MeshVoro: A three-dimensional Voronoi mesh building tool for the TOUGH family of codes

C.M. Freeman [a,*], K.L. Boyle [b], M. Reagan [b], J. Johnson [b], C. Rycroft [c], G.J. Moridis [b]

[a] Earth Sciences Division, Lawrence Berkeley National Laboratory, now with Hilcorp Energy Company, 1201 Louisiana St $1400, Houston, TX, 77002, United States
[b] Earth Sciences Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, United States
[c] Harvard University, School of Engineering and Applied Sciences, Cambridge, MA 02138, United States

ARTICLE INFO

ABSTRACT

Few tools exist for creating and visualizing complex three-dimensional simulation meshes, and these have limitations that restrict their application to particular geometries and circumstances. Mesh generation needs to trend toward ever more general applications. To that end, we have developed MeshVoro, a tool that is based on the Voro++ (Chris H. Rycroft, 2009. Chaos 19, 041111) library and is capable of generating complex three-dimensional Voronoi tessellation-based (unstructured) meshes for the solution of problems of flow and transport in subsurface geologic media that are addressed by the TOUGH (Pruess, K., Oldenburg C., Moridis G., 1999. Report LBNL-43134, 582. Lawrence Berkeley National Laboratory, Berkeley, CA) family of codes. MeshVoro, which includes built-in data visualization routines, is a particularly useful tool because it extends the applicability of the TOUGH family of codes by enabling the scientifically robust and relatively easy discretization of systems with challenging 3D geometries.

We describe several applications of MeshVoro. We illustrate the ability of the tool to straightfor-wardly transform a complex geological grid into a simulation mesh that conforms to the specifications of the TOUGH family of codes. We demonstrate how MeshVoro can describe complex system geometries with a relatively small number of grid blocks, and we construct meshes for geometries that would have been practically intractable with a standard Cartesian grid approach. We also discuss the limitations and appropriate applications of this new technology.

© 2014 Elsevier Ltd. All rights reserved.

1. Unstructured simulation meshing

Problems of modern interest in the subsurface sciences increasingly involve heterogeneities of flow parameters and complex geometrical challenges. Examples of systems with significant heterogeneities include reservoirs with faults and fractures, induced and naturally occurring. Examples of systems with innate geometric complexity include systems involving wells, fractures, geological strata and faults, which may be intersecting at arbitrary angles. Few tools are available which allow flexible mesh generation for arbitrary three-dimensional geometries. TOUGH, as well as other codes capable of solving differential equations over irregular and unstructured meshes, have solved problems involving grids obtained by the "two-and-a-half dimensional" (2-1/2D) Voronoi tessellation. This term describes the practice of performing a two-dimensional Voronoi tessellation of locally refined mesh on the X–Y plane, and then projecting the computed mesh downward into the Z-axis in horizontal layers, modified in the Z-dimension to follow the contours of the geological

layers and to respect discontinuities caused by faults (Palagi, 1992; Verma, 1996). This approach results in computational savings, but it is generally limited to creating more efficient refinements over essentially two-dimensional problems, and, thus, can only describe accurately geological systems that are uniformly layered (stratified).

With the advent of ever more powerful computers and advanced computer language capabilities, previously intractable problems are within reach. Thus, the problems of flow and transport that are currently under consideration involve not only advanced coupled processes, but also intricate three-dimensional geometries and extreme heterogeneities.

The TOUGH (Pruess et al., 1999) family of codes, including the new generation involving the TOUGH+ architecture (Moridis et al., 2008), uses the integral finite difference method (IFD— Edwards, 1972; Narasimhan and Witherspoon., 1976, 1978) in its spatial discretization. This is a finite volume formulation that allows the use of irregularly shaped cells (unstructured grids). In other words, individual grid blocks in IDF may assume any shape and have any number of connected neighbors, so long as certain connectivity conditions are met, which are discussed shortly. This attribute allows TOUGH applications to use Voronoi tessellation-based mesh solutions. Voronoi grids, sometimes referred to in the

literature as Perpendicular Bisection (PEBI) grids, have been used by several practitioners in the petroleum engineering industry (Verma, 1996, Merland et al., 2011). Such, grids offer unique advantages in the description of domains with complex (especially 3D) geometries. Cells can be constructed to conform to geometric features using variants of Voronoi tesellation (Du and Gunzburger, 2002, Merland et al., 2011), in which case they may yield significantly more accurate solutions at a lower number of grid-blocks than algorithms relying on uniform or rectilinear structured grids. In this sense, the availability of easily generated Voronoi-based grids would expand the usefulness of the TOUGH family of codes by enabling the solution of previously intractable problems.

To provide a specific example, take a thin volume of geologic media containing part of a horizontal well with radius 0.05 m, which is itself transversely intersected by a fracture with dimensions of 0.001 m internal thickness, 50 m half-length and 10 m height. In other words, an object with required refinement in the x, y, and z dimensions. By taking advantage of the geometric arrangement of these objects we can express this relatively complex object in detail with 27197 grid elements, down to the interface of the fracture with the well's radius, with the assumption that the well can be approximated with a 16-sided polygon. Capturing a similar level of mesh fidelity with a rectilinear grid would require refinement to be propagated in the x, y, and z dimensions, resulting in at least 300,000 rectilinear elements and potentially orders of magnitude more, depending on the desired precision at the fracture–wellbore interface. With this example we demonstrate a key objective in this work: to provide a means of switching from one paradigm of gridding at the bulk scale to a more appropriate type of gridding at the local scale.

The first connectivity condition for a finite volume discretization is that the surface associated with a given connection must be normal to the straight line connecting the cell centroids (centers of gravity). This requirement is met by definition by any centroidal Voronoi tessellation (Du et al., 1999), where a centroidal Voronoi tessellation is defined as one where the generator points are identical to the centroids. Every facet of a Voronoi cell is mathematically defined as being normal to the line connecting two neighboring points. Voronoi tessellations possess the added attribute that the common interface is equidistant from those two connected points. In Voronoi tessellations, the tessellation generator points are not necessarily identical to the element centroids. In other words, Voronoi tessellations may not necessarily be centroidal, and thus do not strictly satisfy the requirements for accuracy of a finite volume spatial discretization. We discuss the consequences of using non-centroidal Voronoi meshes in Section 4. MeshVoro includes a module that processes the generated mesh to guarantee that the mesh is watertight, i.e., that the mesh does not possess connections which should not be present and is not missing connections which should be present. This post-processing essentially verifies the correctness of each interface.

