2.5D Hexahedral Meshing for Reservoir Simulations

David Lopez, Yoann Coudert-Osmont, David Desobry, Alexandre Benedicto, Wan-Chiu Li, Cédric Borgese, Nicolas Ray, Dmitry Sokolov, Jeanne Pellerin

2.5D Hexahedral Meshes for Reservoir Simualtions

David Lopez[1], Yoann Coudert-Osmont[1], David Desobry[1], Alexandre Benedicto[2], Wan-Chiu Li[2], Cédric Borgese[2], Nicolas Ray[1], Dmitry Sokolov[1], and Jeanne Pellerin[3]

[1]Pixel, Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
[2]Tessael, 54600 Villers-Lès-Nancy, France
[3]TotalEnergies, 91120 Palaiseau, France

Abstract

We present a new method for generating pure hexahedral meshes for reservoir simulations. The grid is obtained by extruding a quadrangular mesh, using ideas from the latest advances in computational geometry, specifically the generation of semi-structured quadrangular meshes based on global parameterization.

Hexahedral elements are automatically constructed to smoothly honor the geometry of input features (domain boundaries, faults, and horizons), thus making it possible to be used for multiple types of physical simulations on the same mesh.

The main contributions are as follows : the introduction of a new semi-structured hexahedral meshing workflow producing high-quality meshes for a wide range of fault systems, and the study and definition of weak verticality on triangulated surface meshes. This allows us to design better and more robust algorithms during the extrusion phase along non-vertical faults.

We demonstrate (i) the simplicity of using such hexahedral meshes generated using the proposed method for coupled flow-geomechanics simulations with state-of-the-art simulators for reservoir studies, and) the possibility of using such semi-structured hexahedral meshes in commercial structured flow simulators, offering an alternative gridding approach to handle a wider family of fault networks without recourse to the stair-step fault approximation.

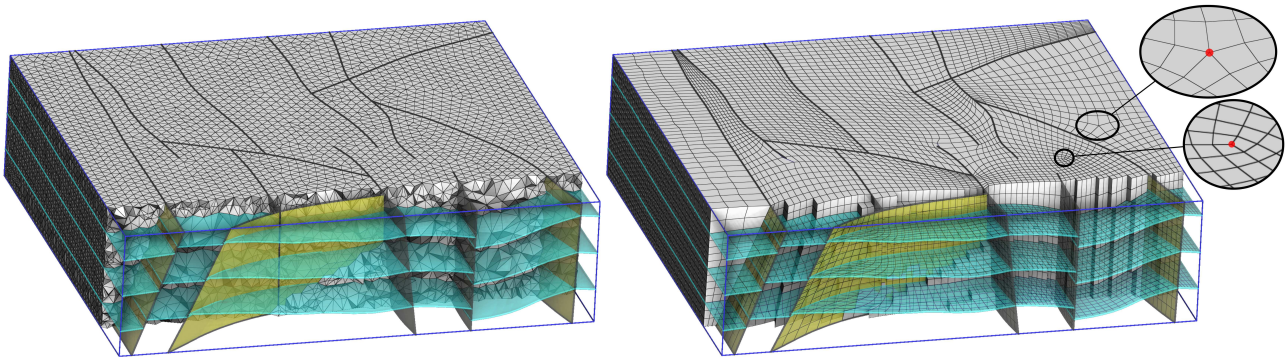

Figure 1: Given a geological model (left), our method automatically generates a full hexahedral mesh aligned with faults and horizons (right) at the cost of singular edges (close-ups).

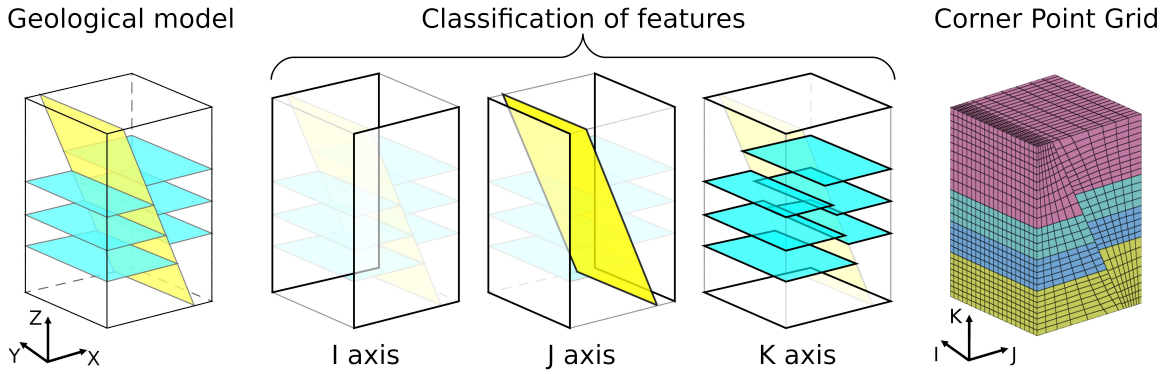

Figure 2: Classic structured hexahedral grid with corner point geometry.

1 Introduction

In order to run reservoir simulations, one needs a volume grid on which mathematical equations describing the physics can be discretized. Grids comprising hexahedral elements are often preferred for aligning cells with the stratigraphic column. In addition, their good orthogonality and low distortion provide an accurate calculation of the transmissibility between cells using the two-point flux approximation (TPFA). And finally, most of the commercial fluid simulation software only support hexahedral grids.

One of the most commonly used types of hexahedral grids for reservoir studies is the three-dimensional structured grids with corner point geometry built using pillar-based approaches by extrusion [Mal02, Fre02]. One way to produce such grids (see Fig. 2) is to first classify domain boundaries and features such as faults and horizons according to their alignment with the three axis I,J and K. Then, for each direction, we compute a smooth scalar field that is constant on each corresponding element and continuous everywhere except along faults in the K direction. Solving such optimization problems typically involves solving a system of linear equations for each direction [WC18]. The final cuts are defined as the iso-values of the scalar fields IJK for the values that constrain the features, along with some intermediate values between consecutive features if higher resolution is desired (Fig. 2-right).

In practice, it is generally easy to match horizons with K, but it is very rare to observe a fault network that can be assigned to either I or J (Fig. 3-left). Existing pillar-based technologies typically require simplification of the fault network structure or approximation of its geometry using a stair-step representation (Fig. 3–middle). This second solution excludes the use of the grid for geomechanical simulations.

We therefore propose a new method that does not require such a classification and which allows the modeling of complex fault networks without simplification by inserting singular edges (with a valence other than 4, Fig. 3-right). As a result, fault geometry is smoothly honored, enabling this grid to be used for fully coupled flow-geomechanical simulations.

Section 2 covers related work and details our contributions. In Section 3, we present our method, assuming that all the faults are vertical. Then, in Section 4, we introduce a generalization that enables the handling of non-vertical faults through a process called verticalization.

Section 5 presents our meshing results, including two extensions of the proposed method: local grid refinements around wells and along faults. Finally, in Section 5.4, we discuss the applicability of these results to numerical simulations.

2 Context

Various methods have been proposed to address the problem of constructing a hexahedral grid on nontrivial sweepable fault networks. While these methods present interesting ideas, none of them provide a fully hexahedral grid that conforms to the geometry of all the faults.

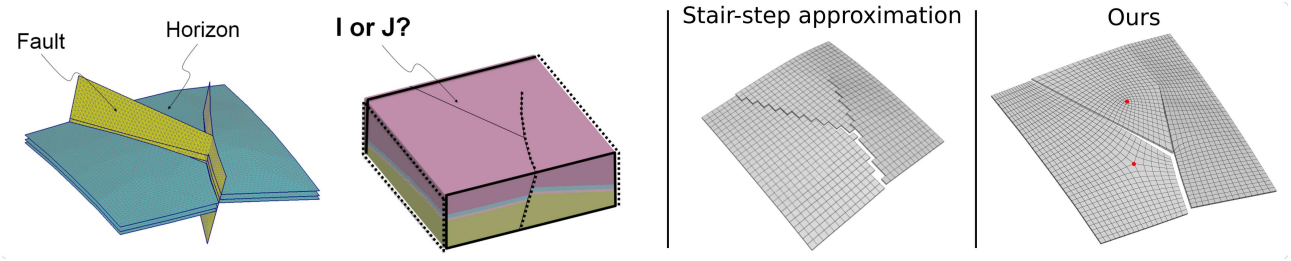

Figure 3: Left – A non-trivial 2.5D fault network that cannot be meshed using a classic corner point grid. The dotted lines represent the I scalar field, while the thick lines represent the J scalar field. However, the last fault cannot be assigned to either I or J without sharing its value with an existing feature. As a result, a portion of the final mesh would collapse.
Middle – A classic approximation. Right – Our solution introduces singular vertices (red dots).

- *Full hexahedral:*
  The approaches proposed by Mallet and Fremming and discussed in [Mal02, Fre02] involve the horizontal stair-step fault approximation for modeling nontrivial fault networks. This idea has been further extended in methods like [HNK03, GHAN09] to approximate non-vertical faults using vertical stair-steps.
  Drawbacks: When coupling flow simulations with mechanical simulations, the stair-step approximation is not suitable due to the need for a smooth representation of the geometry to satisfy mechanical boundary conditions. Consequently, it is often necessary to use two separate grids: a stair-step grid for flow simulation and a tetrahedral mesh or other unstructured grids with a smooth approximation of the faults' geometry for the mechanical simulation. This coupling process involves transferring flow and mechanical properties between the two different representations, which can introduce errors and consume significant time and memory resources.
- *Partially hexahedral:*
  Another approach is to mix a majority of structured hexahedral elements in the non-faulted zones with arbitrary polyhedral elements for the faulted zones [MSV+13, SHK+18]. In some cases, specific types of non-hexahedral elements like tetrahedra, pyramids, and prisms are also considered [GLV+17, KBL+17] to optimize the simulation using known element types from the catalog.
  Drawbacks: Meshes constructed in this manner are not compatible with standard flow simulators that only accept structured hexahedral grids, which are still commonly used. Additionally, the presence of elements such as tetrahedra or pyramids can degrade cell orthogonality, leading to larger approximations when employing TPFA schemes for flow simulation.
- *Polyhedral:*
  Another option is to utilize unstructured grids, such as tetrahedral meshes or polyhedral topologies like Voronoi or perpendicular bisector (PEBI) grids [PA94, BLC16, MLC11, KBL+17], for simulators that can handle them.
  Drawbacks: In addition to the drawbacks of partially hexahedral grids, it is worth noting that polyhedral meshes are even more memory-consuming. This is because the topology and geometry of the cells can vary significantly and are not predefined.

