present an example of a simplified three-dimensional model of the newly born hydrothermal system called "LUSI" in Indonesia (Mazzini et al., 2012; Lupi et al., 2013). This model already consists of multiple geological layers of a sedimentary basin. A third example is a full scale geological model of a part of the Swiss Jura Mountains. The models vary in size and complexity. Table 1 shows detailed information on the test geometries used.

5.1. Dike intrusion

Fig. 6 shows the initial model for the first example. As the model consists only of two geological formations, the volcanic intrusion at the bottom and a host rock on top, only the resulting hexahedral mesh for the volcanic intrusion layer is shown. Note that the input consists only of one shell which describes the geometry of the dike. We assume that the region of interest is a rectangular box around the dike, thus making a second input geometry for the host rock unnecessary.

The geometry was discretized using four refinement cycles. In Table 2 the resulting number of cells in the whole (both volcanic intrusion and host rock) are shown for all cycles. Furthermore we evaluated the performance of volumetric filling for the dike intrusion

Table 1
Three model types from different settings are used to evaluate *HULK*'s performance. They vary in size, amount of shells and their underlying complexity.

Model name	Setting	Volumes	Input size (triangles)	Complexities
Dike intrusion	Volcanic	1	$1.9\cdot10^5$	Locally steep gradients
LUSI	Sedimentary basin	5/7	$4.5–7.5\cdot10^5$	None/large scale faults zones
GeoNE	Swiss Jura	25	$1.5–11.9\cdot10^6$	Folding and faulting including offset

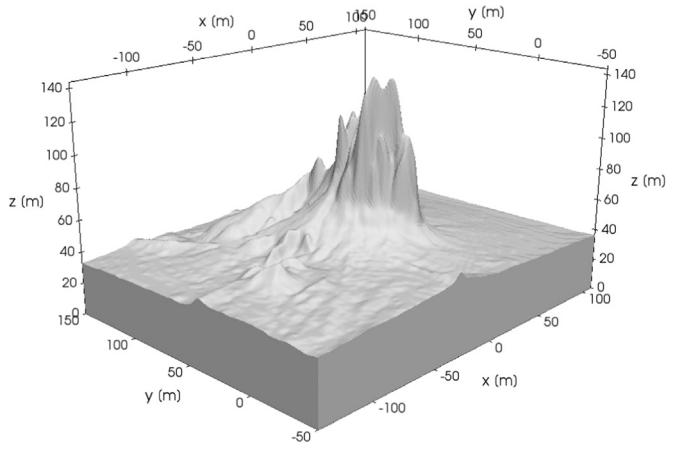

Fig. 6. The geological model and input for the dike model. The dike has a non-trivial shape with steep gradients to evaluate the implemented algorithm.

Table 2
Mesh statistics for the dike model. The number of cells, volumetric filling accuracy as well as the distance to the corresponding BSP tree were analyzed for all performed refinement cycles on the dike intrusion model. The volumetric deviation alone is not a sufficient indicator but shows in combination with the BSP distance an overall increase in mesh accuracy with refinement cycles.

Cycles	Cells	Deviation (%)	BSP distance $(-)\cdot 10^{-3}$
0	32,768	0.04	23.8
1	21,715	1.33	52.6
2	77,708	0.47	34.3
3	269,746	0.13	22.6
4	861,617	0.19	13.8

model. To this end we compared the meshed volume against the computed volume of the input STL file. The resulting deviation in volume as well as the distance (measured to the BSP tree) from the input geometry is also presented in Table 2. The accuracy of the upper layer can easily be computed by $(\delta\%)^1 = 1 - (\delta\%)^0$.

As mentioned in Section 4 the volumetric filling is not a sufficiently good indicator by itself. This is clearly visible as it is already very high even when the visual accuracy of the mesh is still unsatisfying. Regardless, a clear trend towards the correct theoretical volume is visible. In combination with the measured distance to the BSP tree we can conclude that the mesh accuracy increases with the number of refinement cycles.

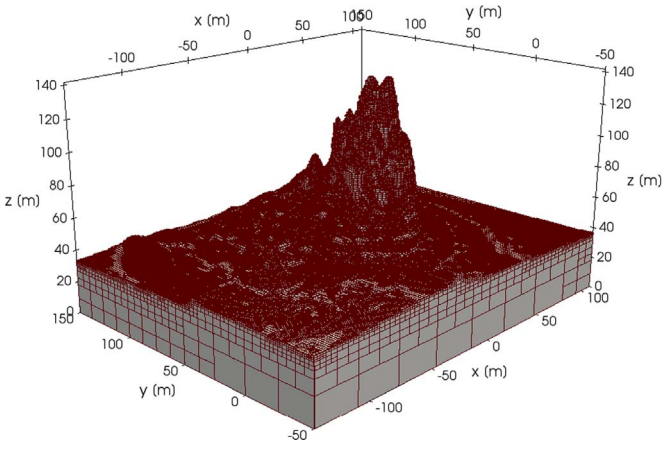

Fig. 8. The final mesh of the dike model after 4 cycles of refinement by HULK. All important geometric features of the input geometry are captured and provide a good hexahedral approximation to the initial geometry (cf. also Table 2 for detailed accuracy evaluation). Different levels of refinement are visible depending on the curvature of the dike's shape. Flat regions are slightly less refined than the peak for which cells are highly refined to capture its extends.

Figs. 7a and b show the mesh after different cycles. Clearly the refinement towards the boundaries is visible. The final model is shown in Fig. 8, which shows that all important features of the initial model were captured by the algorithm. Note that due to the selection of refinement criteria the cell size distribution is not uniform at the interface. Rather flat regions of the model are less refined, whereas areas with steeper geological features exhibit a stronger refinement to capture the curvature of the geometry correctly. After the last cycle the mesh contains roughly 1 million cells which might be too big for initial simulations. However, the accuracy of the preceding cycles is also good and yield a cell count of maximum a few hundred-thousand cells, which can be handled by modern parallel computing systems (Hammond et al., 2014).