2. Algorithmic innovations

In the course of developing and applying MeshVoro to a variety of problems, we faced many unique challenges. Generally, these challenges were related either to the visualization of the simulation results, or to enforcing sufficient smoothness of the mesh to promote numerical tractability.

2.1. Tetrahedralization

A perhaps surprising obstacle to the general use of Voronoi meshes, and unstructured meshes made up of arbitrary polyhedra in general, is the lack of robust and flexible tools for visualizing the

meshes. A basic requirement of any simulation study is the capability to view the state of a simulation at any point in time in a spatially intuitive way for diagnostic and analytical purposes. However, it is typical for even the most cutting-edge visualization software packages to support rendering of only basic shapes, such as tetrahedra and hexahedra, but not of arbitrary polyhedra.

In order to address the issue of visualization, we subdivide the polyhedral Voronoi mesh into tetrahedra. We rely on the fact that any convex polyhedron can be broken into a number of tetrahedra equal to the number of edges of the original polyhedron by following a relatively straightforward algorithm (Chazelle, 1980): For each edge of the Voronoi polyhedron, we begin with the two points defining the beginning (Point 1) and the end (Point 2) of the edge. Then, recalling that each edge constitutes the intersection of two faces, we define Point 3 as the center of one of the faces adjacent to that edge. Finally, we define Point 4 as the centroid of the Voronoi cell. These four points define a tetrahedron. A cube, or hexahedron, having twelve edges, will be subdivided into 24 tetrahedra by the iterative application of this algorithm. The geometries of two such tetrahedra are depicted in Fig. 1.

If the points defining these tetrahedra are stored in the memory during the computations for mesh generation, they can be used to generate output files in standard formats such as .vtk (Schroeder et al., 1998). Thus the tetrahedralization of the original Voronoi mesh can be visualized. The data reflecting the simulation state can be visualized by color-coded mapping onto the mesh elements. This allows us to robustly visualize the state of the simulation system at any point in time. The costs and benefits of retaining these tetrahedra in memory are discussed in the next section.

In a graph-theoretic sense, the Voronoi mesh is the dual of the Delaunay tetrahedralization for the generator points (Florack et al., 2012). Note that the Delaunay tetrahedralization is not the tetrahedralization algorithm employed by MeshVoro. In the MeshVoro tetrahedralization algorithm, the interior volume of each tetrahedron belongs strictly to one Voronoi cell, whereas the volume of a given tetrahedron in the Delaunay tetrahedralization will overlap partially into several different Voronoi cells.

2.2. Extension to very large meshes

In the course of developing and applying the MeshVoro tool to different research problems, we discovered that some simulation meshes are too large to be handled even on powerful modern computers without special tools. A direct decomposition of a mesh consisting of 2264,000 elements (discussed in more detail in
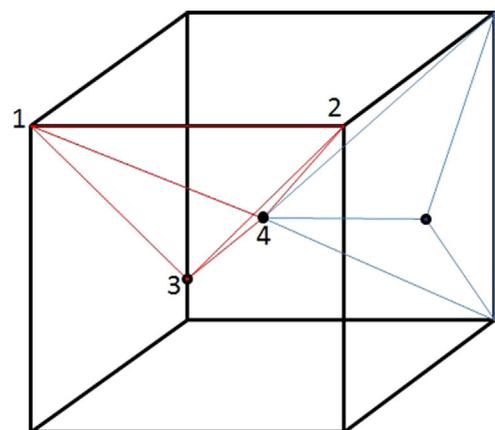
Fig. 1. A cube (hexahedron) is partially decomposed into tetrahedra. Points 1 and 2 represent the two ends of a single edge; Point 3 represents the center of one of the faces bordered by that edge; and Point 4 represents the centroid of the Voronoi element. These four points make up a single tetrahedron.

Section 3.1) into tetrahedra would result in nearly 100 million tetrahedra. Merely storing the relevant metadata needed to express these 100 million objects results in a 22 GB file. Performing the operations necessary for visualization on files of this size requires infeasible amounts of computer memory and processing power.

Using a combination of Python, C and C++, we wrote a special suite of modules that allow a user to specify in a pre-processing step which regions of the mesh should be selectively visualized. This special suite of the MeshVoro code sets aside only the tetrahedra which make up this selected region, then extracts the relevant simulation state data from hundreds of output files generated by massively parallel TOUGH+ runs, each representing one portion of the mesh, and collates the selected data into a more compact file format. Finally a .vtk (Schroeder et al., 1998) output file is generated from the extracted data.

2.3. Smoothing of the point distribution

In general, the intersections and edges of discrete three-dimensional objects pose a significant challenge to accurate mesh building. For example, the point at which the wells depicted in Fig. 5 pass through the overlying interface between geological strata must preserve both the continuity of the well and the continuity of the surface plane of the geological layer. Creating a mesh for the region of intersection between these two objects in a way that respects the continuity of the objects and the physics of the flow simulation is not straightforward.

Another difficulty is posed in cases where a very finely discretized subdomain (often referred to as mesh object) is adjacent to a subdomain that does not require fine discretization. In the confluence (contact) of these two subdomains, a problem will arise because the Voronoi mesh generation algorithm connects each of the coarse grid points with tens or hundreds of neighboring points on the refined object. If this is allowed to occur, it can result in a significantly less sparse Jacobian matrix at simulation time, which may lead to poor solver performance.

A standard technique for improving mesh smoothness in unstructured meshes is Lloyd's (1982) iteration. The Lloyd iteration is a method for smoothing highly anisotropic Voronoi grids by shifting the generator points of the Voronoi cells to the centroids of the Voronoi cells, and then recomputing the Voronoi tesselation based on the new generator points. In the extreme case, this iteration is continued until every Voronoi cell's centroid coincides with its generator point. This degree of smoothing would nullify any local refinement, so it is not appropriate for our purposes. In MeshVoro, we provide an option to select the number of desired Lloyd iterations to be performed. MeshVoro provides an additional option to attach special flags to particular points or regions of points designating them as 'movable,' where by default the generator points will not be updated. This allows selective smoothing of particular regions of the mesh.

We determined that this technique comes with drawbacks that can restrict (potentially severely) its application. Primarily, in all of the applications we discuss in this study, there are numerous 'fixed' objects such as wells, boundary layers, and insulators that are effectively destroyed by applying Lloyd's iteration to them. Applying Lloyd smoothing to the points surrounding or between refined objects can also yield inconsistent results.