Contributions

In pillar-based approaches, the domain is divided into a series of layers consisting of hexahedra. Each layer represents a deformation of a thick two-dimensional regular grid (Fig. 3-middle).
In our proposed solution, we also employ a stack of layers, but with a different approach. Each layer is an expansion of a quadrangular mesh (Fig. 3-right). The key difference is the introduction of singular vertices, which are vertices with a valence not equal to 4. This allows the mesh to conform to all features present in the domain (Fig. 1).

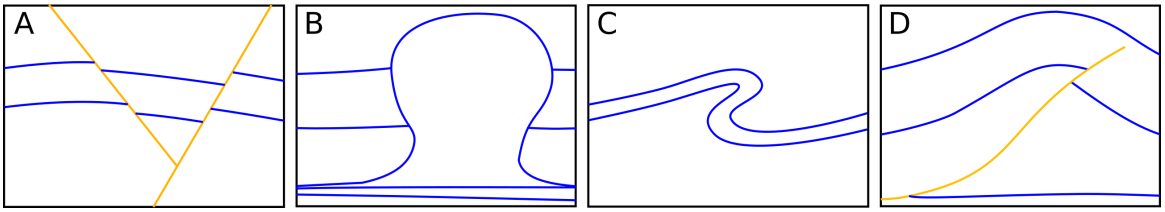

Figure 4: Cross-section views of geological structures that are out of the scope of the study: Y faults (A), salt diapir (B) overturned fold (C) and sub-horizontal faults (D).

Our contribution can be summarized in two main aspects:

- We propose a method for hexahedral grid building through extrusion (Sect. 3). This method includes an algorithm for constructing stratigraphic layers of hexahedral cells using state-of-the-art quad-meshing algorithms driven by global parameterization. It assumes that the faults present in the geological model are vertical.
- Additionally, we conducted a comprehensive study and formulated the concept of weak verticality for triangulated meshes. Building upon this understanding, we developed a new algorithm that allows for the deformation of the input geological model, resulting in the vertical alignment of faults. This advancement enables the generalization of the hexahedral grid building method to accommodate non-vertical faults (see Sect. 4).

Limitations

Our method allows us to tackle a wide variety of problems as long as they remain extrudable, that is, expressible according to the 2.5D decomposition. However, models with Y faults or salt diapirs are excluded from our study. Moreover, our solution is based on two orthogonal deformations, which make it unsuitable for models with overturned folds or sub-horizontal faults (Fig. 4). In other words, we assume that it is possible to achieve vertical faults through deformation of the model. If this is not the case, we recommend that the user remove a subset of fault alignment constraints.

Input data

A 3D tetrahedral mesh is generated from the input B-Rep (Boundary Representation) structural model using Constrained Delaunay Tessellation [Si15]. The faults, horizons, and domain boundaries of the model serve as constraints during the mesh generation process. These constraints correspond to triangle faces in the resulting mesh, which are labeled according to their surface type (refer to Fig. 5–top-left).

3 Main pipeline

Before delving into working with realistic data sets, this section focuses on cases where all faults are vertical. The objective is to generate a hexahedral mesh with a combinatorial structure that is more complex than a simple two-dimensional regular grid, while still conforming to the horizons.

The process begins by computing a 2D mesh in the xy plane (Sect. 3.1) and subsequently slicing the model into layers (Sect. 3.2). Each layer is then filled with hexahedra (Sect. 3.3) through the propagation of the 2D mesh.

3.1 Compute a quad mesh in xy plane

Our objective is to generate a 2D quad mesh that aligns with the projection of the faults onto the xy plane. To achieve this, we utilize existing 2D quad meshing tools that rely on global parameterization techniques.

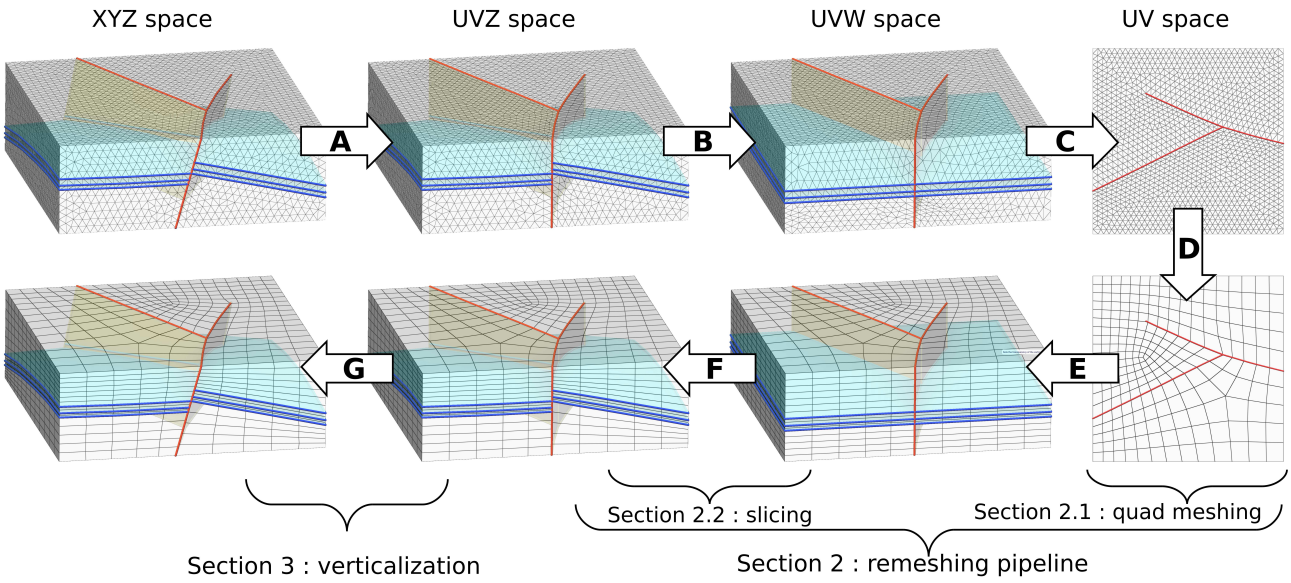

Figure 5: Method overview: (A) deform the model to have vertical faults, (B) deform the model to have horizontal horizons, (C) formulate a 2D quad remeshing problem from top-view, (D) generate a quad mesh, (E) extrude the quad mesh in the w direction to generate a hexahedral mesh, (F) inverse-map $uvw \rightarrow uvz$, and (G) inverse-map $uvz \rightarrow xyz$.

Firstly, we convert and simplify our problem to provide a suitable input for the quad meshing algorithm (see Fig. 5–top-right, Section 3.1.1). Then, we generate a quad mesh (see Fig. 5–bottom-right, Section 3.1.2) that satisfies the specified input constraints.

3.1.1 Generate the 2D problem

Faults are represented by a collection of triangles, which correspond to the facets of tetrahedra. In this section, we assume that the faults are vertical, and their projections onto the xy plane are line segments. Therefore, the projected faults in the xy plane can be represented as a set of connected line segments.

Based on this observation, we compute an appropriate input for the quad remeshing algorithm using the following steps:

- Project all the vertices of the tetrahedral mesh onto the xy plane.
- Connect pairs of vertices that correspond to an edge of a fault boundary located beneath the fault (refer to Fig. 6–left).
- Clean up the resulting graph by collapsing edges that are shorter than the expected edge length of the final hexahedral mesh. This step removes insignificant polylines and eliminates noisy details in long polylines. To ensure the preservation of long polyline extremities, collapsed edges are replaced with a vertex positioned at the extremity with the lowest valence. Additionally, intersecting edges are removed as they are invalid inputs for the quad meshing algorithm.
- Generate a Delaunay triangular mesh that is constrained by the edges and refined to achieve a consistent mesh density (refer to Fig. 6–middle).

This process produces the standard input required for quad remeshing algorithms based on global parameterization, which is a triangular mesh with certain edges flagged as features.

3.1.2 Quad mesh generation

In our project, we adopt the global parameterization-based approach for quad mesh generation. The idea is to combine two numerical optimization methods: the first one provides a heuristic giving

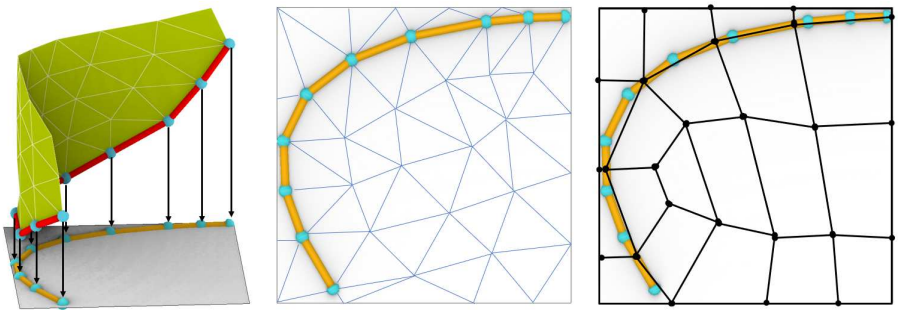

Figure 6: Left: Bottom edges (red) of the fault (yellow) are projected on xy plane (grey) to produce the 2D constraints (orange). Middle: a triangulation of xy domain with the constraints (orange) located on edges. Right: a quad mesh that conforms with the constraints.

an orientation to the quads, and the second one adjusts the number, size, shape, and position of the quads to produce a mesh.

- The heuristic part produces a frame field [RVLL08] that must be aligned with the feature edges. Many algorithms [VCD+17] address this problem, but our feature edges may touch each other with thin angles, which leads to choosing an algorithm that handles these complex cases well [DPR+22].
- For the second part, classic algorithms [BZK09, KNP07] could be used, but we have opted for [CBK15] due to its robustness: it never collapses blocks, even when very thin quads are required. In this step, the desired size of the quads must be prescribed, and we set it to the average edge length of the input tetrahedral mesh to preserve the mesh density.

The output of this step is a map of the triangulated mesh with discontinuities along some edges. However, the map is designed in such a way that a regular grid within the map corresponds to a quad mesh of the original geometry. The quad mesh can be extracted from the map using the algorithm proposed by [EBCK13] (Fig. 6–right).

3.2 Slice the model in the z direction

The hexahedral mesh we aim to generate consists of stacked layers of hexahedra. Each layer follows the structure of the quad mesh obtained in the previous section, expanded by a thickness of one hexahedron.

Now, we need to determine how to slice the model into these layers of hexahedra. The challenge lies in ensuring that the horizons are positioned precisely at the interface between two adjacent layers.

To represent the layers, we utilize a piecewise linear scalar function $\mathcal{W}$ defined over the model. This function is characterized by its values w_i assigned to each vertex i. Each layer is then defined as the portion of the model where the values of $\mathcal{W}$ fall within a specific range.