5.2. LUSI

The second example is a simplified model of the LUSI mud eruption in Eastern Java, which appeared in 2006 and continues today as a geyser system (Mazzini et al., 2012; Lupi et al., 2013; Sawolo et al., 2009). LUSI is located in a back-arc sedimentary basin that is likely influenced by nearby active volcanism. The eruption is located at the intersection of two major regional faults which were included in the

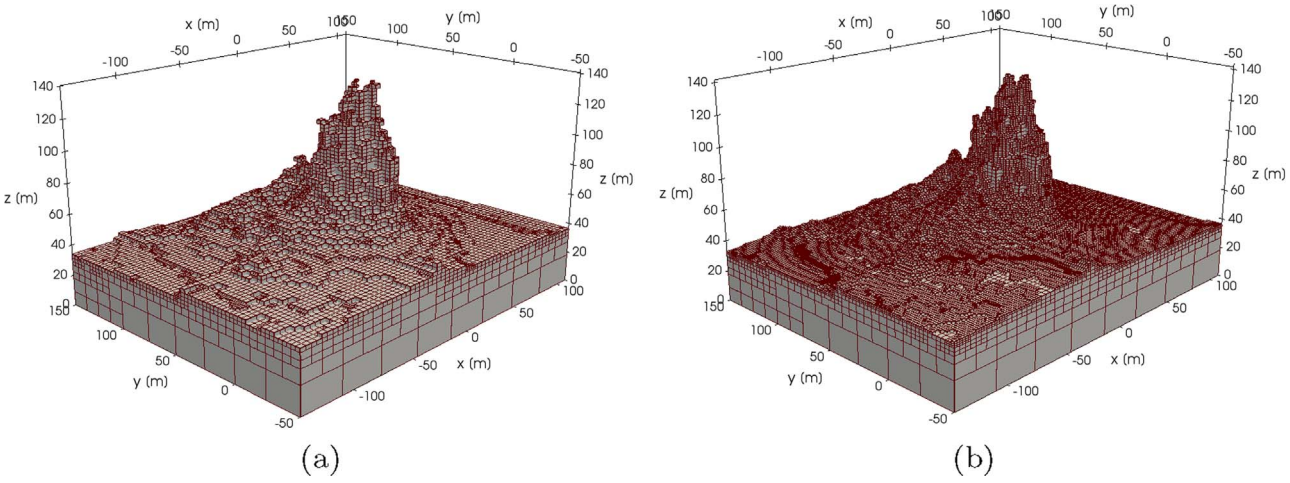

Fig. 7. The generated mesh for the dike model after different refinement cycles. (a) After the 2nd cycle. The general shape of the dike is already captured but its extends are slightly overestimated and smaller features not yet captured. (b) Mesh after the 3rd cycle of refinement. The peak shape is still not matching the input model completely. Yet on the rather flat zones of the interface refinement towards smaller features is visible.

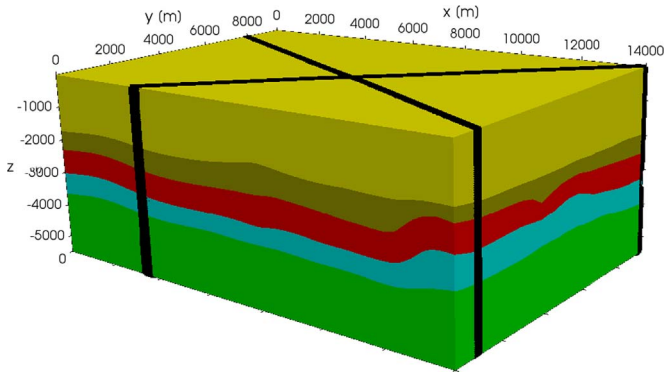

Fig. 9. The geological model for LUSI used to generate a hexahedral mesh with HULK. Note that the two faults are implemented a posteriori into the model which makes certain analysis unavailable for this model.

geological model as finite width faults. In the following we present the model with and without the faults to increase the clarity in analyses. In general, the model does not show many complexities, as the sedimentary basin is mostly undisturbed except for the intersecting strike-slip faults. We generated a model based on seismic and borehole data obtained in the framework of the LUSI Lab project and computed with *Geomodeller* (Lajaunie et al., 1997; Calcagno et al., 2008). The number of input geometries range from 5 formations up to 7 formations if the two major faults are included. The initial model is shown in Fig. 9.

The final discretizations of the LUSI model consist of 346,991 hexahedral cells without and 513,409 cells including the faults. The final meshes without and including the faults are shown in Fig. 10. A visual comparison of the input geometry and the final discretization shows that all important geological features of the sedimentary basin are preserved. The LUSI geological model was discretized using five refinement cycles in HULK. For each cycle all volumes are compared to the reference volume of the input geometry. Unlike the dike model, the accuracy of the volumetric filling is not as high during the first refinement cycles. This is mostly due to a different refinement strategy chosen in this example. For this model we decided to change the refinement strategy to a uniform refinement without coarsening or removal of refinement flags in order to show the excellent refinement towards the models boundaries. However, as shown in Fig. 11 the accuracy is greatly increased in later refinement cycles. The final accuracy can also be seen in the y-slice view of the model which is shown in Fig. 12 for the final mesh where the boundary cells for each layer and the outline of the corresponding input geometry are shown. Here it is very well visible that the boundary cells coincide very well with the input geometry after the last refinement step.

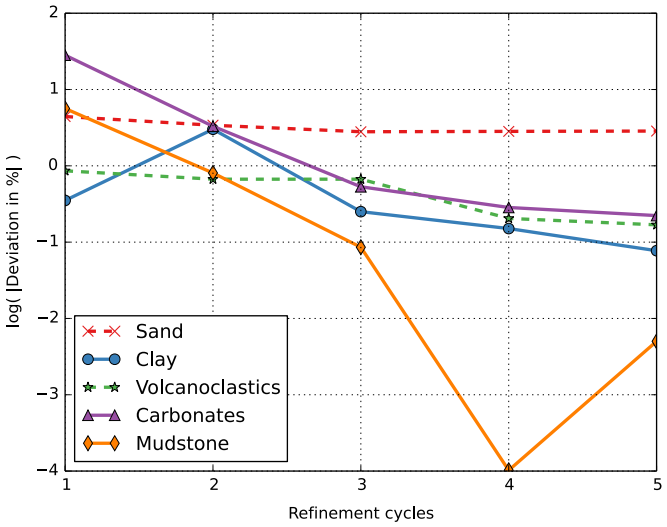

Fig. 11. The deviation of volumetric filling for the LUSI model is shown for the refinement cycles used. The accuracy is increased by one or two order of magnitude for all formations.