Several alternatives to Lloyd iteration exist in the literature. Centroidal Voronoi Tessellation (CVT) is a research topic of considerable interest, and has been explored by several authors (Du et al., 1999, Merland et al., 2011, for example), and brings the mathematics of optimization theory to bear on the point smoothing problem. At lowest order, CVT can reproduce Lloyd iteration, but the definition of an objective function can be

enriched to accommodate anisotropy in a geological model, as well as special considerations on the boundary of the domain. Point density functions can also be defined to smooth points in a way that produces regions of gradually different resolutions, though designing properly parametrized density functions is something of an art form.

Another technique that has been recently demonstrated is a "force-equilibrium" approach described by Persson (2005). Here the author generates meshes using Delaunay triangulation, balancing forces aligned with the edges of the triangles to produce smooth node distributions that respect density functions similar to those described for CVT by Du et al. (1999). He also describes methods for producing anisotropic cells that conform to boundaries on the exterior and within the interior of domains. Persson has made available a MATLAB code, *DistMesh*, that demonstrates some of these capabilities (Persson and Strang, 2004).

In any case, MeshVoro separates the process of defining the mesh points from the tessellation, allowing any of these techniques to be implemented in Python, producing a set of points that can then be fed to the Voro++ tessellator.

2.4. Strategies for mesh object integrity

If the techniques of the previous section cannot be applied, the default option for preserving the geometric integrity of specific mesh objects (subdomains) – such as wells and fault planes – is to carefully hand-code a purpose-built algorithm which places points exactly where they are needed, so as to conform to the geometric requirements of all the objects involved in the mesh. This can be extremely time-consuming and the solutions that are programmed in this way are typically not re-usable in other contexts, owing to the intricacy of fitting together multiple three-dimensional objects in the same region of space.

The general strategy behind implementing specific mesh objects in this fashion is to approximate the desired geometric shapes using regular point clouds. For example, a well consists of stacked discs of points. A fracture, depending on the desired configuration, consists of layers of regular 2D grids of points. Multiple layers in all dimensions are needed to ensure adequate mesh smoothness. While technically a well could be approximated with a cylinder consisting of a double-ring of points, this configuration would be insufficiently smooth; the volume of the elements comprising the inside of the well would be vastly smaller than the volume of adjacent elements in the mesh outside the well. The term used for this transition region populated by increasingly widely spaced grid elements is the "fade zone."

There are a number of heuristics that we determined in the course of developing many such tailor-made meshing solutions. These heuristics are the result of numerous personal experiments and are provided in the interest of avoiding pitfalls such as insufficiently smooth meshes, inconsistent behavior, or over-discretization resulting in numerical problems.

1. It is advisable, in order to achieve consistent smoothness, to provide a fade zone surrounding any refined object (typically a fracture or well), with a thickness greater than twice the distance between the points representing the refined object in the least refined dimension.
2. Treating the inside of a wellbore as a ring of points rather than a single central point results in significant numerical overhead and provides no additional physical insight; thus, single central points for wells are to be used.
3. When creating complex objects involving well-fracture continua (such as wells intersected by transverse fractures), it is recommended to use a radial geometry to represent the fracture near the wellbore, and to transition to a rectilinear

geometry at a radius greater than twice the distance between the well rings.

4. A relatively reliable method to ensure that the continuity of geological layers in the mesh is not marred by stray points belonging to nearby mesh objects is to 'mirror' all the points that are closest to one side of the plane which defines the layer interface (i.e., in the first "stratum" of such points) onto the other side of that plane, normal to the plane in both directions. This cannot be applied if any other mesh objects (subdomains) intersect the geological layer at a non-right angle.

5. Attempting to blend one mesh object fade zone into another by 'feathering' the two fade zones (i.e., by allowing the fade zones to sparsely intermingle) is strongly discouraged. Hard cutoffs between fade zone interfaces are recommended.

6. As a post-processing step, MeshVoro provides a means of removing elements outside of a predetermined area, which may be a complex shape. Consequently, while the original generator cloud input must be a box, the boundaries of the system output by MeshVoro may be non-box or non-convex. Particular attention should be paid to the behavior of elements at the periphery of the boundary, to ensure that the boundaries of exclusion are set and performing as intended.

7. If possible, it is a good practice to visually inspect every new mesh that is generated for unexpected meshing behavior. Intuition can be a poor guide with regards to which points will end up connected and how surfaces will be realized.

3. Methods and applications of MeshVoro

The Voro++ (Rycroft, 2009) library possesses a large variety of features not used for the purposes of this work. We primarily rely on the capability of the library to import large numbers of grid points or cell centers, and to rapidly compute the Voronoi tessellation that conforms to these generator points. Furthermore, the Voro++ library enables direct access to all the relevant details of the Voronoi cells computed from these grid points, such as volumes, surface areas, reference indices of neighboring cells, and distances from each point to each surface. This information is sufficient to produce a valid (finite volume) TOUGH mesh in the appropriate format.

Prior to the development of MeshVoro, there was no straightforward method to transform a complex geological model (such as one obtained from seismic inversion) into a TOUGH mesh. With MeshVoro, the geological model can be directly converted into a TOUGH mesh that respects the cell centers of the original geological data points.