To compute $\mathcal{W}$, the horizons are arranged in a stratigraphic order and assigned real numbers based on their stratigraphic depth. Each vertex belonging to the same horizon h is constrained to have the same value W_h. The function $\mathcal{W}$ is then defined as a harmonic field that satisfies these constraints by interpolating between the assigned values W_h. The model is subsequently decomposed by cutting it along the level sets of the iso-surfaces defined by W_h.

In practice, $\mathcal{W}$ is obtained as the solution to the following optimization problem:

$$\begin{cases} \min \sum_{ij} c_{ij} \|w_i - w_j\|^2 & \text{if } ij \text{ is an edge} \\ w_i = W_h & \text{if } i \text{ is on horizon } h \end{cases}$$

In our case of isotropic meshes, we simplify the problem by using $c_{ij} = 1$. However, for anisotropic meshes, it is recommended to use cotangent weights [Cra19] (while ensuring that $c_{ij} > 0$ to avoid local extrema in $\mathcal{W}$).

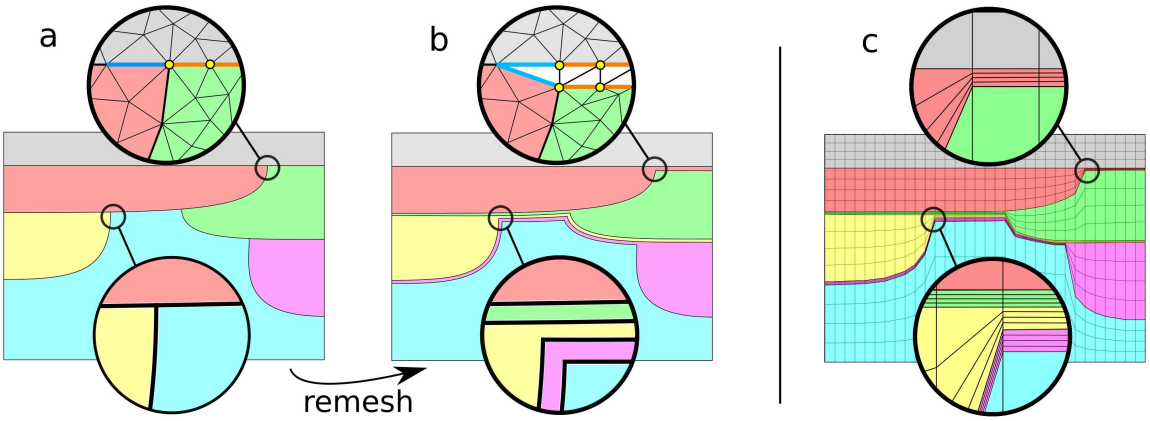

Figure 7: Erosion and sedimentation phenomena lead to horizons intersections (a). Additional tetrahedra of zero volume are inserted to make the discontinuity disappear (b). The resulting hexahedral mesh is conformal, therefore, it also contains zero volume cells (c).

Pre-treatment *Discontinuities in $\mathcal{W}_i$:* To account for horizon discontinuities across faults, a pre-processing step is performed to disconnect the tetrahedral mesh along fault boundaries. This step involves splitting each fault vertex into two new vertices, one on each side of the fault. As a result, the function $\mathcal{W}$ is represented by its corresponding value w_i on each vertex i of the modified tetrahedral mesh, allowing for the representation of horizon discontinuities across fault boundaries. *Stratigraphic unconformities:* Pre-meshing is also necessary when dealing with stratigraphic unconformities, such as erosions or baselaps, as it involves modeling the missing parts of the sedimentary layers. To achieve this, the input mesh is augmented with zero-volume tetrahedra until each layer effectively separates the volume of interest into distinct compartments (Fig. 7).

Post-treatment If more than one layer is required between two successive horizons, it is subdivided by splitting its interval of values $\mathcal{W}$.

3.3 Hexahedral mesh generation

The final mesh is obtained by lifting the quad mesh computed in the xy plane to each iso-surface W_h and generating hexahedra between each pair of successive iso-surfaces.

However, a challenge arises when dealing with points located on faults, as they may have multiple Z values. This is due to the fact that the tetrahedral mesh and the hexahedral mesh represent the UVW space discretizations and are not able to precisely capture faults with the same geometry. As a result, points near faults are not guaranteed to be on the same side of the fault in both representations.

To address this issue, when lifting the quad mesh, it is crucial to ensure that the Z value is read from the same side of the fault in the tetrahedral mesh. Simply reading the Z value from the tetrahedron containing the point is not sufficient. Instead, we need to find a tetrahedron that is on the correct side of the fault and close enough to the point.

To achieve this, we employ the following algorithm, depicted in 2D in Figure 8:

For each hexahedron H and each vertex i of H:

1. Compute all tetrahedra $\mathcal{T}_i$ that intersect a small sphere centered on i in the xyw space.
2. Compute the tetrahedron t_g that contains the barycenter of H.
3. Select the tetrahedron $t \in \mathcal{T}_i$ that is closest to t_g and use its linear map to evaluate the z coordinate. In this context, the distance between tetrahedra t and t_g is defined as the minimal number of tetrahedra in a path between t and t_g that does not cross any faults.

The key concept behind this algorithm is to assume that the center of the hexahedra is located on the correct side of the fault in the tetrahedral mesh. Consequently, the Z value should be evaluated

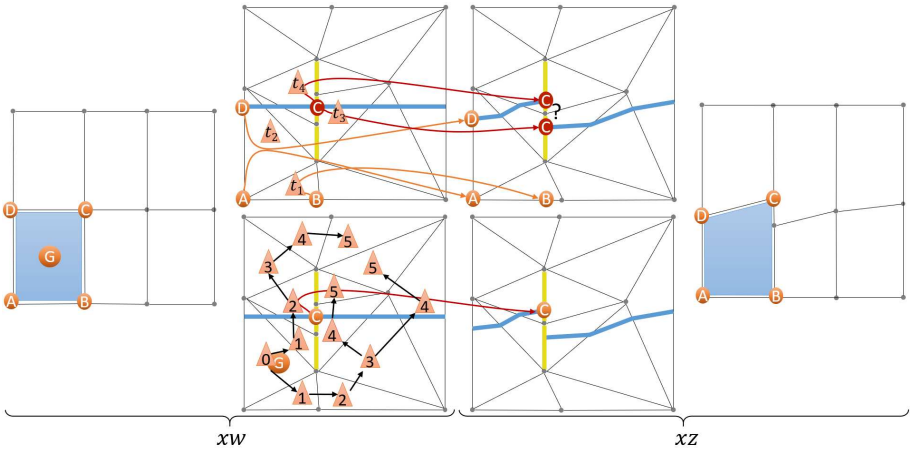

Figure 8: Mesh generation illustrated in a 2D space xz. The input model (third column) consists of a mesh with a vertical fault (yellow) and a horizon (blue). By replacing the z coordinate with the field $\mathcal{W}$ (second column), the horizon becomes horizontal, allowing for an axis-aligned grid in the xw space (left) to conform with faults and horizons. Projecting each corner of the grid cells in the xz space results in the quad remeshing of the model (right). Corners A, B, and D are located on triangles t_2, t_1, and t_2 respectively (middle left), and their projection is determined by the linear maps of their respective triangles (top middle columns). On the other hand, corner C is located on both triangles t_3 and t_4, leading to two possible projections. To determine which projection should be used, we find the triangle (t_2) that contains the center of the cell (G) and compute the number of triangles that need to be crossed to reach each of the other triangles (bottom middle columns). The closest triangle (in this case, t_4 with a distance of 2) is chosen to define the projection of corner C.

from a tetrahedron on this side of the fault. We select this tetrahedron by choosing the one that can reach the barycenter of H without crossing the fault and with the shortest path. As a default value, we set the radius of the small sphere to 10% of the expected edge size of the hexahedra, which is significantly larger than the maximum geometric error caused by the discretization.

4 Generalization to non-vertical faults

The pipeline described above assumes that faults are vertical, which is not typically the case in practical scenarios. To address faults with more general geometries, we propose a mapping of the tetrahedral mesh of the geological model to a UVZ space, where faults are vertical. We then apply our pipeline in the UVZ space and subsequently apply the inverse transformation to bring the hexahedral mesh back to the real-world XYZ space. The challenge lies in defining the mapping function $\mathcal{U} : xyz \to uvz$.

Since the geological input model is a tetrahedral mesh and our pipeline operates on tetrahedral meshes, the function $\mathcal{U}$ is naturally linear within each tetrahedron. It can be represented by the image U_i of each vertex i.

Note: the deformation does not involve displacements in the z direction; rather, moving points in the x and y dimensions is sufficient to achieve the verticalization of faults.

4.1 Formulation with weakly vertical fault

In our specific settings, a fault is defined as a subset of triangles within a tetrahedral mesh. It would be natural to consider a fault as vertical if the vector $(0, 0, 1)$ belongs to its tangent plane at all points. However, using this definition, the projection of the fault onto the UV plane becomes a polyline, with each triangle being projected onto a single segment of the polyline. As depicted in Figure 9, this constraint results in the majority of vertices being vertically aligned.

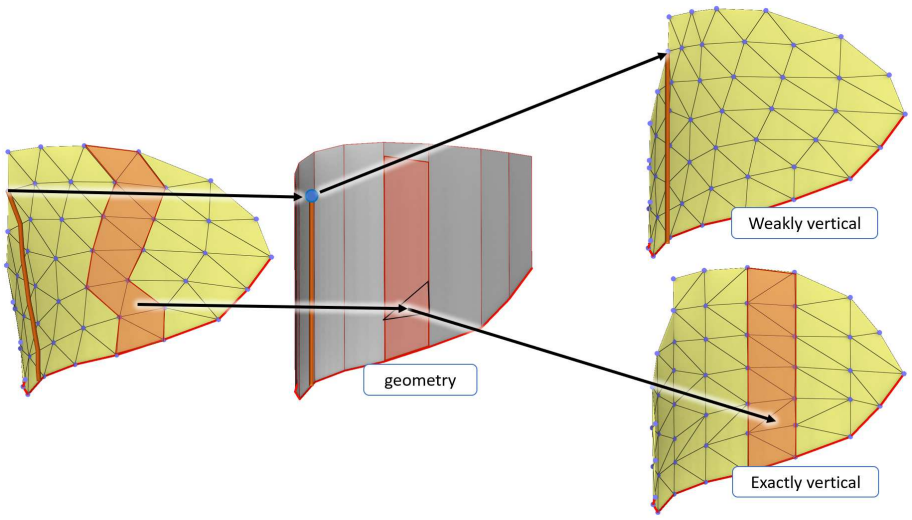

Figure 9: Weak versus exact verticality. To make a fault (left) vertical without moving its bottom edges (red), the final surface must be located inside the grey surface (middle). Weak verticality (top-right) only forces vertices to be located on the vertical geometry. Exact verticality (bottom-right) forces triangles to be located on the vertical geometry. Even in this toy example, exact verticality generates a lot of distortion in the mesh.

