Computational Geosciences (2018) 22:1561–1581
https://doi.org/10.1007/s10596-018-9774-0

ORIGINAL PAPER

Generalized prism grid: a pillar-based unstructured grid for simulation of reservoirs with complicated geological geometries

Xin Li[1] · **Xiang Li**[1] · **Dongxiao Zhang**[1,2,3]

Received: 19 July 2018 / Accepted: 3 September 2018 / Published online: 19 September 2018

Abstract

Grid generation is critical to numerical reservoir simulations. High-quality grids guarantee the fidelity of a reservoir model and keep the flow calculations simple. In this study, we propose a 3D unstructured grid, the generalized prism grid (GPG), to model reservoirs with complicated geological geometries, including horizons, pinch-outs, faults, fractures, and bore holes. GPG is a layered, pillar-based grid. The location of a face node is specified by its elevation, and the pillar to which it is attached. Compared with the hexahedral corner point grid (CPG), GPG is a polygon prism and therefore more flexible; whereas, compared with the 2.5D perpendicular bisection (PEBI) grid, GPG allows polygons morphing through the stratum. We built a gridding algorithm to fulfil the features of GPG. The algorithm first constructs a 2D triangular mesh for one layer by setting up control points and grid densities for geological objects, such as fractures, faults, and wells, distributing triangular grid points with the "advancing front method," and performing Delaunay optimization to the points. The polygon mesh is the dual grid of the triangular mesh. Taking the polygon mesh as a reference, the mesh for each layer of the strata is a morphing of it, with edges being stretched and points being assigned with heights. We also designed a compact file format to store GPG data and implemented the flux calculation method for GPG in a reservoir simulator. The attractive features of GPG are demonstrated through four examples. The conciseness and flexibility of GPG make it a potential new standard grid format replacing CPG.

Keywords Unstructured grids · Reservoir simulation · Fractured reservoir · Homeomorphic mapping

1 Introduction

A numerical reservoir simulation is performed to model a reservoir's dynamic status on a discrete grid. Due to the form of Darcy's law and the complexities of modeling multiphase flow, the control volume finite difference (CVFD) method remains the most commonly used discretization method for reservoir simulation. This approach requires the orthogonality of the grid; otherwise, the flux calculation would lose accuracy. Structured grids are naturally orthogonal, but the discontinuous nature of geological bodies makes it almost impossible to align structured grids with arbitrarily distributed geological objects, e.g., horizons, pinch-outs, faults, fractures, bore holes, etc. Although the improved finite difference (FD) schemes, such as MPFA [1–3], mimetic finite difference (MFD) [4–7], and nonlinear two-point flux approximation(NTPFA) [8, 9], can diminish the effect of non-orthogonality, these schemes complicate the flux stencil, introducing additional difficulties to the solution of the discretized equations. To preserve the geometrical details and keep the flux stencil simple, building a high-quality grid is always the optimal choice.

A grid that can faithfully model irregular and discontinuous reservoirs should have the following abilities:

– Conformation to the stratum: a grid block should not be partially in one layer and partially in another, because

✉ Xiang Li
lixiangcn@pku.edu.cn

1 ERE, College of Engineering, Peking University, Room 1009, Wangkezhen Building, Beijing, China

2 SKLTCS, College of Engineering, Peking University, Yannanyuan #60, Beijing, China

3 BIC-ESAT, College of Engineering, Peking University, Yannanyuan #60, Beijing, China

most reservoirs are a layered sedimentary bed. Hence, the grid arrangement should allow cell degeneracy and disappearance to represent pinch-outs.

– Representation of faults: faults are discontinuous faces in both geometry and mechanics. In a reservoir simulation, the usual way to represent the barrier effect of faults is to scale the transmissivities between the grid blocks on the opposite sides of the fault. The grid should allow misaligned and hanging cells. The corresponding flux stencil should include non-neighbor connections.
– Representation of fractures: Fractured reservoirs make up a large and increasing percentage of the world's oil and gas resources [10]. Fractures act as high-speed flow paths for reservoir fluids. Modeling fractures in a reservoir simulation is always a complex problem. On the one hand, the pseudo-continuum method averages the fractures as an equivalent permeability. Although greatly simplifying the grid, it loses geometric details and fails to capture the transient behaviors of flows. On the other hand, explicitly modeling fractures is challenging both to the gridding algorithm and to the numerical solution. When the fracture network is complicated, generating meshes with smoothly changing cell densities around the fractures is difficult; whereas, the smooth transition between small size cells and large size cells constitutes a guarantee for the convergence of the solution. Moreover, strong heterogeneity and large grid number would reduce solution speed.
– Representation of boreholes: In reality, a well bore is unlikely to be exactly parallel or perpendicular to the layer. Grid blocks built only in respect to well bores would usually conflict with the stratum, and vice versa. Furthermore, the pressure gradient near the well is radial and close to logarithmic. Therefore, there should be enough transitional grid blocks between the well bore and stratum, and the grid around the well bore should be radial-shaped and logarithmically refined.