5.3. GeoNE

The GeoNE model is by far the most complex model that we use to evaluate the performance of *HULK*. Between May 2010 and July 2012, the Laboratory of Geothermics at the University of Neuchâtel studied the potential of deep geothermal resources in the Canton of Neuchâtel in Switzerland (cf. CREGE, 2014). As part of the project a detailed 3D geological model was built. We present a part of this geological model here to generate a hexahedral mesh. This model is going to be used also for the input mesh scaling in the next section. In this section the GeoNE50 model containing a total of 5.9 million triangles is used. The geological model as we use it here is shown in Fig. 13a. The model features geological formations of different sizes as well as folds and faults including offset. Fig. 13b shows a cross section through the model, where the intersection of multiple faults in a fold is visible. This provides an excellent opportunity to investigate the algorithm's performance in complex settings. The model consists of overall 25 parts which are all included in the meshing procedure.

Due to the high complexity of this model the final discretization includes 1.4 million hexahedral cells. The final result for the whole model and the cross section through the model are shown in Figs. 14 and 15 respectively. A visual comparison of the input geometry and the final discretization shows very good agreement. The GeoNE50 geological model was discretized using 4 refinement cycles in HULK. The high complexity of the model prohibits a meaningful volume analysis as performed in the other examples. However, the mean distance to the

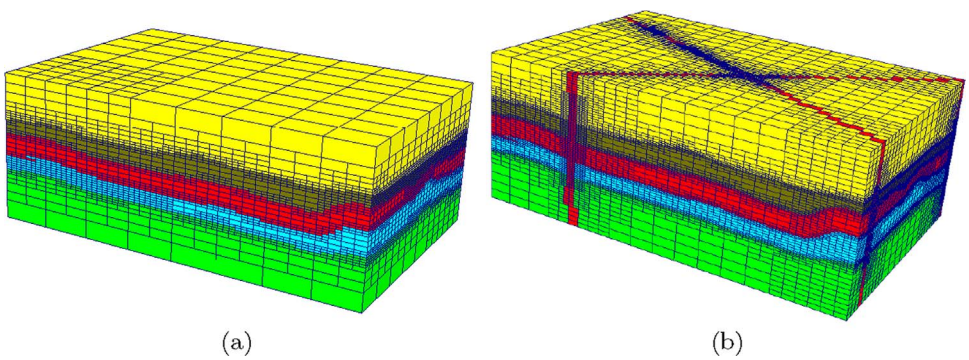

Fig. 10. The final LUSI model based only on the geological layers of the sedimentary basin with and without big scale faults. The successive refinement into the boundaries of the geological formations is visible and working well. Yellow – sandstone formation, brown – clay formation, red – volcanoclastic formation, blue – carbonate formation and green – mudstone formation. (a) Without faults and (b) with faults. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

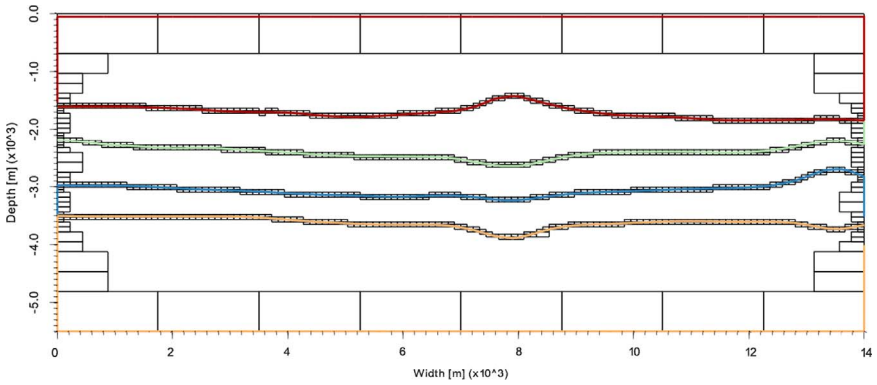

Fig. 12. The accuracy around the formation boundaries after the 5th and final cycle of refinement is greatly increased as visible in this slice through the model. The achieved accuracy is sufficient for a structured hexahedral approximation of the input geometries.

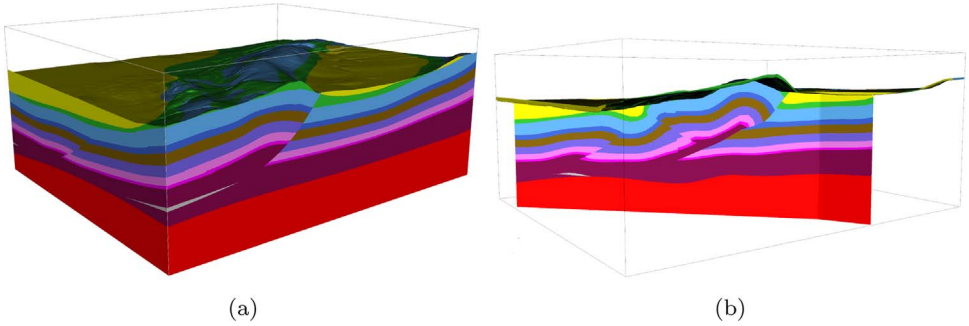

Fig. 13. The GeoNE model containing 11 formations in a total of 25 parts. (a) An outside view of the complete model. (b) Cross section through the model showing the complex interior including multiple faults including offset.