It's important to note that we aim to compute a deformation $\mathcal{U}$ of the input mesh without altering its combinatorial structure. Consequently, when making faults vertical, we need to manipulate only the positions of the vertices. As a consequence, enforcing strict verticality of faults would likely lead to significant distortion in the mesh.

To address this concern, we propose a **weaker definition of verticality**. Under this definition, a fault is represented by a polyline in the UV plane that satisfies the following conditions: each vertex of the fault is projected onto the polyline, and the polyline itself corresponds to the projection of the bottom edges in UVZ space. Here, the term **bottom edges** refers to the edges of the fault's boundary that are situated beneath the fault.

By employing this weaker definition of fault verticality, we can formulate our problem as finding $\mathcal{U}$ with minimal distortion, subject to the following condition for each vertex i belonging to a fault:

$$U_i = \lambda_i U_{org(i)} + (1 - \lambda_i) U_{dest(i)}$$

Here, λ_i takes values in the range $[0, 1]$, and vertex i is projected onto the line segment defined by $org(i)$ and $dest(i)$ at barycentric coordinates $\lambda_i, (1 - \lambda_i)$ (Fig. 10–right).

To ensure the tractability of this problem, we first determine $org(i)$, $dest(i)$, and λ_i (Sect. 4.2) using a heuristic approach that approximates "the curves that will be transformed into vertical lines (in UVZ)" by employing streamlines of a smooth vector field. Subsequently (in Section 4.3), we compute the map U with minimal distortion, while considering the constraints imposed by weakly vertical faults defined by the values of $org(i)$, $dest(i)$, and λ_i. The resulting map U is continuous, linear within each tetrahedron, and ideally one-to-one. Hence, it can be used to deform the input tetrahedral mesh (via U) and the resulting hexahedral mesh (via U^{-1}). It is essential to ensure that the sphere used in Section 3.3 is sufficiently large to accommodate the differences in sampling arising from these transformations.

4.2 Determine $org(i), dest(i), \lambda_i$

Ideally, $org(i)$, $dest(i)$, and λ_i should be chosen to minimize the distortion of U. Unfortunately, $org(i)$ and $dest(i)$ are qualitative variables that are very hard to optimize using numerical methods. For this reason, we prefer to determine them using a heuristic approach.

To this end, we define a "gravity" vector field G in the XYZ space (more on that below). G is tangent to the faults, so for every fault vertex i, we trace a streamline starting from that vertex.

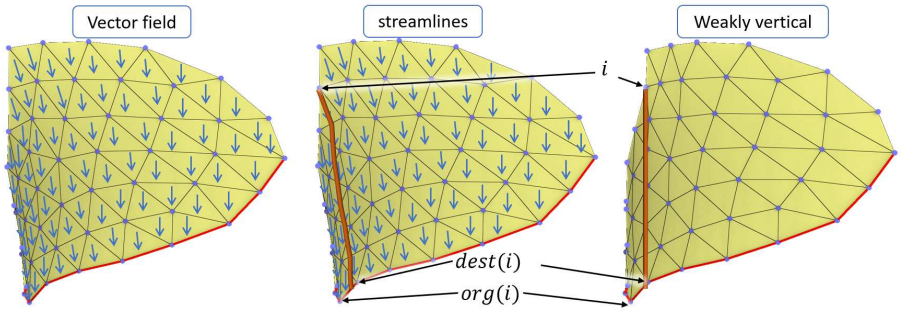

Figure 10: Verticalization pipeline: we compute a 3D vector field that is tangent to the faults (a restriction to the fault is given on the left), we trace streamlines from each fault vertex i (middle) to determine where it is leaving the fault $(org(i), dest(i), \lambda_i)$, and we find a minimal deformation of the model such that the extremities of the streamline becomes aligned in z coordinate.

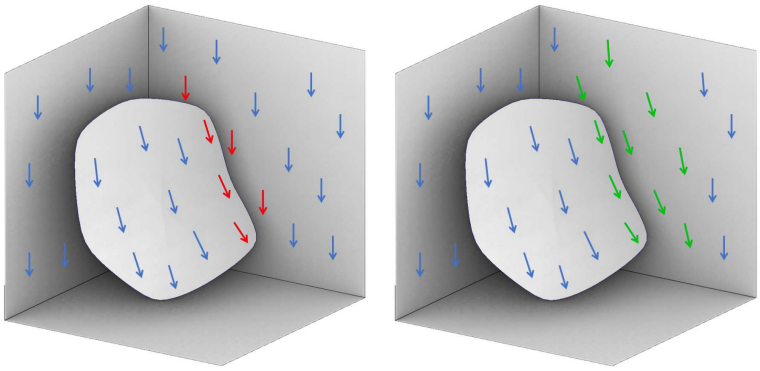

Figure 11: If the vector field is optimized only on faults/boundaries (left), the mesh will be strongly distorted between them. That's why we prefer optimizing it everywhere (right).

This streamline escapes the fault at some point $H_i = \lambda_i X_{org(i)} + (1-\lambda_i) X_{dest(i)}$, thus defining the bottom edge $org(i), dest(i)$ as well as its barycentric coordinate λ_i.

The vector field G is constant on each tetrahedron and is obtained as the solution of the following optimization problem:

$$\begin{cases} \sum_{\text{adjacent tets } t,t'} \|G_t - G_{t'}\|^2 & \text{is minimal} \\ G_t \cdot (0,0,1) = -1, & \text{for all tet } t \\ G_t \cdot n(f) = 0, & \text{if tet } t \text{ has a fault facet } f \end{cases} \tag{1}$$

where G_t and $G_{t'}$ are the field values on neighboring tetrahedra t and t', and $n(f)$ is the normal of the fault triangle f.

This formalization is based on the following observations:

- We use the streamlines of G to approximate the pre-image of vertical lines (in UVZ). Therefore, we expect that minimizing the variation of G at this step will help minimize the distortion of $\mathcal{U}$ at the end. As illustrated in Figure 11, it is important to define G everywhere in the volume, despite using it only to trace streamlines located on faults.
- Setting the z coordinate of G to -1 is a simple way to force streamlines to cross the model from top to bottom. As the norm of G does not affect streamlines, this constraint only forces the field to point downward.
- The field being orthogonal to the fault normal vector simply ensures that the streamlines will follow the fault and escape it through a fault boundary (i.e., a bottom edge).

This optimization problem is easy to solve because it is a quadratic energy function subject to linear constraints. As proposed in [BZK12], we can transform the problem into an unconstrained

one by eliminating certain variables. This transformation leads to a standard least squares problem, which is equivalent to solving a linear system of equations.

Once the vector field G is computed, we can proceed to move each fault vertex i along the fault by following G until it reaches the bottom edge $[org(i), dest(i)]$ with barycentric coordinates $\lambda_i, (1-\lambda_i)$.

Implementation details

When the edges of the fault boundary are nearly aligned with the vector field, classifying them as bottom edges may not provide meaningful results. However, our process can still handle such situations by employing a clean-up step, as described in Section 3.1.1.

Nevertheless, it is preferable to avoid these situations altogether as it allows for better preservation of topological information. For instance, when two faults intersect, we can directly enforce the intersecting edges to be vertical. This constraint ensures that the quad meshing algorithm will place a vertex precisely at the intersection, thereby preserving crucial features such as fault intersections.

Similarly, we can apply the same type of constraint to force wells to be vertical. This constraint will adjust the resulting mesh to accurately match the position and trajectory of the wells.

4.3 Optimize deformation $\mathcal{U}$

The deformation $\mathcal{U}$ is represented by vertex coordinates U_i and is linear on each tetrahedron. Similar to the vector field G, we obtain $\mathcal{U}$ by solving an optimization problem:

$$\begin{cases} \texttt{minimize } \sum_t \|\nabla U_t - Id_3\|^2 \\ (\lambda_i U_{org(i)} + (1-\lambda_i) U_{dest(i)} - U_i) \cdot (1,0,0) = 0 \\ (\lambda_i U_{org(i)} + (1-\lambda_i) U_{dest(i)} - U_i) \cdot (0,1,0) = 0 \end{cases} \tag{2}$$

In this formulation, ∇U_t represents the Jacobian matrix of $\mathcal{U}$ on tetrahedron t, Id_3 is the 3×3 identity matrix, and $i, org(i)$, and $dest(i)$ are vertices.

The objective of the optimization problem is to minimize map distortion while allowing for translation. The verticality constraint ensures that fault vertices are projected onto their bottom edge, but the geometry of the bottom edges is not predetermined. Instead, it is determined during the optimization process.

Similar to the previous section, this optimization has a quadratic objective function and linear constraints, which can be solved by reducing it to a linear system of equations.

5 Results

In this section, we evaluate the performance of our algorithm using a set of synthetic models that capture the challenges encountered in real field data.

Figure 12 presents the results of our method on benchmark scenarios from [PCJ+15]. Following the remeshing process, faults and horizons are represented as subsets of the faces of the hexahedra, forming continuous quadrangular meshes. Consequently, fault boundaries are now approximated by stair steps (Fig. 12–bottom). As anticipated, the quad layers accurately conform to the triangulated horizons on both sides of vertical discontinuities (Fig. 12–top-right).
For stratigraphic unconformities, our conformal approach creates zero-volume hexahedra that represent the "missing" portion of each sedimentary layer (Fig. 13). This representation of "dead-cell" satisfies the input constraints of standard flow simulators and can be easily removed if necessary.

To demonstrate the effectiveness of our method, we conducted further tests on fault networks with complex intersections between faults and domain boundaries (Fig. 14). Thanks to the robustness of our quad meshing algorithm, we can handle intricate fault configurations at the expense of a small number of singular vertices.

We provide examples that explore the capabilities of our quad meshing method with additional constraints to locally improve the mesh (Fig. 15) and refine it further (Sect. 5.2).