3.1. Application to a complex hydrate problem

We applied the MeshVoro code to the geophysically-obtained geological model in a project involving the study of gas production from offshore gas hydrate deposits. The geological model was received as a field of points in space associated with geophysical
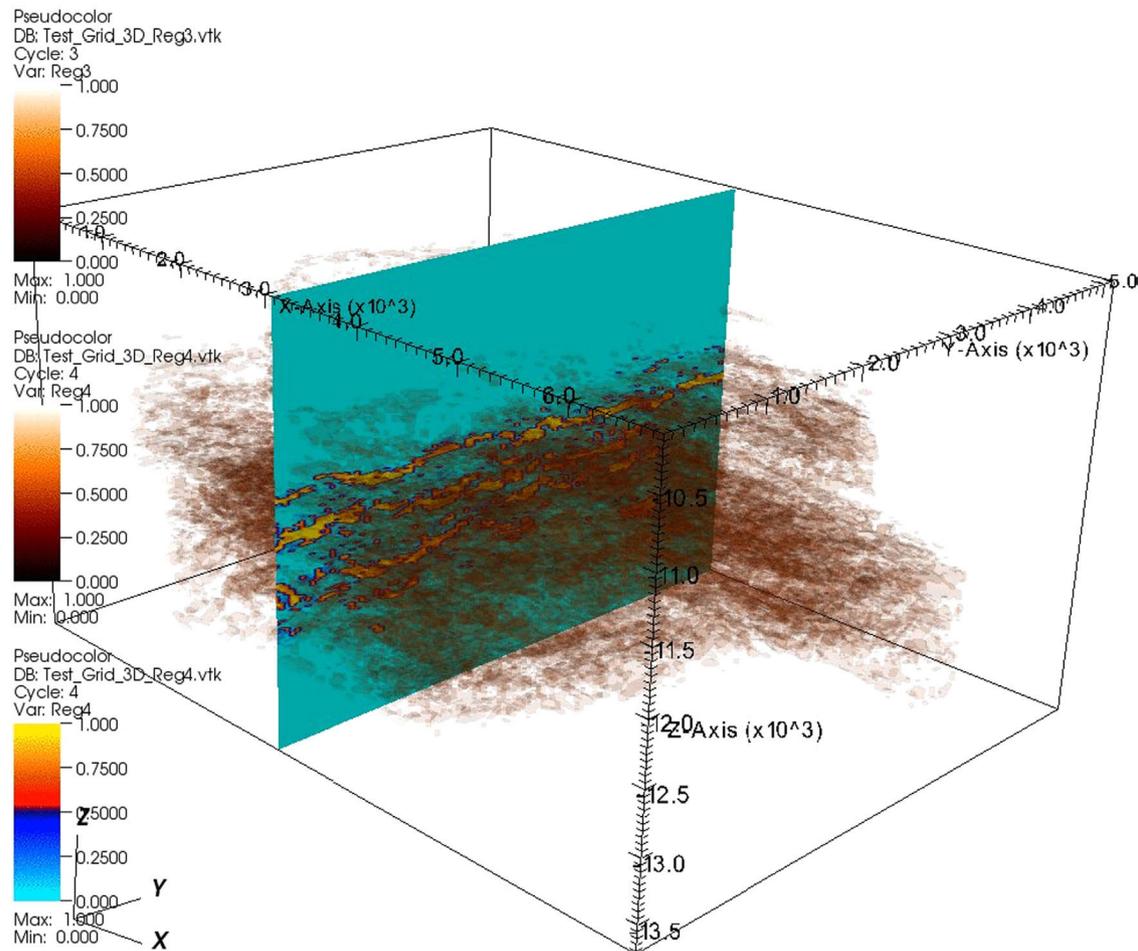
Fig. 2. The geological model of 2264,000 elements shown here was transformed directly into a Voronoi mesh using MeshVoro. This is the largest grid ever developed for any TOUGH simulation up to the time of preparation of this paper.

data. In addition to creating a TOUGH mesh from the provided geological model, we also needed to represent a horizontal well that was to be positioned at an appropriate location in the system.

The appropriate location for the horizontal well was determined by using a search algorithm to find the longest continuous interval of hydrate-bearing media possessing adequate gross thickness in the entire domain. This was done with the aim of maximizing the productivity of the well.

The points lying within a 250 m radius of the desired horizontal well location were removed from the mesh. The region was then repopulated with a cylindrically refined region of points, with radially logarithmic grid spacing, to optimize smoothness, increasing from a few centimeters near the wellbore to the spacing of the background mesh at the extent of the cylindrical region. In the axis of the horizontal well's penetration (the y-axis) the discretization used was the original discretization of the background geologic mesh. The points in this cylindrically refined subdomain inherited their geological material designation and initial state from the computed nearest-neighbor point in the original geologic data (see Fig. 2).

This mesh is the largest grid ever generated for a TOUGH simulation up to the time of preparation of this paper, involving 2264,000 elements and requiring the solution of over 9000,000 simultaneous coupled equations (Figs. 3 and 4).

The code modules developed for visualizing very large meshes (see discussion in Section 2.2) were written specifically for this highly refined field-scale hydrate mesh. Visualizing the results of the simulation required more human and computational time investment than generating the simulation mesh had demanded in the first place.

3.2. Application to a problem involving wells with complex 3D orientation

The next application of the MeshVoro code involves the relatively free-form generation of meshes conforming to complex, yet somewhat idealized, geometries, as motivated by a study of the potential for the impact of hydraulic fracturing of deep tight/shale gas reservoirs on shallow drinking water aquifers. For example, the creation of a TOUGH mesh conforming to a system possessing two vertical wells and a horizontal well each passing through multiple distinct geological layers would have been prohibitively difficult if we had attempted to use a structured (standard finite difference) grid because of the requirement of high degrees of refinement in all dimensions. With the MeshVoro tool, it becomes possible to straightforwardly construct objects such as wells, fractures and layers from cylindrical and planar fields of points that are carefully assembled into a point cloud. The point cloud is then transformed into a Voronoi mesh by the main MeshVoro code (see Fig. 5).

MeshVoro attempts to address these issues with three options, the suitability and appropriateness of each of which depend on the particular circumstances.