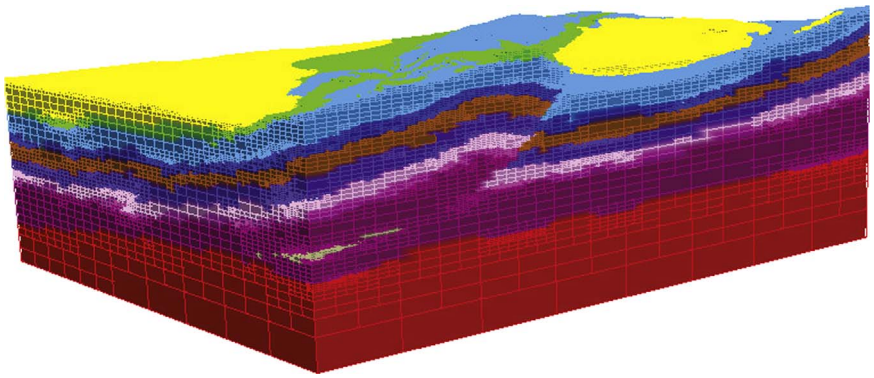

Fig. 14. The final mesh of the GeoNE50 model after 4 cycles of refinement by HULK and contains 1.4 million cells. All important geometric features of the input geometries are captured and provide a good hexahedral approximation to the initial geometry. Even smaller features such as a small intrusion (shown in gray at the front corner) as well as the very fine quaternary deposits at the top of the model are well-resolved. Dynamic cell size changes are visible especially in vicinity of the large offset fault visible on the right face of the model.

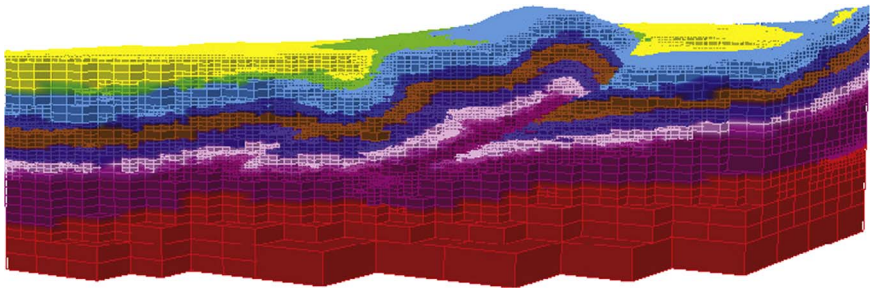

Fig. 15. Cross section through the complex interior of the final mesh of the GeoNE50 model after 4 cycles of refinement by HULK. The whole model contains roughly 1.4 million hexahedral cells. In comparison with Fig. 13b the accuracy is very good. All three main offset faults visible in the cross section are resolved by the mesh. Solely, a small intrusion (shown in gray in Fig. 13b) is not resolved by the hexahedral approximation. Note that the adaptivity of the octree mesh is not as clearly visible in this section through the mesh as many complexities have to be resolved, leaving little room for cell number optimization through mesh coarsening.

Table 3
Simple mesh statistics for the GeoNE50 model. The number of cells as well as the distance to the corresponding BSP tree was analyzed for all performed refinement cycles on the complex geological model. The BSP distance indicates an overall increase in mesh accuracy with refinement cycles.

Cycles	Cells	BSP distance (–)
0	32,768	1.91
1	40,489	2.84
2	135,290	2.74
3	459,040	1.91
4	1,408,975	1.20

BSP tree of the corresponding geological formation could be calculated and is shown in Table 3. The overall decrease in the distance indicates a distinct increase in accuracy with later refinement cycles. This further supports the conclusion of the visual analysis of the final discretization. A detailed comparison on the cross section through the geological model (cf. Fig. 13) and the discretization (cf. Fig. 15) shows how detailed the hexahedral approximation by *HULK* really is. Although the faults in the model could not be included in the meshing process as they were modeled as 2D surfaces instead of finite width faults, their resulting offset is clearly visible. Overall, all 11 formations were resolved with very good accuracy, capturing also the complex interior with intersecting fault offsets in a folding environment.

6. Discussion

In the previous section, three models with varying degree of complexity were used. However, in order to evaluate execution timings as well as scalability a more reliable basis needs to be created. To this end the GeoNE model is used. We use the quadric edge collapse decimation technique to reduce and increase the number of triangles in the input geometries while conserving all formations and faults in the model (Hoppe, 1999). The initial model is the GeoNE50 model

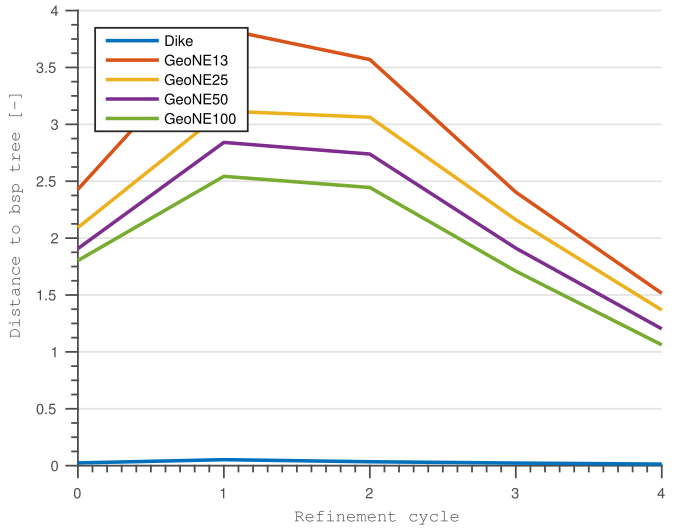

Fig. 17. The mean distance to the BSP tree is shown for the dike and GeoNE models. The initial increase in distance is caused by a dominating cell coarsening in the first refinement cycle of the GeoNE models. The final discretizations exhibit distances well below the initial value.

consisted of roughly 6 million triangles. In the following we will consider two down-sized models (GeoNE25 ~3 million and GeoNE13 ~1.5 million triangles) as well as one up-sized model GeoNE100 with a total of ~12 million triangles in the input geometries in addition to the initial GeoNE50 (~6 million), LUSI (~0.5 – 0.8 million) and Dike (~0.2 million triangle models.