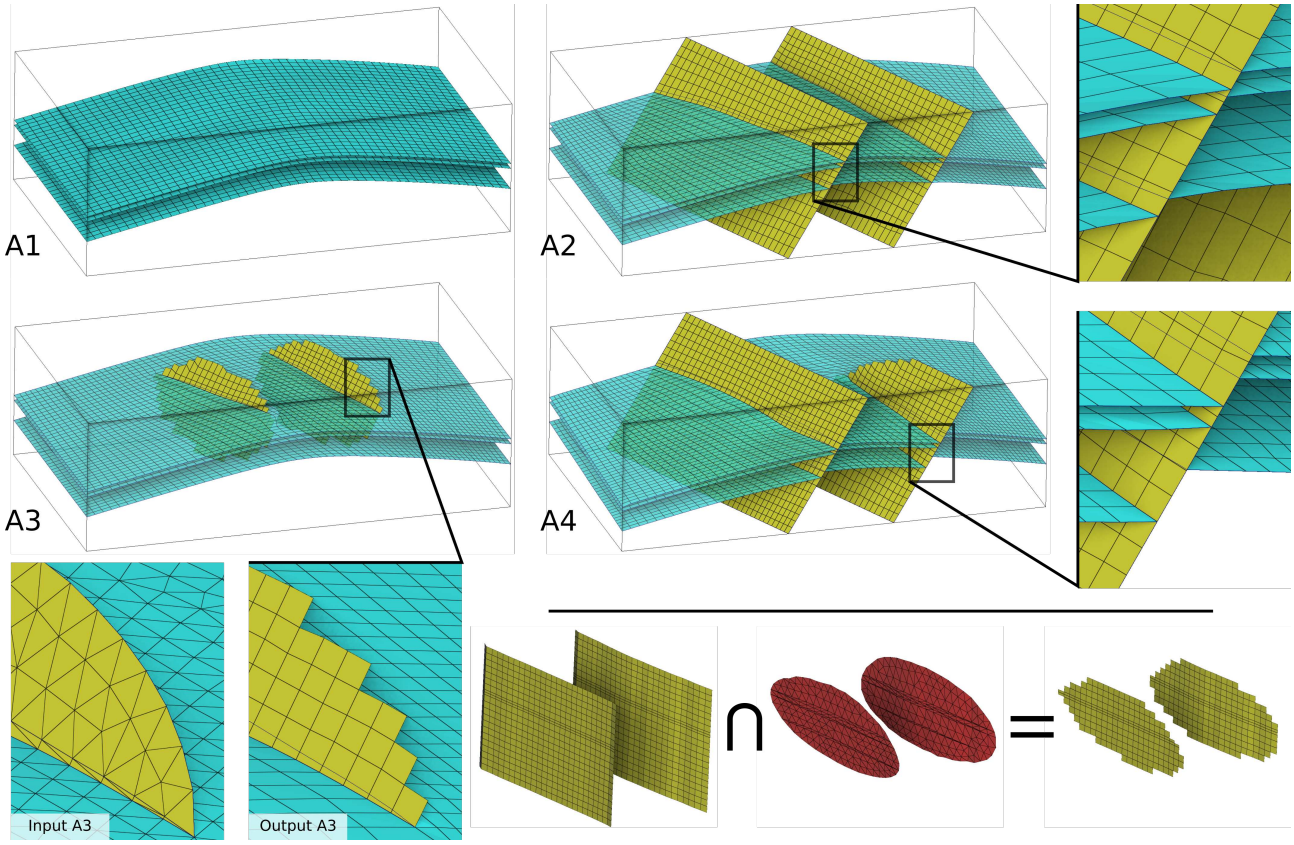

Figure 12: Benchmark models used in [PCJ+15]. Features are well preserved whether with small or large fault throws in $\mathcal{W}_i$ (close-up on the right).
Dying faults are solved by tracing a ray passing through the centroid of each candidate quad and having its normal as direction (bottom).

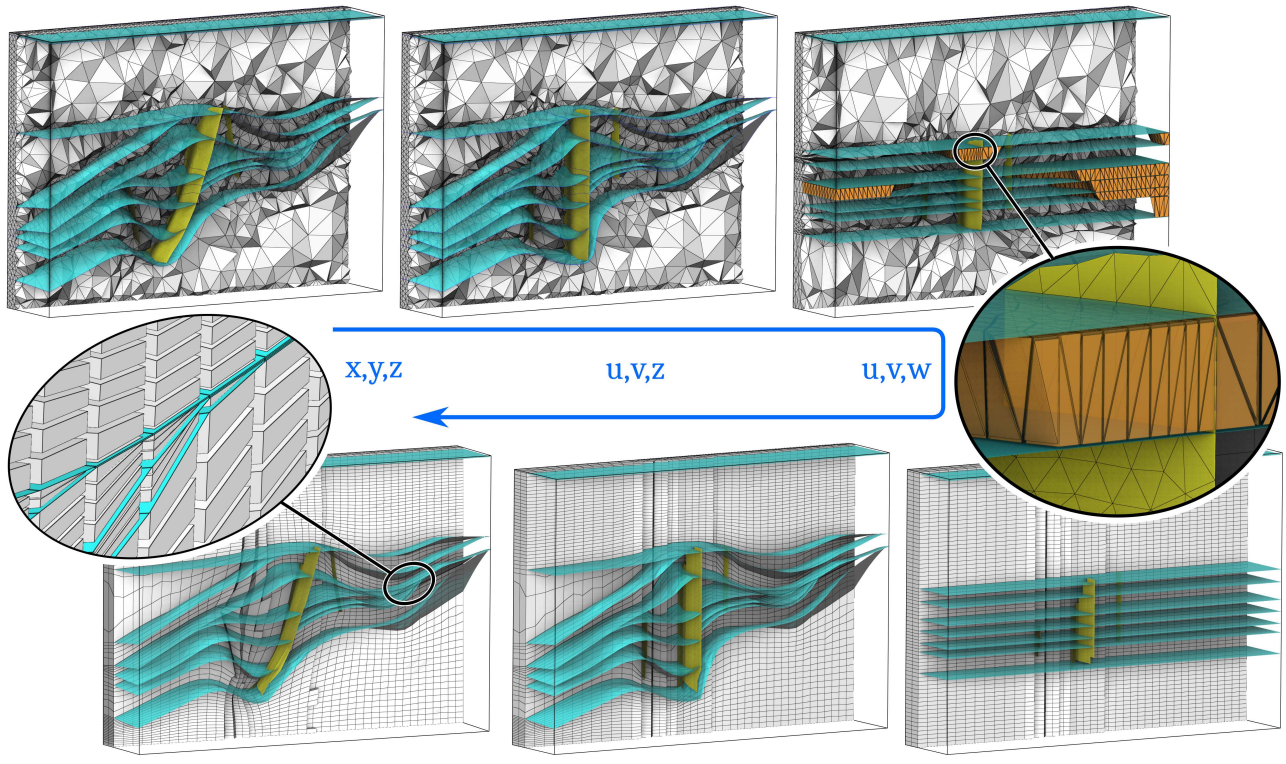

Figure 13: Stratigraphic unconformities given in input (erosion, top left) are compensated by adding cells: they allow to compute $\mathcal{W}$ (close-up on the right). Once the deformation is reversed (bottom left), the volume of the corresponding hexahedra tends to zero.

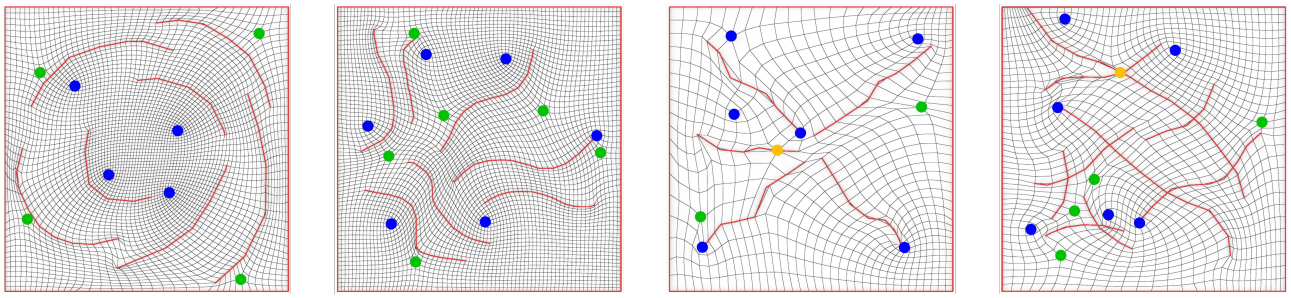

Figure 14: Test cases of artificially generated fault-networks and corresponding quadrangular meshes. Singular points are highlighted in color according to their valence.

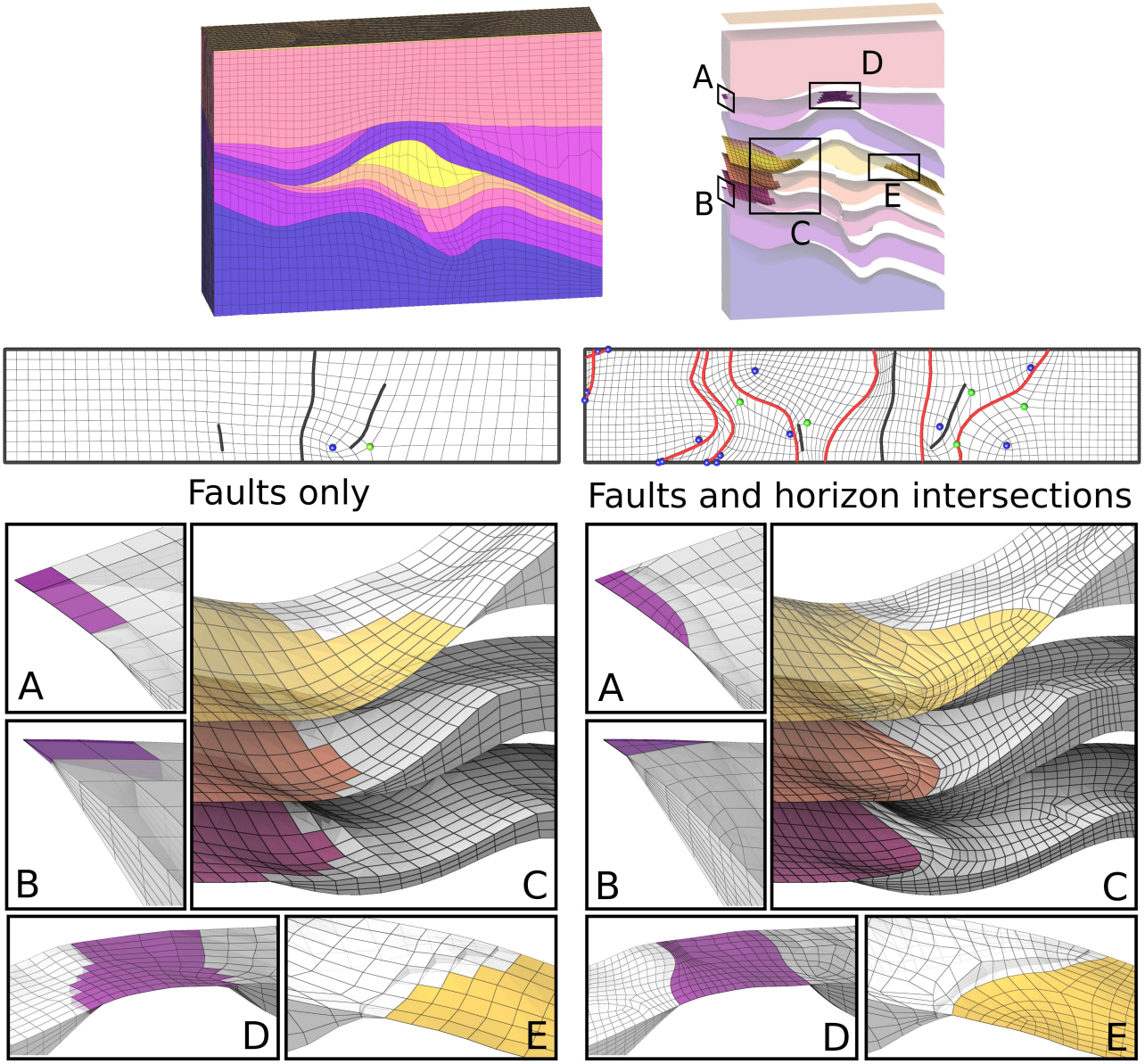

Figure 15: To test the quad mesher and improve the geometric approximation, we attempt to align cells with horizon intersections (so far approximated by stair-step, left). In order to achieve this, we introduce a second set of features to the input of the quad meshing algorithm (right). The resulting quad mesh becomes more complex (20 singular points instead of 2), but it enables the creation of hexahedral elements that precisely conform to the curves of the horizon intersections (close-ups).