## 3.3. Application to meshing a hydrate core sample

While TOUGH and similar codes for modeling fluid flow in porous media are normally used to simulate large-scale processes in geologic media, the TOUGH codes can in fact render accurate results for a domain of any size, so long as the underlying basic equations of flow in the available options (i.e., Darcy's law, the Forchheimer, 1901, and the Barree and Conway, 2004, 2007
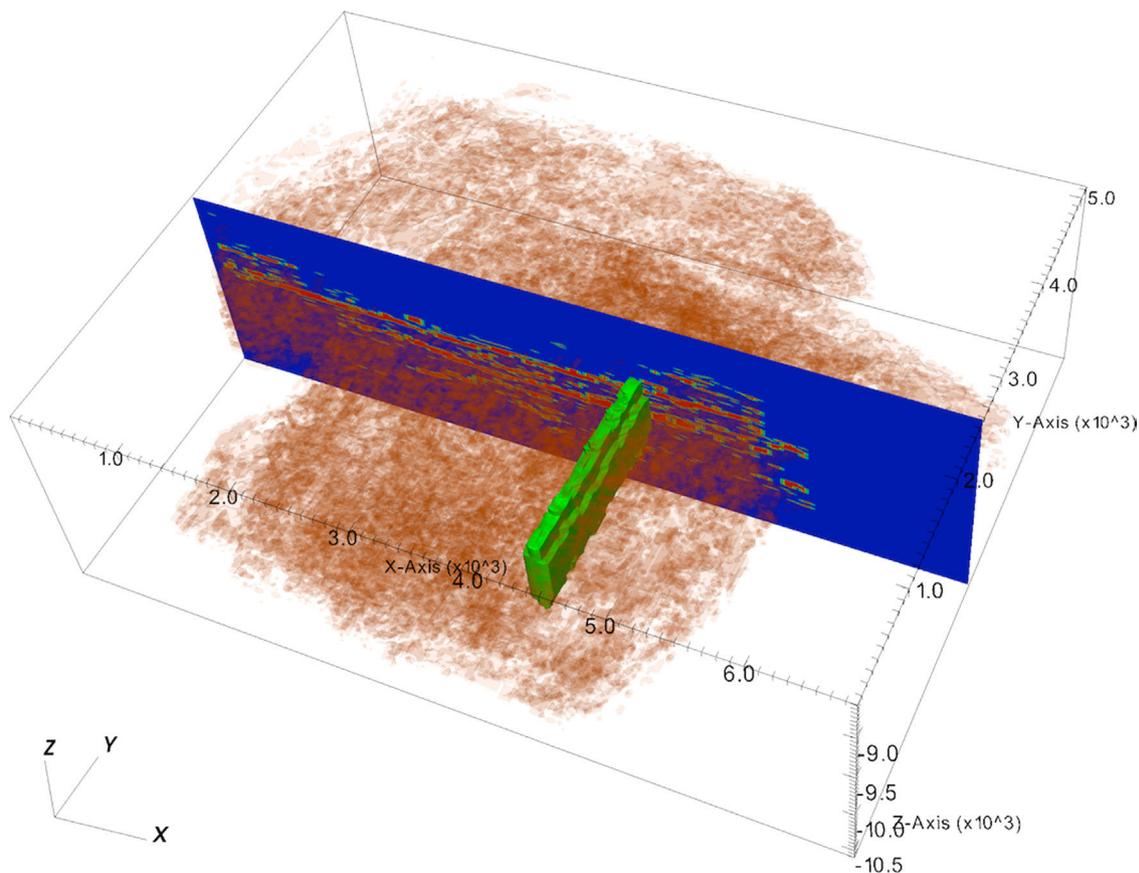
Fig. 3. A horizontal well is inserted into the geological model in this figure. The properties of the 'fade zone' a (the green region) around the horizontal well are inherited from the properties of the nearest-neighbor points in the original grid. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
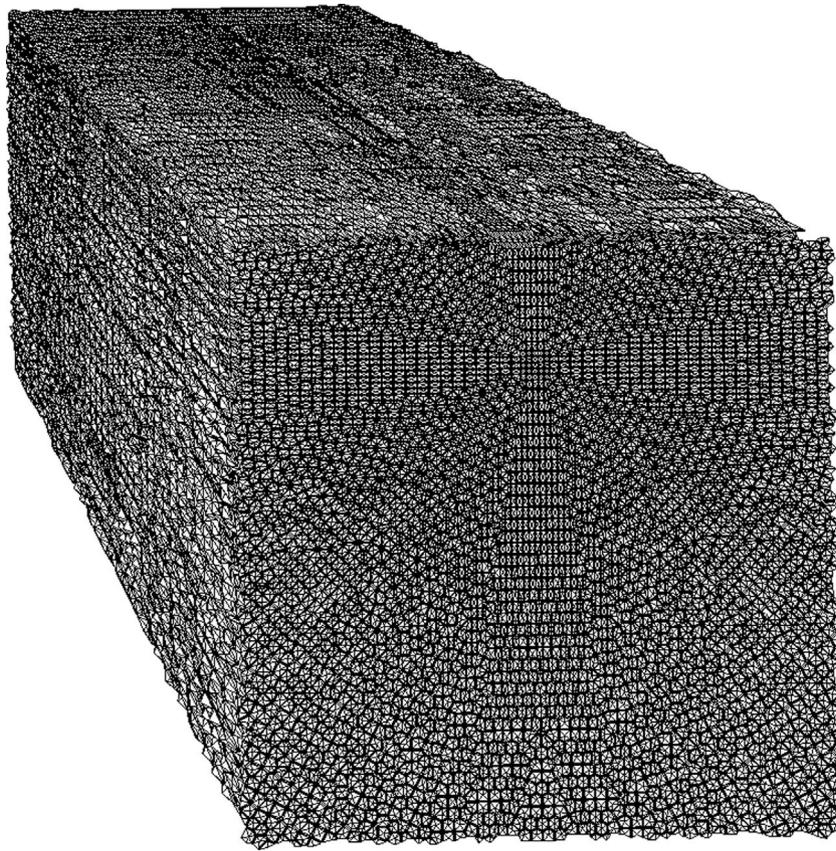
Fig. 4. A view of the tetrahedralization of the region surrounding the horizontal well, processed selectively using the tools designed specifically for manipulating very large meshes.
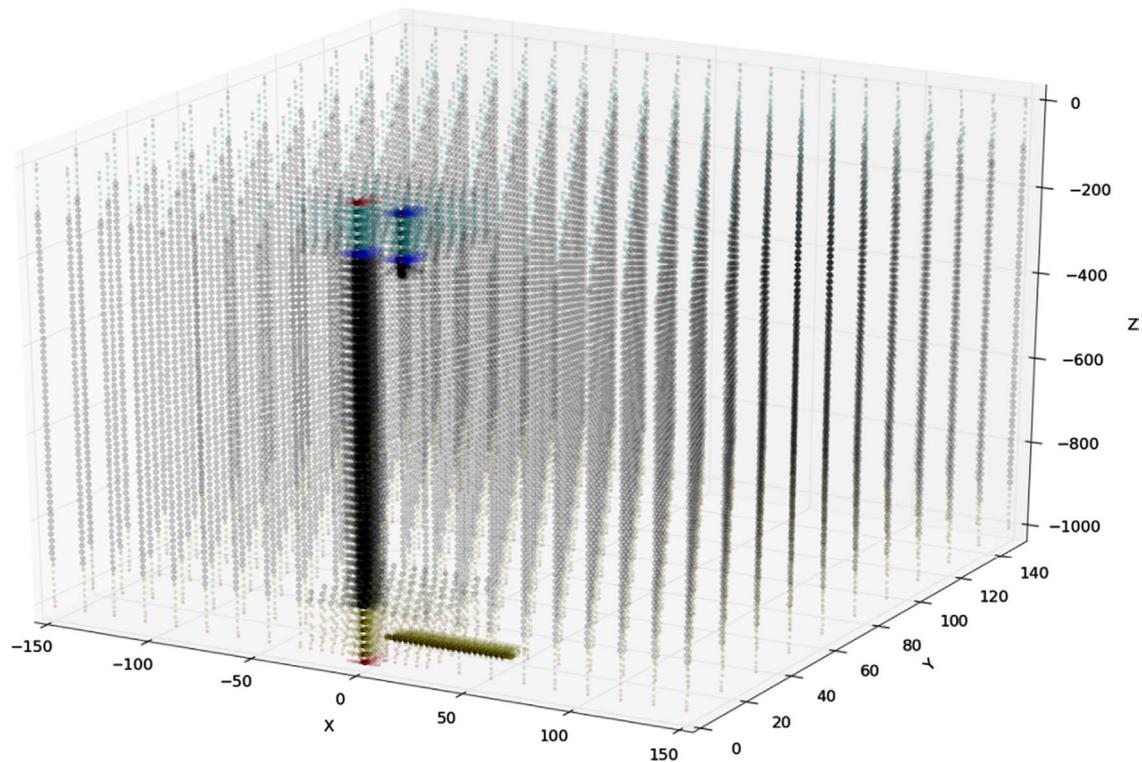
Fig. 5. A domain including two vertical wells and a horizontal well (with local refinement in the vicinity around each well), in the midst of a larger more coarsely refined region.
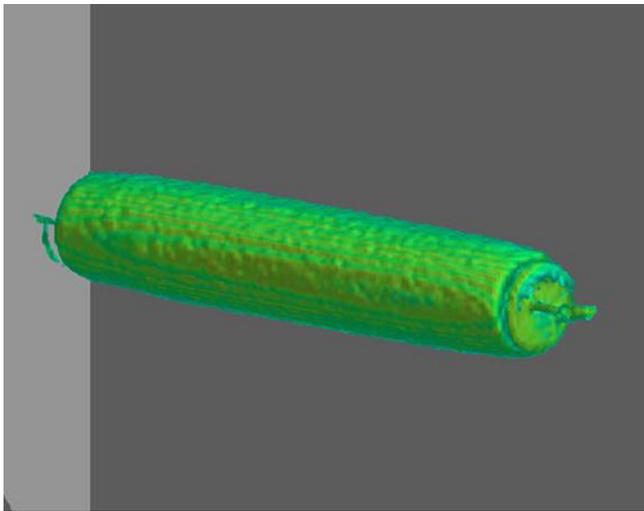
equation) is valid. We simulated the entire process of dissociation of methane hydrate in a hydrate-impregnated core sample that had been scanned using Magnetic Resonance Imaging (MRI) in the study of Birkedal et al. (2011). Figs. 6 and 7 show the saturation of liquid water in the core sample. The sample shown in Fig. 8a was subjected to depressurization, causing dissociation of the methane