6.1. Mesh quality

As discussed in Section 4 the quality of octree based meshes is in general quite good. However, for accurate simulation results the cell

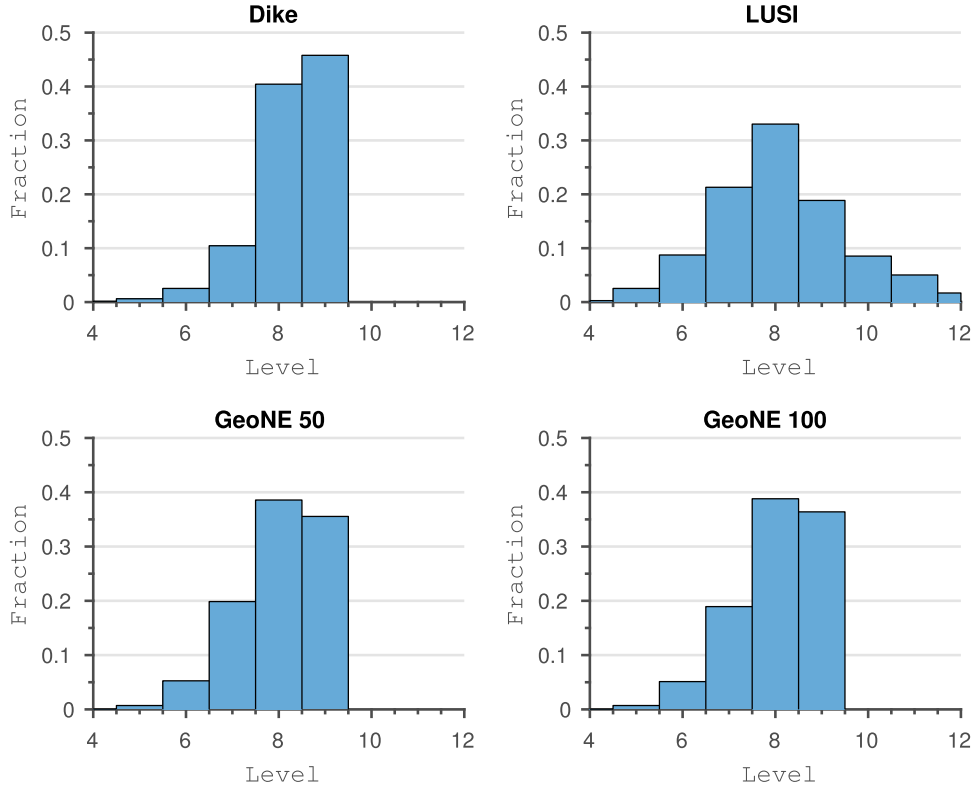

Fig. 16. Mesh quality analysis based on the refinement level distribution of the cells in four different models. Sharp distributions are likely due to sharp gradients in the input geometries, as for example the dike model. Rather smooth distributions as seen for the LUSI model also indicate rather smooth input geometries.

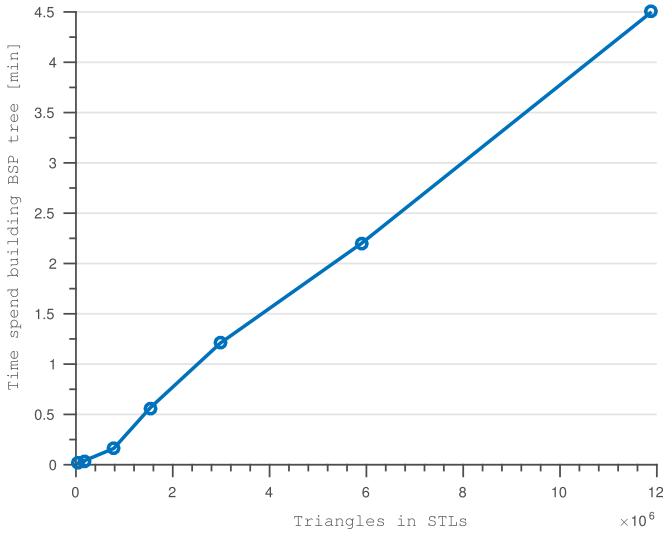

Fig. 18. The BSP generation shows excellent linear scaling for moderate to large input files. Small input files suffer from a slight computational overhead if multiple BSP trees are generated for a series of geological formations such as the smallest GeoNE model.

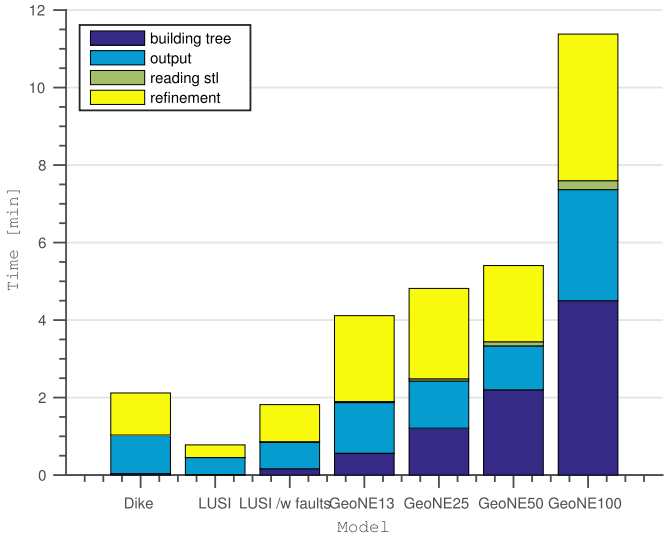

Fig. 19. Comparison of the execution times of the different models discretized by *HULK*. All models were executed in a single threaded on a Intel Xeon E5-2650 CPU with 2.3 GHz clock speed. The execution time strongly depends on the complexities in the geological model (cf. Table 1).

size distribution within the model might be important. In order to guarantee high accuracy the maximum level difference between neighboring cells is set to 1. Here we present the distribution of refinement levels for the cells in the different models. Fig. 16 shows the distributions for all three models presented in the previous section and additionally the largest version of the GeoNE model. In general, smooth distributions are visible for all models, which indicates good quality with respect to the expected numerical accuracy. The dike model's distribution has its peak at the highest level of refinement, which is due to the high curvatures in the input geometry. The effect of the curvature of the input geometries is also visible in the LUSI model that contains only moderate curvatures and which leads to a more widened distribution of the cell sizes within the model. Both GeoNE models show a similar distribution of cell sizes indicating that the resolution of the input geometries – given that it is reasonably good – does not significantly influence the distribution of cell sizes in the final discretization.