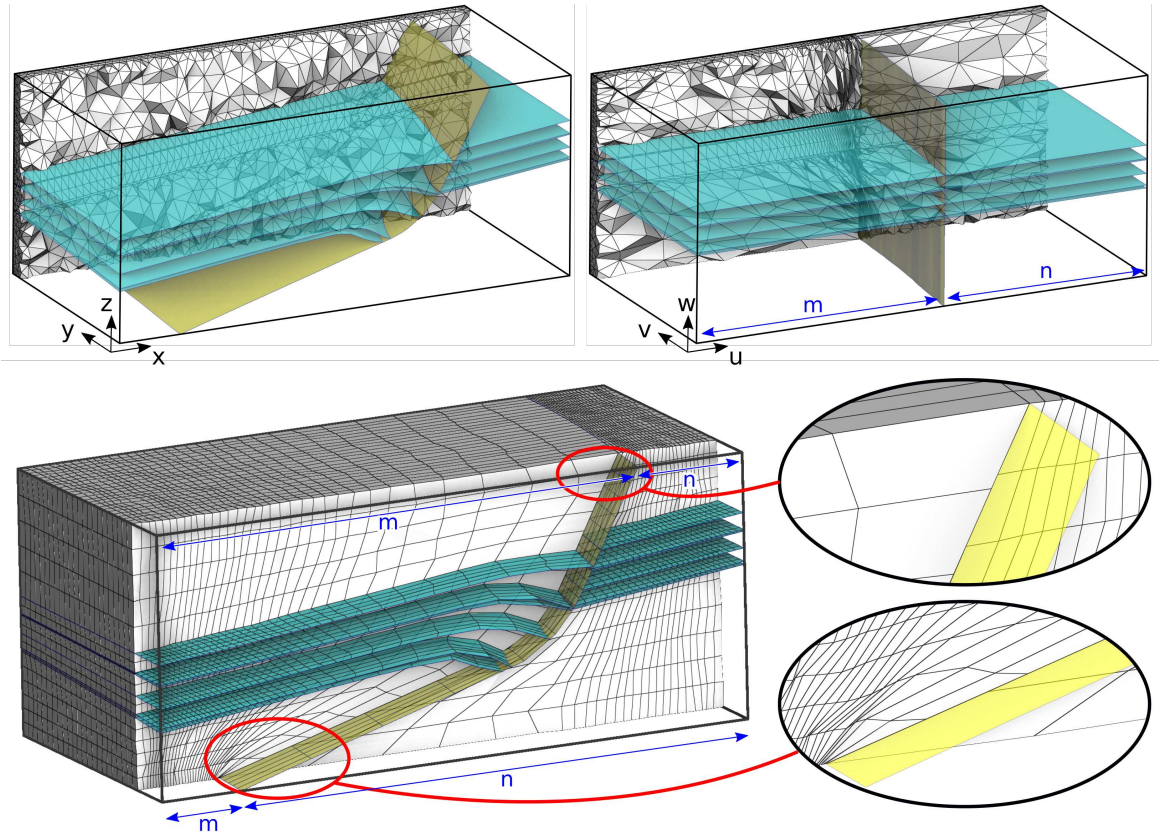

Figure 16: Large deformation stress test: despite undergoing significant deformation (top), the resulting mesh remains valid, free of any flipped elements. Although there is a noticeable variation in element sizes (close-up), this variation appears relatively smooth. Note that the presented result is without the final optimization step. The values of m and n indicate the number of hexahedral elements along the x-axis on either side of the fault.

Furthermore, we demonstrate the robustness of the deformation computation during the verticalization step using our weak-verticality definition, specifically on a model with a listric fault (Sect. 5.1).

5.1 Robustness and efficiency

The deformation $\mathcal{U}$ is obtained by minimizing distortion under weak verticality constraints imposed by the faults. While this approach generally yields a valid deformation without flipped tetrahedra in most cases, there is no guarantee. However, in the Listric stress test (Fig. 16), extreme deformation is required to achieve weak verticality, resulting in flipped tetrahedra. To address this issue, a more robust deformation algorithm proposed by Garanzha et al. [GKK+21] can be employed.

When dealing with sub-horizontal faults using our pillar-based approach, the hexahedral decomposition tends to be of poor quality.

The computational performance of our approach is reasonable, with the results presented here obtained in less than two minutes for input and output meshes on the order of 100K elements using a laptop computer equipped with an i5-11500H CPU running at 2.90GHz. For larger meshes with 1M elements, the computation time ranges from 10 to 15 minutes.

One advantage of our approach is that the number of features has minimal influence on the execution time. However, the most critical phase of the process is the computation of the correspondence between xyz and uvw, and the execution time primarily depends on the magnitude of the distortions. Strong distortions often require the use of an optimization algorithm, as described in [GKK+21]. For example, the stress test in Figure 16 took nearly 10 minutes to compute for an

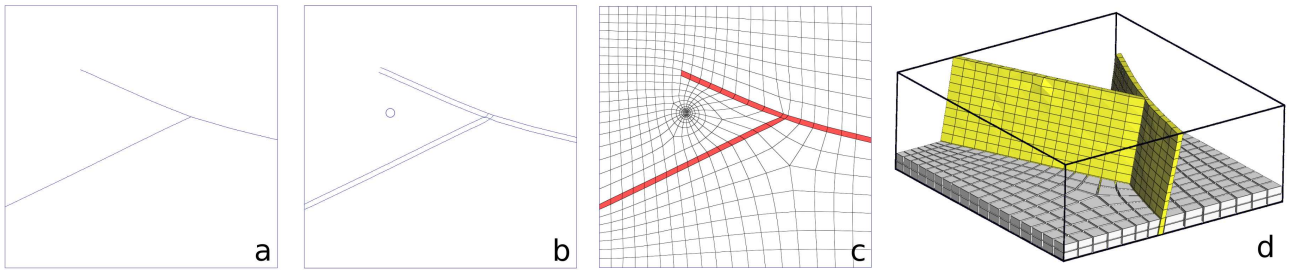

Figure 17: From left to right: fault footprints are duplicated, resulting in a quadrilateral approximation of the fault bottom edges. After extrusion and applying $\mathcal{U}^{-1}$, it results in a volumetric representation of the fault.

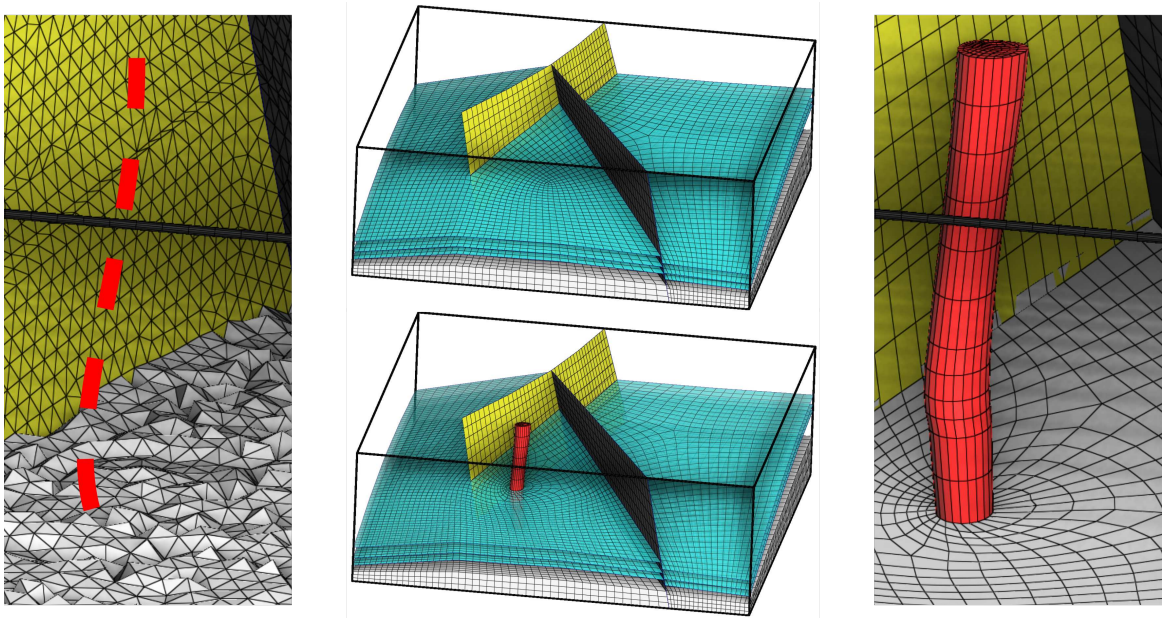

Figure 18: From left to right: well in tetrahedral mesh, classic hexahedral mesh and locally refined mesh around the well (a detailed view on the right.)

input mesh with 160k tetrahedra (40k output hexahedra).

5.2 Local grid refinement

Sometimes, additional grid resolution might be needed near geological features such as faults or artificial objects such as well bores for better studies. Using our method, it is possible to achieve this without the need for refining the mesh globally.