hydrate, and the resulting change in the phase saturations inside the core was monitored and measured (Fig. 8b and c).

In order to (a) explore whether the hydrate dissociation in this experiment was an equilibrium or a kinetic process, and (b) to determine the corresponding parameters, we simulated the behavior of the entire system accounting for all the important conditions, properties and characteristics that defined it. To accomplish this, we needed a refined simulation mesh capturing all of the data extracted from the MRI scan and the surrounding experimental apparatus.
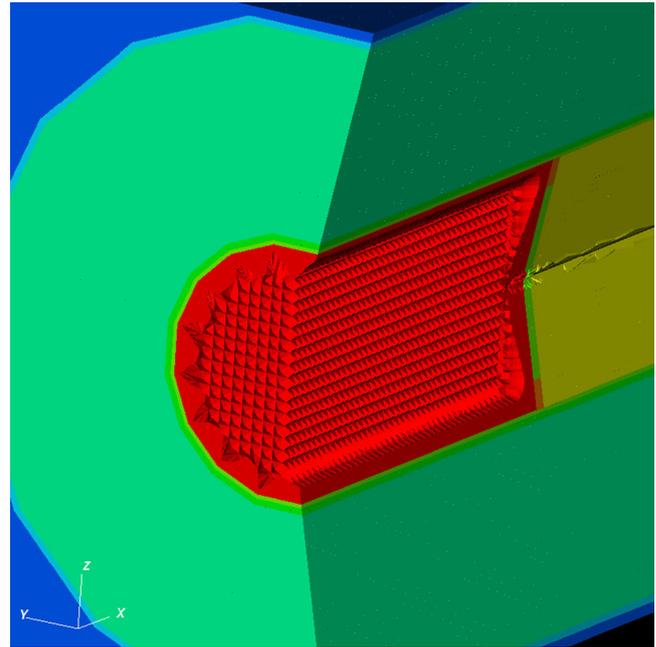
In describing the system geometry, the dissociation apparatus needed to be modeled accurately in order to capture the heat transfers through the entire domain (including the boundaries). As depicted in Fig. 9, the hydrate-bearing core was surrounded by

an insulating Teflon sleeve and a very thin (micron-scale) gas pocket (gap) along its cylindrical outer surface, and was fastened in place on each end by plastic spacers. Accurately capturing the geometry of these elements, and accurately describing their thermal properties, was vital to represent the heat flow into the core from the outer boundary (Fluorinert – a heat transfer fluid – circulating at a monitored, nearly constant temperature).

Earlier attempts to capture the geometry of the dissociation system using a standard rectilinear grid has indicated early-on the


Fig. 6. A core sample saturated with gas hydrate was scanned using a Magnetic Resonance Imaging (MRI). The MRI process generates useful data, including the saturation of liquid water in the core, depicted here using high-resolution voxels. This MRI-derived saturation data is then processed to generate the initial conditions of the TOUGH simulation (Birkedal et al. 2011).