The distance to the BSP tree is also an indicator for the accuracy of the discretizations. Fig. 17 shows the calculated mean distances to the

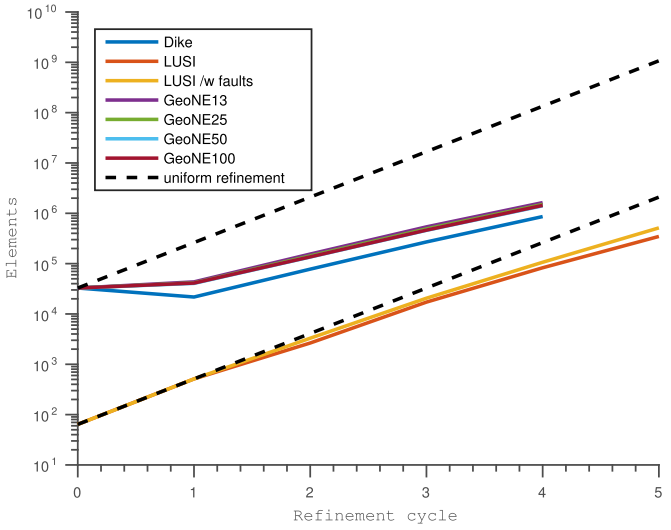

Fig. 20. Comparing the refinement factor for the models under consideration. The dashed lines represent uniform global refinement. All models show a slope much lower than the unfavorable uniform refinement curve, showing the efficiency of the chosen octree refinement in the algorithm. (Note the logarithmic scale on the *y*-axis.).

corresponding BSP tree structures for all cells in each refinement step for the dike and GeoNE models. The mean distance is decreasing continuously after the first cycle for the GeoNE models. This is due to the fact that indeed within the first refinement cycle more cells are coarsened than refined, leading to an initial shift in the medium distance towards bigger distances. For the final models the distances are well below the initial distances, indicating the overall increase in accuracy as already indicated by the volumetric deviation and the visible confirmation.

6.2. Computing timings

We show the performance and efficiency of the proposed method by a series of execution timings. All simulations were run in serial, single threaded on a Intel Xeon E5-2650 CPU with 2.3 GHz clock speed. Fig. 18 shows the time needed to generate the BSP trees for all models used in this paper. The generation of the BSP tree shows a linear scaling with size of the input geometry. Solely for small inputs small deviations from the linear behavior are visible. This in however not related to the BSP tree construction but due to a small computational overhead when generating multiple BSP trees from different geological formations. With increasing size of each formation this overhead is relatively reduced, showing better linear scaling.

The overall timings for the generation of the final discretization for all models is shown in Fig. 19. Output of the final discretization as well as the actual refinement are the main contributors to the overall runtime of the execution. Again, the near-linear scaling of the generation of the BSP trees is visible, particularly for the four GeoNE models. We observe that the time consumed by the refinement procedure does not appear to strongly depend on the size of the BSP tree, and also not strongly dependent on the number of triangles in the input. The figure further shows that the a main contributor to overall execution time is the output. This time is shared also with other mesh generators and needs to be taken into account when comparing execution timings. In general, the execution times range from less than 1 min for the simple LUSI model without faults up to about 12 min for the largest of the GeoNE models with an overall input size of roughly 12 million triangles. It becomes evident that the execution time strongly depends on the complexities in the model. Better refinement criteria might mitigate this effect in the future.

6.3. Comparison with uniform refinement

We can further evaluate the efficiency of our refinement technique by comparing the refinement behavior of all models compared to uniform global refinement. The refinement curves in Fig. 20 show the number of cells depending on the refinement cycle. Two theoretical curves for uniform global refinement with different number of initial cells are also shown. It can be seen that the proposed method is effective in terms of refinement cost as both models have a refinement factor much less than the uniform refinement. Nevertheless, the possibility to utilize the algorithm also on a structured grid with uniform refinement is very useful because many reservoir simulators still utilize this type of grid, and where our algorithm can simplify and speed up the grid generation process. In cases where the structured grid is coarse, our algorithm can be interpreted as a very simple upscaling technique mapping the geological domains and properties on the structured grid.

7. Conclusion

All three examples demonstrate that the implemented algorithm works very well. The input geometries are reasonably well represented in the hexahedral discretizations. In general the number of cells needed for a satisfying model is on the order of a few hundred thousand to roughly 1.5 million cells. This is quite a large number, but compared to the grid resolution used in classical benchmark models (e.g. Johannsen formation with $149 \times 189 \times 16 = 450,576$ cells and SPE10 with $60 \times 220 \times 85 = 1,122,000$ cells) it is competitive (Eigestad et al., 2009; Christie et al., 2001). A common solution strategy for these large full field models is the division into smaller sections and a successive solution of each section. This procedure could also be applied to the mesh constructed by our algorithm if the required number of cells is too big for a full field solution. Nonetheless, state-of-the-art parallel subsurface simulators such as *pflotran* are able to handle large number of elements (Hammond et al., 2014). With the utilization of new programming techniques and the advances in hardware accelerators will enable and promote simulations with multi-million cells. The number of hexahedral cells necessary for the models is also comparable to the number of cells in a corresponding tetrahedral triangulation. To compare against a tetrahedral discretization, we used the open source software *gmsh* and computed a conforming tetrahedral triangulation on each formation of the LUSI model (Geuzaine and Remacle, 2009). We find that in this case the number of tetrahedral cells (gmsh: 421,392) needed is even greater than the hexahedral approximation by HULK (346,991). It is important to note that the tetrahedral discretization algorithm builds a conforming mesh which coincides exactly with the input geometry. Thus the quality, namely the number of elements of the input geometries' discretization, is directly linked to the number of cells in the final mesh. In HULK however, the accuracy of the initial boundary description is not directly linked to the number of cells as we are only interested in a good quality non-conforming approximation.