5.2.1 Volumetric faults

Geological faults often have a certain thickness, but in reservoir grids, they are commonly represented as surfaces. However, in cases where faults serve as flow corridors, it is crucial to represent them as volumetric entities, allowing for the assignment of more realistic physical attributes. Our algorithm addresses this by automatically creating volumetric faults in the mesh.
In practice, we achieve this by introducing additional constraints. We duplicate the fault footprint and slightly shift it on both sides of its initial position (by an amount less than or equal to one-third of the expected edge length) in the normal direction (Fig. 17-b). By incorporating these constraints into the input, the quad meshing algorithm typically generates a quad mesh with a single layer

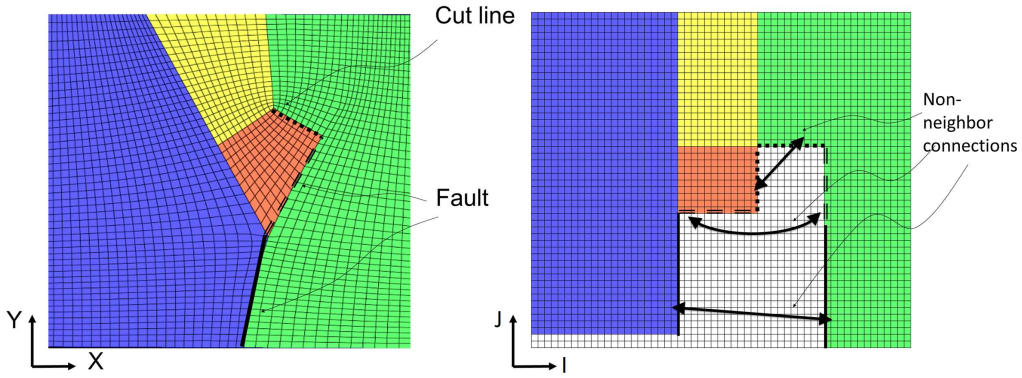

Figure 19: Converting a singular hexahedral grid into a structured grid.

of quads between each pair of features (Fig. 17-c). After layering and applying the inverse deformation $\mathcal{U}^{-1}$, each fault corresponds to a smooth, one-cell-thick vertical layer of hexahedra (Fig. 17-d).

5.2.2 Wells

Furthermore, it is important to consider the geometry around wells and make appropriate adjustments. As mentioned earlier (Sect. 4.2 – Implementation details), we already incorporate wells during the verticalization process to ensure mesh alignment. However, instead of using a single point as the footprint, we propose using a circular shape (Fig. 17-c). By employing a circular footprint, the resulting mesh will exhibit a thinner configuration around the well (Fig. 18).

5.3 Backward-compatibility to standard structured flow simulators

It is important to note that our hexahedral meshes are semi-structured and contain singular edges. While these meshes are compatible with state-of-the-art unstructured grid simulators, they cannot be easily indexed using a simple tuple [I, J, K] in a single index space.

To ensure compatibility with structured flow simulators, we propose partitioning our grid into blocked structured sub-grids, as shown in Figure 19–left. Each sub-grid has its own local index space, which can potentially overlap with other sub-grids. In order to map these local index spaces into a global index space and maintain compatibility with standard flow simulators, it is necessary to minimize the number of unoccupied indices in the global index space (Fig. 19–right). This mapping process is known to be NP-complete. To address this, we have adopted a heuristic similar to the one proposed in [LPRM02] for texture mapping.

After the packing process of the sub-grids, pairs of hexahedra that are geometrically connected or in contact are assigned [I, J, K] indices in such a way that they may not be immediate neighbors in the global index space. For certain standard flow simulators, these geometric contacts need to be explicitly defined as Non-Neighborhood connections (Fig. 19–right).

5.4 Example of application: simulation of fluid injection

The hexahedral grid generated using the proposed method is an excellent choice for conducting fully coupled flow-geomechanics studies as it satisfies the meshing criteria for both types of simulations.

The grid illustrated in Figure 5 is used for running coupled simulations of CO_2 injection using an open-source simulator [GM21]. The synthetic reservoir consists of 8 layers of cells and is located between two layers of overburden and underburden, respectively. Initially, the entire domain is filled with water and set at an initial pressure of 40E05 Pa. During the simulation, we inject a fluid with a CO_2-water volume fraction of 0.995-0.005 at a pressure of 100E05 Pa. Regarding the mechanical constraints, we enforce a condition where the boundaries of the grid have zero displacement along the axis parallel to the surface normal throughout the entire simulation. For

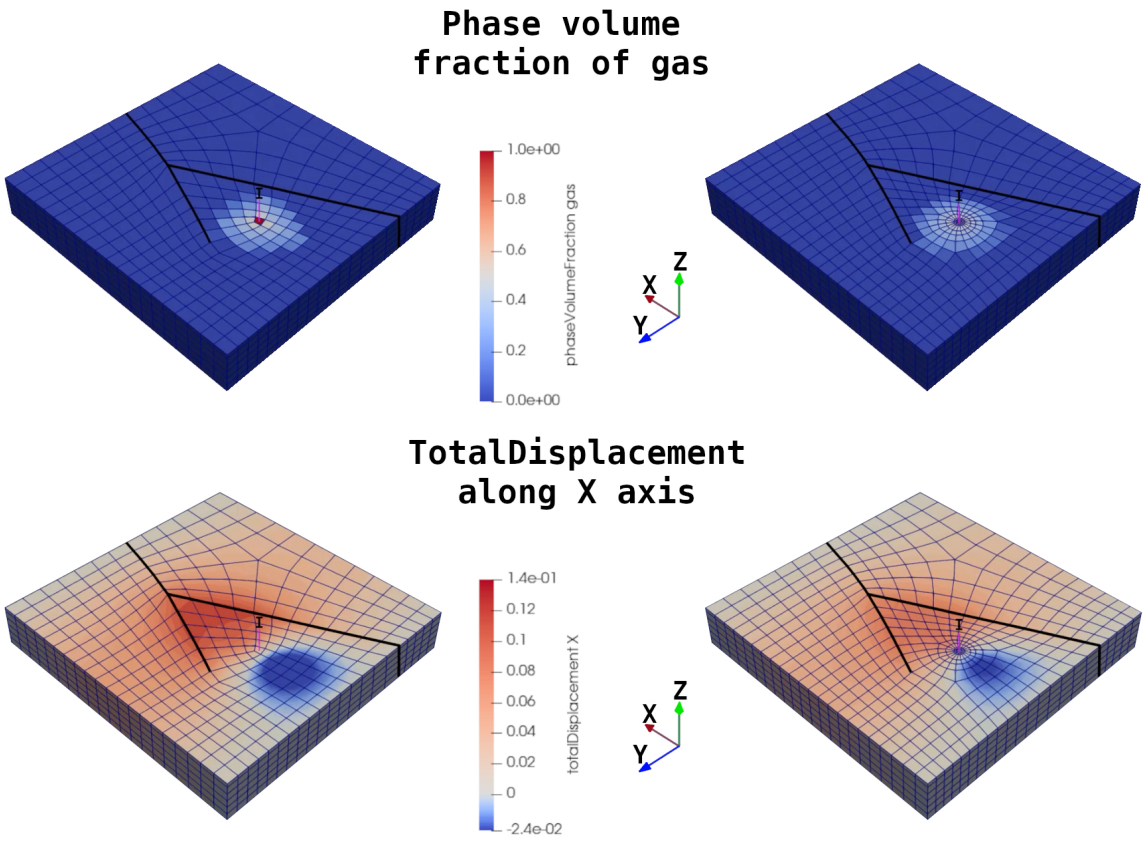

Figure 20: Coupled flow and mechanical simulation results with and without well local refinement. Note that the view is clipped to the middle in the vertical direction.

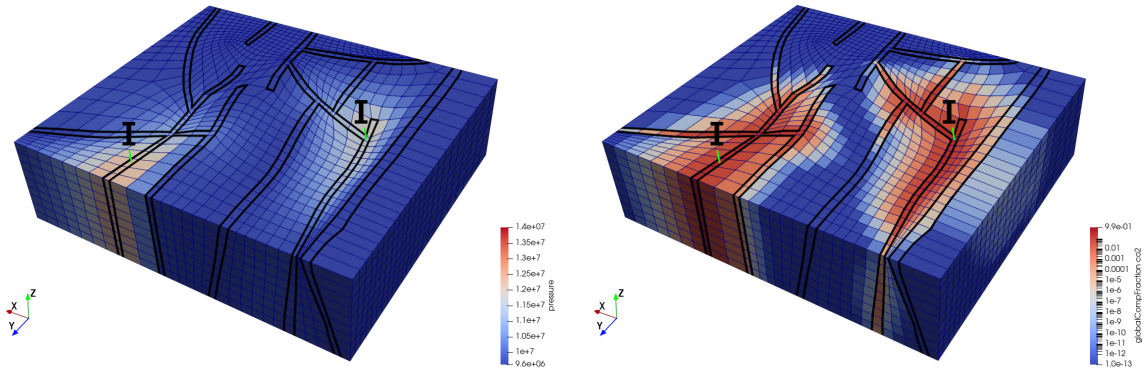

Figure 21: Flow simulation results on CO_2 injection inside fault corridors.

Regions	**Burdens**	**Reservoir**
Rock Type	Shale	Sand
Porosity	0.05	0.25
PERMX (m^2)	1.00E-19	1.00E-14
PERMY (m^2)	1.00E-19	1.00E-14
PERMZ (m^2)	1.00E-19	1.00E-15
Bulk Modulus (Pa)	1.00E+10	7.00E+08
Shear Modulus (Pa)	1.60E+09	4.00E+08
Initial Pressure (Pa)	4.00E+06	
Fault Transmissibility	0.001	

Simulation Time	20 years	
Well Bore Hole Pressure (Pa)		1.0E07
Total CO_2 Injected (Ton)		4.5E05
Injection Rate (Ton/year)		2.25E04

Table 1: Flow and mechanical property settings and simulation results for reservoir CO_2 injection.

instance, the boundaries at x = xMin and x = xMax, represented by a surface in the YZ plane, have a zero displacement condition along the X axis. Detailed property settings can be found in Table 5.4. The simulation results, presented in Figure 20–top, demonstrate the flow and mechanical properties. The CO_2 saturation and pressure exhibit a homogeneous increase around the injection well. The stress also increases, with higher values observed in the two burdens due to their higher bulk and shear modulus. Consequently, higher displacement values are observed in the more deformable reservoir layer, particularly near the well area.

Furthermore, we show similar simulations on a grid with local grid refinement around the injection well, as described in Section 5.2.2. The flow pattern and mechanical response are similar to the case without refinement. The local refinement provides the additional resolution needed to study precise near-well behavior, such as in well testing studies, without the need to excessively refine the entire grid (Fig. 20–bottom).