Fig. 8. A cross-section of the simulation mesh for the hydrate core dissociation experiment, color coded by material type. The inside of the hydrate core (red) uses rectilinear gridding while the remainder of the grid uses pseudo-radial gridding. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
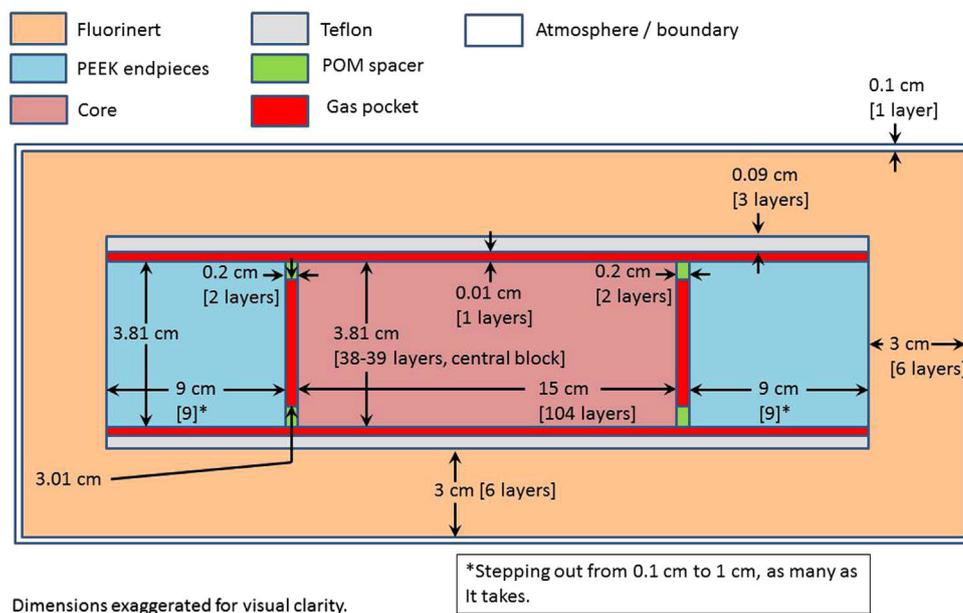
Fig. 7. Schematic representation of the laboratory setup for the thermal dissociation of a methane hydrate core. The dimensions of the illustration are skewed to clearly show the very thin but important air pocket layer. The Fluorinert layer is an inert pad of gas meant to thermally insulate the apparatus. PEEK refers to polyether ether ketone and POM refers to polyoxymethylene; these are insulators.

futility of this approach, as it would require a prohibitively large simulation grid to capture the relevant details of the system at the needed detail. Application of the MeshVoro tool to this geometry enabled very fine gridding where needed. More importantly, MeshVoro enabled a smooth transition between rectilinear gridding inside the hydrate core and pseudo-radial gridding outside the core. This transition is depicted in the cross-sectional in Fig. 8, where the MRI scan data exactly determined the locations and properties of the grid cells inside the core (Fig. 9); a custom-made dynamic radial gridding algorithm populated the mesh describing the rest of the laboratory apparatus (Figs. 10 and 11).

These examples show the power of the MeshVoro application, and the strict control it allows for the description of the details of a system over several spatial scales and in complex geometries.

4. Discussion

3D Voronoi cells can easily possess 12 or more connected neighbors with no structural regularity, whereas a traditional 3D Cartesian or radial mesh can only have a maximum of six neighbors with consistent orientation. Consequently, the Jacobian matrix developed by the TOUGH codes to compute the solution of the flow equations originating from Voronoi meshes will be significantly less sparse, more ill-conditioned, and with a less

regular corresponding incidence matrix than those from Cartesian meshes. This implies poorer solver performance for Voronoi meshes compared to that corresponding to Cartesian meshes with a similar number of grid cells. Note that the reference to "poorer" solver performance does not mean deterioration in the accuracy of the solution, which is unaffected, as it is controlled exclusively by the convergence criterion in the Newton–Raphson iterative process (Pruess et al., 1999); it means an increased number of iterations to convergence and, consequently, smaller time-step sizes, a larger number of time steps to cover the desired simulation period and a longer execution time. The solution performance of problems involving Voronoi grids may be further aggravated by improper spatial discretization, which would result in larger space discretization errors and correspondingly increased convergence difficulty. We must emphasize that the purpose of Voronoi meshes is to represent geometries that cannot be efficiently represented by Cartesian meshes, where the expected tradeoff in numerical performance is warranted, and the additional execution time and larger memory requirements are not an issue as there is practically no alternative. For simple, geometrically regular problems, structured meshes are sufficient and are almost certain to outperform Voronoi meshes in terms of speed and memory requirements. Given the intensive labor requirements to construct them, Voronoi meshes should be used where they are suited to the demands of the problem, such as in the irregular three-dimensional meshes described above.

Another crucial property of the meshes generated by MeshVoro is that the Voronoi tessellations are not required to be centroidal, meaning that the element generator points are not necessarily coincident with the element centroids. The option to perform Lloyd's (1982) iteration on the meshes is available, but we note that the smoothing of very refined mesh regions such as wells and fractures can severely distort these objects. Obtaining mesh "cendroidality" eats the expense of the geometric integrity of crucial flow and geometry features defeats the purpose of an unstructured locally-refined mesh. The Lloyd's iteration sacrifices some numerical accuracy for geometric resolution and flexibility, and in so doing opens up a new range of problems.