The implemented algorithm could be improved by a technique called *edge-snipping*, which would result in a conforming mesh by moving the outer boundary vertices to the exact position of the input geometry surface. This procedure, however, comes at a high computational effort, and even prohibits some parallelization techniques such as scalable adaptive mesh refinement that performs better on structured octree grids (Burstedde et al., 2011). Therefore it has been omitted in the current work but yields a possible extension of the algorithm for the future.

Accounting for structures in the subsurface using a geological model efficiently helps increase the accuracy of any kind of numerical subsurface simulation. We developed and implemented a fast and efficient hexahedral mesh generator for subsurface simulations. The underlying theory and the simple structure of the algorithm makes it also possible to implement the algorithm directly in the discretization part of other simulation software. However, it can also be used as a stand-alone preprocessing unit. Simulators that use adaptive mesh refinement based on the physics can utilize our method within their simulations to dynamically resolve only those parts of the input geometry that are of interest in the current state of the simulation.

Acknowledgment

Gunnar Jansen and Reza Sohrabi have contributed equally to this paper. We appreciate the helpful discussions with B. Galvan, and comments from two anonymous reviewers significantly improved this paper. This work was supported by the Swiss National Fond (SNF) Project no. 200021-16005/1 and through the grant NFP70:Energy Turnaround under the project no. 153971. Additionally, we thank Intrepid Geophysics Ltd. for providing an academic license of the GeoModeller software.

References

Abate, A.F., Nappi, M., Riccio, D., Sabatino, G., 2007. 2d and 3d face recognition: a survey. Pattern Recognit. Lett. 28 (14), 1885–1906.
Bárdossy, G., Fodor, J., 2001. Traditional and new ways to handle uncertainty in geology. Nat. Resour. Res. 10 (3), 179–187.
Bangerth, W., Hartmann, R., Kanschat, G., 2007. deal.II – a general purpose object oriented finite element library. ACM Trans. Math. Softw., 33 (4), 24/1–24/27.
Bangerth, W., Heister, T., Heltai, L., Kanschat, G., Kronbichler, M., Maier, M., Turcksin, B., 2015. The deal.II Library, Version 8.3, Preprint.
Botella, A., Lévy, B., Caumon, G., 2014. Indirect unstructured hex-dominant mesh generation using tetrahedra recombination. In: ECMOR XIV-14th European Conference on the Mathematics of Oil Recovery.
Botella, A., Lévy, B., Caumon, G., 2016. Indirect unstructured hex-dominant mesh generation using tetrahedra recombination. Comput. Geosci. 20 (3), 437–451.
Burstedde, C., Wilcox, L.C., Ghattas, O., 2011. `p4est`: scalable algorithms for parallel adaptive mesh refinement on forests of octrees. SIAM J. Sci. Comput. 33 (3), 1103–1133.
Calcagno, P., Chilès, J.-P., Courrioux, G., Guillen, A., 2008. Geological modelling from field data and geological knowledge. Part I: modelling method coupling 3d potential-field interpolation and geological rules. Phys. Earth Planet. Inter. 171 (1), 147–157.
Casarotti, E., Stupazzini, M., Lee, S.J., Komatitsch, D., Piersanti, A., Tromp, J., 2008. Cubit and seismic wave propagation based upon the spectral-element method: an advanced unstructured mesher for complex 3d geological media. In: Proceedings of the 16th International Meshing Roundtable. Springer, New York, pp. 579–597.
Chan, C., Anastasiou, K., 1997. An automatic tetrahedral mesh generation scheme by the advancing front method. Commun. Numer. Methods Eng. 13 (1), 33–46.
Chong, C., Kumar, A.S., Lee, H., 2007. Automatic mesh-healing technique for model repair and finite element model generation. Finite Elem. Anal. Des. 43 (15), 1109–1119.
Christie, M., Blunt, M., et al., 2001. Tenth SPE comparative solution project: a comparison of upscaling techniques. In: SPE Reservoir Simulation Symposium. Society of Petroleum Engineers, Houston.
Chueh, C., Secanell, M., Bangerth, W., Djilali, N., 2010. Multi-level adaptive simulation of transient two-phase flow in heterogeneous porous media. Comput. Fluids 39 (9), 1585–1596.
Chueh, C.-C., Djilali, N., Bangerth, W., 2013. An *h*-adaptive operator splitting method for two-phase flow in 3d heterogeneous porous media. SIAM J. Sci. Comput. 35 (1), B149–B175.
CREGE, 2014. Laboratoire de Géothermie, Programme GeoNE-Développement de la géothermie profonde dans le canton de Neuchâtel. Rapport final de la Phase 1.
Du, Q., Wang, D., 2006. Recent progress in robust and quality Delaunay mesh generation. J. Comput. Appl. Math. 195 (1), 8–23.
Eigestad, G.T., Dahle, H.K., Hellevang, B., Riis, F., Johansen, W.T., Øian, E., 2009. Geological modeling and simulation of CO2 injection in the Johansen formation. Comput. Geosci. 13 (4), 435–450.
Ericson, C., 2004. Real-Time Collision Detection. CRC Press, Boca Raton.
Fuchs, H., Kedem, Z.M., Naylor, B.F., 1980. On visible surface generation by a priori tree structures. SIGGRAPH Comput. Graph. 14 (July), 124–133.
Geuzaine, C., Remacle, J.-F., 2009. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. Int. J. Numer. Methods Eng. 79 (11), 1309–1331.
Hammond, G., Lichtner, P., Lu, C., Mills, R., 2012. Pflotran: reactive flow and transport code for use on laptops to leadership-class supercomputers. In: Zhang (Ed.), Groundwater Reactive Transport Models. Bentham Science Publishers, 141–159.
Hammond, G., Lichtner, P., Mills, R., 2014. Evaluating the performance of parallel subsurface simulators: an illustrative example with pflotran. Water Resour. Res. 50 (1), 208–228.
Ho-Le, K., 1988. Finite element mesh generation methods: a review and classification. Comput.-Aided Des. 20 (1), 27–38.
Hoppe, H., 1999. New quadric metric for simplifying meshes with appearance attributes. In: Proceedings of the Conference on Visualization'99: Celebrating Ten Years. IEEE