A second geological model with a more complex fault model (Fig. 14), is used to investigate the injection of CO_2 along fault corridors. In this case, only the results of the flow simulation are presented. The initial pressure and fluid composition are identical throughout the domain. Initially, the model is filled with water in a reservoir at a pressure of 140E05 Pa. The detailed properties of the simulator are specified in Table 5.4. The simulation results, displayed in Figure 21, show the pressure and CO_2 saturation. As expected, the fault corridor, with its greater permeability, serves as a favorable pathway for the injected fluid to flow.

6 Limitations

There are two types of limitations in this work: the first one concerns the formulation of our objective, and the second one relates to the algorithmic aspects involved in optimizing this objective

6.1 Formulation of the objective

Our hexahedral meshes are used to discretize the physical characteristics of geological models in numerical simulations. While we consider the classical mesh quality criteria such as cell deformation, we also need to account for application-specific constraints that play a crucial role in the applicability of our results.

In our current implementation, the mesh must conform across horizons, meaning that it should align with these geological boundaries. Additionally, the mesh should conform in uv coordinates across faults. We relax the conformity constraint on the w component along faults to model the

Regions	**Host rock**	**Fault corridor**
Rock Type	Shale	Sand
Porosity	0.05	0.2
Compressibility	4.00E-10	4.00E-07
PERMX (m^2)	1.00E-15	5.00E-14
PERMY (m^2)	1.00E-15	5.00E-14
PERMZ (m^2)	1.00E-15	5.00E-14
Initial Pressure (Pa)	9.50E06	
Fault Transmissibility	1.0	

Simulation Time	10 years	
Well Bore Hole Pressure (Pa)		1.4E07
Total CO_2 Injected (Ton)	6.12E06	3.77E07
Injection Rate (Ton/year)	5.36E05	3.3E06

Table 2: Flow property settings and simulation results for fault corridor CO_2 injection.

discontinuities in the vertical throw of horizons. However, we do not capture lateral throw as explicit discontinuities in our meshes due to the *uv* conformity constraint. If necessary in the future, we can easily remove this constraint by considering the fault projection as a boundary of the domain to be remeshed with quadrilaterals.

To achieve large block structured meshes, we accept some degenerated cells on feature boundaries. This includes dead cells and pinch-outs on horizon boundaries, as well as stair-step approximations of dying fault boundaries. We have chosen proportional layering over parallel layering to facilitate the coupling of geomechanics and flow simulations. However, it should be feasible to adapt our method for producing parallel layering, potentially through a simple post-processing step in the UVZ space.

6.2 Algorithm limitations

Obviously, the major limitation is that the final mesh is constrained to have a combinatorial structure of a 2D quad mesh extruded in the last dimension. In practice, the verticalization of sub-horizontal faults may produce highly distorted UVZ space, resulting in poor quality meshes, as shown in Fig. 16.

Our method is also limited by the robustness of the XYZ to UVZ mapping algorithm and the quad meshing algorithm. Fortunately, the recent works [GKK+21, DPR+22] we use for these tasks are quite robust. Another possible limitation is that our simple cleanup strategy (section 2.1.1) for generating constraints for the 2D quad meshing problem may be sub-optimal. For example, a dying fault with a zigzag path boundary would produce many short polylines.

Remaining limitations are due to the different samplings of faults in UVW space: a subset of tetrahedra's faces and a subset of hexahedra's faces. For straight faults that are not dying, the geometry perfectly matches in both representations. For curved faults, the geometric deviation is approximately equal to the distance between the projected curve and the edges that sample it in each representation. As faults typically have low horizontal curvature, this geometric difference is likely to be very small. Despite being small, our algorithm has to account for it in three steps:

- In Section 3.1.1, the geometry of faults is approximated by 2D polylines. Considering that faults have low curvature in UV space, the geometric error is never greater than a few percent of the goal hexahedral edge length. If a higher error is observed, a simple refinement of the input mesh would mechanically lower this geometric error.
- The quad meshing algorithm has to conform to the polylines that represent the faults. Exact geometric matching would over-constrain the mesh, so this property is ensured only at quad vertices. Therefore, another geometric approximation is made at this step. Again, the error remains very small with respect to the quad mesh edge lengths.

- In Section 3.3, we take this error into account for the mapping of the hexahedral mesh from UVW space to UVZ space.

In practice, the only algorithmic limitation we observe is that our method is limited to producing a 2.5D mesh combinatorial structure. The reason why other limitations are not observed is that tetrahedral meshes are much easier to generate than hexahedral meshes in commercially available software suites for the subsurface. Therefore, we expect our input to be high-quality geological models in the form of watertight BRep (built using implicit modeling or volume-based modeling approaches [WC18]) and their associated tetrahedral meshes intended for finite-element simulations.

Such meshes have fair dihedral angles and size distributions (not too close to 0 and π) to avoid foldovers in the maps that we compute. As illustrated in Fig. 13, we support enough tetrahedra's scale and shape variations to manage horizons that are very close to each other. Moreover, we cannot expect our algorithm to succeed when there are no 2.5D meshes with acceptable distortion, which is the case for the most challenging scenarios like our Listric stress test.

7 Conclusion

This paper presents a novel approach for building hexahedral grids for reservoirs using an extrusion-based method. The 2D topology of the grid is a quadrangular mesh generated through global parameterization, allowing the mesh to conform to all boundary features by introducing singular vertices. This method significantly expands the range of fault networks that can be handled using structured hexahedral grids without resorting to stair-step fault approximations.

The paper demonstrates promising results from fully-coupled flow and geomechanics simulations using an open-source solver designed for unstructured grids. Additionally, expected results are shown, highlighting that the hexahedral grid with singularities can be used in standard flow simulators after applying the "Packing" treatment.

In the future, the authors plan to further explore various local refinement strategies and consider coarsening the resolution in regions far from features, where reducing the number of cells can improve performance.

Acknowledgements

We thank TotalEnergies and IFP Energies Nouvelles for kindly providing the geological model shown in Figure 13 and 15 as testing case with stratigraphic unconformities.

References

[BLC16] Arnaud Botella, Bruno Lévy, and Guillaume Caumon. Indirect unstructured hex-dominant mesh generation using tetrahedra recombination. volume 20, pages 437–451, 06 2016.

[BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77, 2009.

[BZK12] David Bommes, Henrik Zimmer, and Leif Kobbelt. Practical mixed-integer optimization for geometry processing. In *Proceedings of the 7th international conference on Curves and Surfaces*, pages 193–206, Berlin, Heidelberg, 2012. Springer-Verlag.

[CBK15] Marcel Campen, David Bommes, and Leif Kobbelt. Quantized global parametrization. *ACM Trans. Graph.*, 34(6), oct 2015.

[Cra19] Keenan Crane. The n-dimensional cotangent formula. 2019.

[DPR+22] David Desobry, François Protais, Nicolas Ray, Etienne Corman, and Dmitry Sokolov. Frame Fields for CAD models. *Lecture Notes in Computer Science*, 13018:421–434, January 2022.

[EBCK13] Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. QEx: Robust Quad Mesh Extraction. *ACM Transactions on Graphics*, (4):168:1–168:10, 2013.

[Fre02] N. Fremming. 3d geological model construction using a 3d grid. 09 2002.

[GHAN09] Emmanuel Gringarten, Aymen Haouesse, Burc Arpat, and Long Nghiem. Advantages of Using Vertical Stair Step Faults in Reservoir Grids for Flow Simulation. In *Proc. SPE Reservoir Simulation Conference*, 2009.

[GKK+21] Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. Foldover-free maps in 50 lines of code. *ACM Trans. Graph.*, 40(4), jul 2021.

[GLV+17] Emmanuel Gringarten, Jean Daniel Lecuyer, Elsa Villarubias, Camille Cosson, and Wan-Chiu Li. Optimized Grids for Accurately Representing Geology in Geomechanical Simulations. In *Proc. SPE Annual Technical Conference and Exhibition*, 2017.

[GM21] Herve Gross and Antoine Mazuyer. GEOSX: A Multiphysics, Multilevel Simulator Designed for Exascale Computing. In *SPE Reservoir Simulation Conference*, page D011S010R007, On-Demand, October 2021. SPE.

[HNK03] K. Hoffman, J. Neave, and R. Klein. Streamlining the workflow from structure model to reservoir grid. 10 2003.

[KBL+17] Ø. S. Klemetsdal, R. L. Berge, K. A. Lie, H. M. Nilsen, and O.. Møyner. Unstructured Gridding and Consistent Discretizations for Reservoirs with Faults and Complex Wells. In *Proc. SPE Reservoir Simulation Conference*, 2017.

[KNP07] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover - surface parameterization using branched coverings. *Comput. Graph. Forum*, 26(3):375–384, 2007.

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In ACM, editor, *ACM SIGGRAPH conference proceedings*, Jul 2002.

[Mal02] Jean-Laurent Mallet. *Geomodeling / Jean-Laurent Mallet.* Applied geostatistics series. Oxford University Press, Oxford ;, 2002.

[MLC11] Romain Merland, Bruno Lévy, and Guillaume Caumon. Building PEBI Grids Conforming To 3D Geological Features Using Centroidal Voronoi Tessellations. In *Proc. IAMG*, 2011.

[MSV+13] Bradley Mallison, Charles Sword, Thomas Viard, William Milliken, and Amy Cheng. Unstructured Cut-Cell Grids for Modeling Complex Reservoirs. *SPE Journal*, 19, 2013.

[PA94] C. L. Palagi and K. Aziz. Use of voronoi grid in reservoir simulation. *SPE Advanced Technology Series*, 2:69–77, 1994.

[PCJ+15] Jeanne Pellerin, Guillaume Caumon, Charline Julio, Pablo Mejia-Herrera, and Arnaud Botella. Elements for measuring the complexity of 3d structural models: Connectivity and geometry. *Computers & Geosciences*, 76:130–140, 2015.

[RVLL08] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2), may 2008.

[SHK+18] Samita Santoshini, S. Harris, Sheleem Kashem, Arnaud Levannier, Azeddine Benabbou, Thomas Viard, and Laetitia Macé. Depogrid: Next generation unstructured grids for accurate reservoir modeling and simulation. 10 2018.

[Si15] Hang Si. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Transactions on Mathematical Software*, 41:1–36, 2015.

[VCD+17] Amir Vaxman, Marcel Campen, Olga Diamanti, David Bommes, Klaus Hildebrandt, Mirela Ben-Chen Technion, and Daniele Panozzo. Directional field synthesis, design, and processing. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, New York, NY, USA, 2017. Association for Computing Machinery.

[WC18] Florian Wellmann and Guillaume Caumon. Chapter one - 3d structural geological models: Concepts, methods, and uncertainties. volume 59 of *Advances in Geophysics*, pages 1–121. Elsevier, 2018.