Regarding visualization of simulation results in a three-dimensional fashion, we find no truly adequate tools exist for creating publication-quality representations of complex three-dimensional data. The tools we discuss and develop here may be used for qualitative illustration and for data exploration, but these tools do not provide precise quantitative illustrations. Events of scientific interest may occur on a scale of centimeters or kilometers, depending on the nature of the problem, and it is difficult to imbue static graphical representations with sufficient context to unambiguously communicate the intent of any given figure.
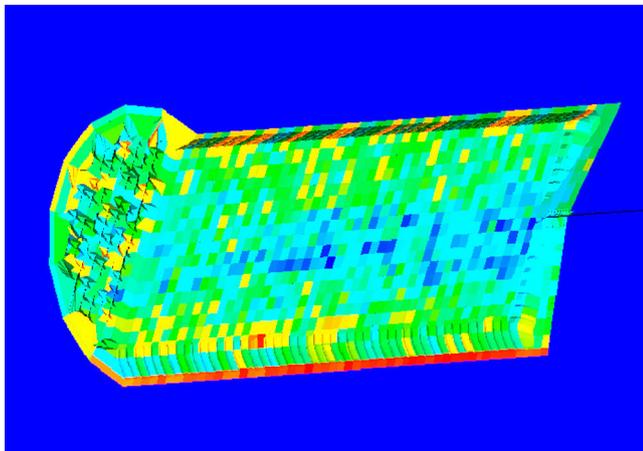
Fig. 9. A cross-section of the simulation mesh for the dissociation experiment of the hydrate-bearing core. The different colors depict differences in saturation of liquid water determined from the MRI scans. The grid center points of the mesh and the properties assigned to these points are inherited directly from the MRI scan data.
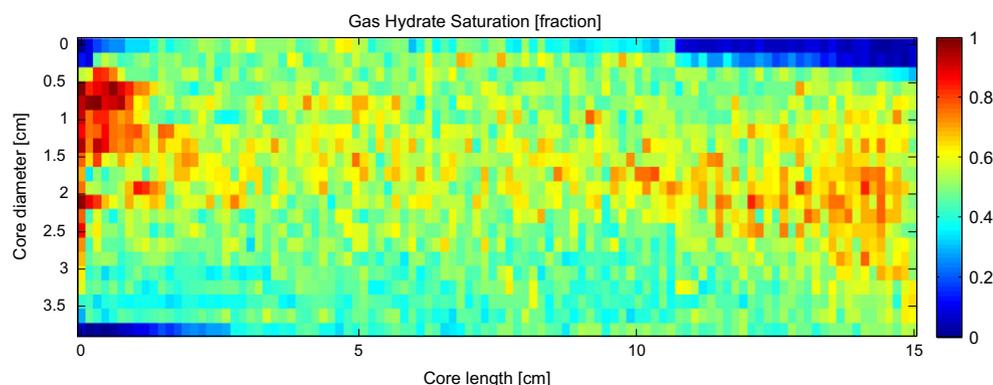
Fig. 10. An axial cross-section view of the simulation state of the hydrate core sample showing liquid water saturation at its initial state, as determined from interpolation from the MRI scan data.
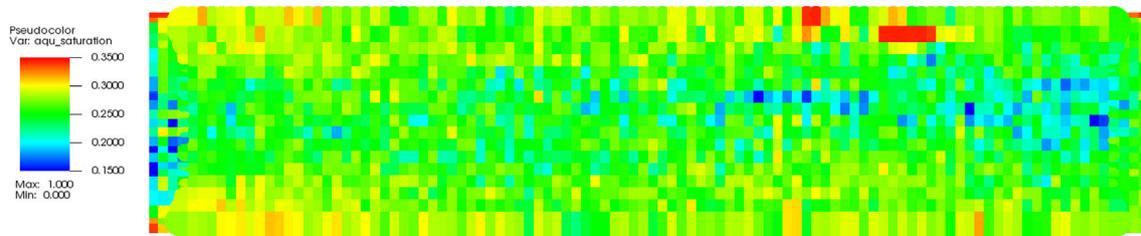
Fig. 11. An axial cross-section view of the simulation state of the hydrate core sample showing liquid water saturation at the end of the simulated depressurization.

Clearly representing complex three-dimensional geometries with printed two-dimensional figures remains a challenge.

References

Barree R.D., Conway M.W.: Beyond beta factors: a complete model for Darcy, Forchheimer and trans-Forchheimer flow in porous media. In: Paper SPE 89325 Presented at the 2004 Annual Technical Conference and Exhibition, 26–29 Sept 2004, Houston, TX.

Barree R.D., Conway M.W., 2007. Multiphase non-Darcy flow in proppant packs. In: Paper SPE 109561, Presented at the 2007 Annual Technical Conference and Exhibition, 11–14 Nov 2007, Anaheim, CA.

Birkedal, K.A., Ersland, G., Hauge, L.P.O., Graue, A., Hester, K., Stevens, J., Howard, J., 2011. Electrical resistivity measurements of CH_4 hydrate-bearing sandstone during formation. In: Proceedings of the Seventh International Conference on Gas Hydrates held in Edinburgh, Scotland, July 17–21.

Chazelle, N., 1980. Computational Geometry and Convexity. Carnegie-Mellon University (Ph.D. Thesis. CMU-CS-80-150, Computer Science).

Rycroft, Chris H., 2009. Voro++: a three-dimensional Voronoi cell library in C++. Chaos 19, 041111.

Du, Q., Faber, V., Gunzburger, M., 1999. Centroidal Voronoi tessellations: applications and algorithms. SIAM Rev. 41 (4), 637–676.

Du, Q., Gunzburger, M., 2002. Grid generation and optimization based on centroidal Voronoi tessellations. Appl. Math. Comput. 133, 591–607.

Edwards, A.L., 1972. TRUMP: A Computer Program for Transient and Steady State Temperature Distributions in Multidimensional Systems, National Technical Information Service. National Bureau of Standards, Springfield, VA.

Florack, L., Duits, R., Jongbloed, G., 2012. Mathematical Methods for Signal and Image Analysis and Representation. Springer, London (105).

Forchheimer, P., 1901. Wasserbewegung durch Bode. ZVDI 1901, 45.

Lloyd, Stuart P., 1982. Least squares quantization in PCM. IEEE Trans. Inf. Theory **28** (2), 129–137, http://dx.doi.org/10.1109/TIT.1982.1056489.

Merland, R., Levy, B., Caumon, G., 2011. Building PEBI grids conforming to 3D geological features using centroidal Voronoi tessellations. In: IAMG Proceedings.

Moridis, G.J., Kowalsky, M., Pruess, K., 2008. TOUGH+HYDRATE v1.0 User's Manual. LBNL-161E. Lawrence Berkeley National Laboratory, Berkeley, CA.

Narasimhan, T.N., Witherspoon., P.A., 1976. An integrated finite difference method for analyzing fluid flow in porous media. Water Resour. Res. 12 (1), 57–a64.

Narasimhan, T.N., Witherspoon, P.A., Edwards, A.L., 1978. Numerical model for saturated-unsaturated flow in deformable porous media, Part 2: The algorithm. Water Resour. Res. 14 (2), 255–261.

Palagi, C., 1992. Generation and Application of Voronoi Grids to Model Flow in Heterogeneous Reservoirs. Stanford Univerity (Ph.D. Thesis).

Persson, P., 2005. Mesh Generation for Implicit Geometries. MIT (Ph.D. Thesis).

Persson, P., Strang, G., 2004. A simple mesh generator in MATLAB. SIAM Rev. 46 (2), 329–345.

Pruess, K., Oldenburg, C., Moridis, G., 1999. TOUGH2 User's Guide, Version 2.0. Lawrence Berkeley National Laboratory, Berkeley, CA (Report LBNL-43134).

Rycroft, C.H., 2009. Voro++: A three-dimensional Voronoi cell library in C++. Chaos 19, 041111.

Schroeder, W., Martin, K., Lorensen, B., 1998. The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics. Prentice Hall, New Jersey.

Verma, K., 1996. Flexible Grids for Reservoir Simulation. Stanford University (Ph.D. Thesis).