Computer Society Press, San Francisco, pp. 59–66.
Jasak, H., Jemcov, A., Tukovic, Z., 2007. Openfoam: a C++ library for complex physics simulations. In: International Workshop on Coupled Methods in Numerical Dynamics, vol. 1000, pp. 1–20.
Lajaunie, C., Courrioux, G., Manuel, L., 1997. Foliation fields and 3d cartography in geology: principles of a method based on potential interpolation. Math. Geol. 29 (4), 571–584.
Li, Q., Ito, K., Wu, Z., Lowry, C.S., Loheide, I., Steven, P., 2009. Comsol multiphysics: a novel approach to ground water modeling. Groundwater 47 (4), 480–487.
Lin, M., Gottschalk, S., 1998. Collision detection between geometric models: a survey. In: Proceedings of IMA Conference on Mathematics of Surfaces, vol. 1, pp. 602–608.
Lindsay, M.D., Aillères, L., Jessell, M.W., de Kemp, E.A., Betts, P.G., 2012. Locating and quantifying geological uncertainty in three-dimensional models: analysis of the Gippsland Basin, Southeastern Australia. Tectonophysics 546, 10–27.
Löhner, R., 1996. Progress in grid generation via the advancing front technique. Eng. Comput. 12 (3–4), 186–210.
Lupi, M., Saenger, E., Fuchs, F., Miller, S., 2013. Lusi mud eruption triggered by geometric focusing of seismic waves. Nat. Geosci. 6 (8), 642–646.
Maréchal, L., 2009. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In: Proceedings of the 18th International Meshing Roundtable. Springer, Salt Lake City, pp. 65–84.
Maréchal, L., 2001. A new approach to octree-based hexahedral meshing. In: 10th International Meshing Roundtable, Newport Beach, pp. 209–221.
Mazzini, A., Etiope, G., Svensen, H., 2012. A new hydrothermal scenario for the 2006 Lusi eruption, Indonesia. Insights from gas geochemistry. Earth Planet. Sci. Lett. 317, 305–318.
Miller, T.A., Vessilinov, V., Stauffer, P.H., Birdsell, K., Gable, C.W., 2007. Integration of geologic frameworks in meshing and setup of computational hydrogeologic models, Pajarito Plateau, New Mexico. In: New Mexico Geological Society Guidebook, 58th Field Conference, Geology of the Jemez Mountains Region II.
Owen, S.J., Shelton, T.R., 2015. Evaluation of grid-based hex meshes for solid mechanics. Eng. Comput. 31 (3), 529–543.
Pruess, K., Oldenburg, C., Moridis, G., 1999. Tough2 User's Guide Version 2. Lawrence Berkeley National Laboratory, Berkeley.
Sawolo, N., Sutriono, E., Istadi, B.P., Darmoyo, A.B., 2009. The Lusi mud volcano triggering controversy: was it caused by drilling? Mar. Pet. Geol. 26 (9), 1766–1784.
Schneiders, R., 1996. A grid-based algorithm for the generation of hexahedral element meshes. Eng. Comput. 12 (3–4), 168–177.
Schneiders, R., 1997. An algorithm for the generation of hexahedral element meshes based on an octree technique. In: 6th International Meshing Roundtable, pp. 195–196.
Scott, M.A., Earp, M.N., Benzley, S.E., Stephenson, M.B., 2005. Adaptive sweeping techniques. In: Proceedings of the 14th International Meshing Roundtable. Springer, San Diego, pp. 417–432.
Shephard, M.S., Georges, M.K., 1991. Automatic three-dimensional mesh generation by the finite octree technique. Int. J. Numer. Methods Eng. 32 (4), 709–749.
Smits, B., 2005. Efficiency issues for ray tracing. In: ACM SIGGRAPH 2005 Courses, SIGGRAPH'05. ACM, New York, NY, USA.
Stupazzini, M., 2004. A Spectral Element Approach for 3d Dynamic Soil–Structure Interaction Problems (Ph.D. Thesis).
Thore, P., Shtuka, A., Lecour, M., Ait-Ettajer, T., Cognot, R., 2002. Structural uncertainties: determination, management, and applications. Geophysics 67 (3), 840–852.
Trefry, M.G., Muffels, C., 2007. Feflow: a finite-element ground water flow and transport modeling tool. Groundwater 45 (5), 525–528.
Tu, T., O'Hallaron, D.R., 2004. Extracting Hexahedral Mesh Structures from Balanced Linear Octrees.
Wellmann, J.F., Finsterle, S., Croucher, A., 2014. Integrating structural geological data into the inverse modelling framework of itough2. Comput. Geosci. 65, 95–109.
Xing, H., Yu, W., Zhang, J., 2009. 3d mesh generation in geocomputing. In: Advances in Geocomputing. Springer, Berlin, pp. 27–64.
Yerry, M.A., Shephard, M.S., 1984. Automatic three-dimensional mesh generation by the modified-octree technique. Int. J. Numer. Methods Eng. 20 (11), 1965–1990.
Yu, W., Zhang, K., Li, X., 2015. Recent algorithms on automatic hexahedral mesh generation. In: 2015 10th International Conference on Computer Science Education, ICCSE, pp. 697–702, July.
Zehner, B., Hellwig, O., Linke, M., Görz, I., Buske, S., 2015. A method for converting triangle-mesh-based 3d geological models into hexahedral grids for parallel finite difference simulation. In: 35th Gocad Meeting – 2015 RING Meeting, ASGA.