The corner-point grid (CPG) [11–15] is a structured and pillar-based grid widely utilized in industrial reservoir simulators. CPG is made of hexahedral cells numbered by (I, J, and K) indices. Each CPG cell is defined by four pillars and the heights of the eight corners, each two of which are attached to one of the pillars. CPG uses grid layers to model sedimentary bedding layers. The cells can degenerate (have less than eight corners) to adapt to eroded beds. The CPG format allows discontinuities across cell faces and degeneracies of cell thicknesses to model faults and pinch-outs, respectively. However, CPG only offers limited flexibility in matching with geologists' perceptions of reservoirs and is often inexact to model fractures and wells due to the constraint of grid regularity [16].

Another choice is using an unstructured grid. Essentially, there are two categories of unstructured grid, i.e., the Voronoi grid [17–22] and the triangular-shaped grid [23, 24]. The Voronoi grid, whose edges are orthogonal to grid-centroid connecting lines, is commonly used in reservoir simulation and known as the perpendicular bisection (PEBI) grid. In fact, a PEBI grid is defined as a convex polygon within which any point is closer to its own center than to any other block center. The two main types of PEBI grids are the full-3D PEBI grid and the 2.5D PEBI grid. Full-3D PEBI grids allow faces to have arbitrary numbers of edges. The trade-off for this flexibility is losing the layered grid structure [16, 25]. 2.5D PEBI grids are prism grids with the mesh of each layer being the same. The contacted faces of any pair of grid blocks are aligned. The trade-off here is losing the ability to model declining geological objects. Triangular-shaped grids also comprise two main types: the tetrahedron grid and the prismatic grid [26, 27]. The tetrahedron grid offers the best flexibility to model subsurface structures. The prismatic grid, which is also known as the triangular prism grid, is not as flexible as the tetrahedron grid but is more suitable for layered structures. The applications of triangular-shaped grids are extensive in computational fluid dynamics and structural mechanics but limited in reservoir simulations. The reasons for this are (1) the triangular-shaped grids are hardly orthogonal if being conformed to subsurface structures, making the CVFD scheme inaccurate; and (2) the well inflow calculation is more difficult to implement compared with that in CPG or PEBI grids [23, 28]. From a practical perspective, a layered grid is always preferred to a non-layered grid in reservoir simulations.

By reconsidering the advantages and disadvantages of CPG and PEBI grids, we propose a new type of 3D grid, the generalized prism grid (GPG), to meet the requirements of both flexibility and practicability. GPG is a pillar-based grid like CPG, but the basic grid block is extended to a polygon prism. Different from the full-3D PEBI grid, GPG is a layered grid. GPG also differs from the 2.5D PEBI grid, in that the GPG allows a high degree of grid misalignment. In this work, we built a gridding algorithm for GPG. The algorithm can be summarized as the following four steps:

1) Setting up control points for determined geological objects. The control points draw the outlines of the horizontal projections of the geological objects. They will not move in 2D reference mesh generation (step 2).
2) Generating the 2D reference mesh. The reference mesh is a 2D Voronoi mesh used to constrain the topology of each layer of the final GPG. In this work, we generate the dual mesh (Delaunay triangular) first, using the "advancing front method." This method requires knowing the grids' density field in advance, which is

governed by a Poisson equation taking wells and fractures as sources [29].

3) Constructing the GPG layer by layer. The mapping from the reference mesh to a GPG layer is homeomorphic, i.e., the topology remains constant. However, the points of the GPG layer could have displacements to the reference mesh, because the geological structures may incline. To smooth the point distribution and prevent polygon-overlapping after mesh morphing, for each layer, the (x, y) coordinates of points are optimized using the "equilibrium spring" method [30], with the fixed displacement boundary condition.
4) Assigning depths to the grid points of each layer. With a known contour map, the Discrete Smoothing Interpolation [31, 32] is a good choice.

The remainder of this paper is arranged as follows. Section 2 describes the basic definition of GPG. Section 3 presents the gridding algorithm, including fracture and fault representation, 2D reference mesh generation, and layer morphing. Section 4 introduces a compact data format to store GPG. Section 5 proposes the method of calculating the flux terms for reservoir simulations based on GPG. Section 6 presents four GPG examples to demonstrate the modeling of the well trajectory, the declining fractures, the 3D faults, and the complex fracture networks, respectively. The work is concluded in Section 7 with a discussion of the advantages of GPG.

2 Definition of GPG

The basic idea of GPG is to approximate layered stratum using columns of prisms. In each column, the prisms stack neatly and share the same set of pillars (Fig. 1a). Let $S = \left\{S_j^i\right\}_{i=1\ldots n,\, j=1\ldots m}$ be a set of points in 3D space, where n is the number of the pillars and m is the layer number. A GPG cell is a polyhedron $\Omega = \left\{s_a^1, \ldots, s_a^n; s_{a+1}^1, \ldots, s_{a+1}^n,\right\}$, where the superscripts (1 to n) are the pillar indexes and the subscripts (a and $a+1$) are the layer indexes. The 2.5D PEBI grid is a special case of GPG with all prisms being vertical. The CPG and prismatic grid are special cases of GPG where all elements are hexahedrons and tetrahedrons, respectively. The index of GPG is a tuple (I, K), where I is the column number and K is the layer number. This is different from the triple index of CPG. Figure 1 is the schematic diagram of GPG. In GPG, the horizons can split (Fig. 1b), the layers can degenerate (Fig. 1d), and the pillars can bend and overlap (Fig. 1e).

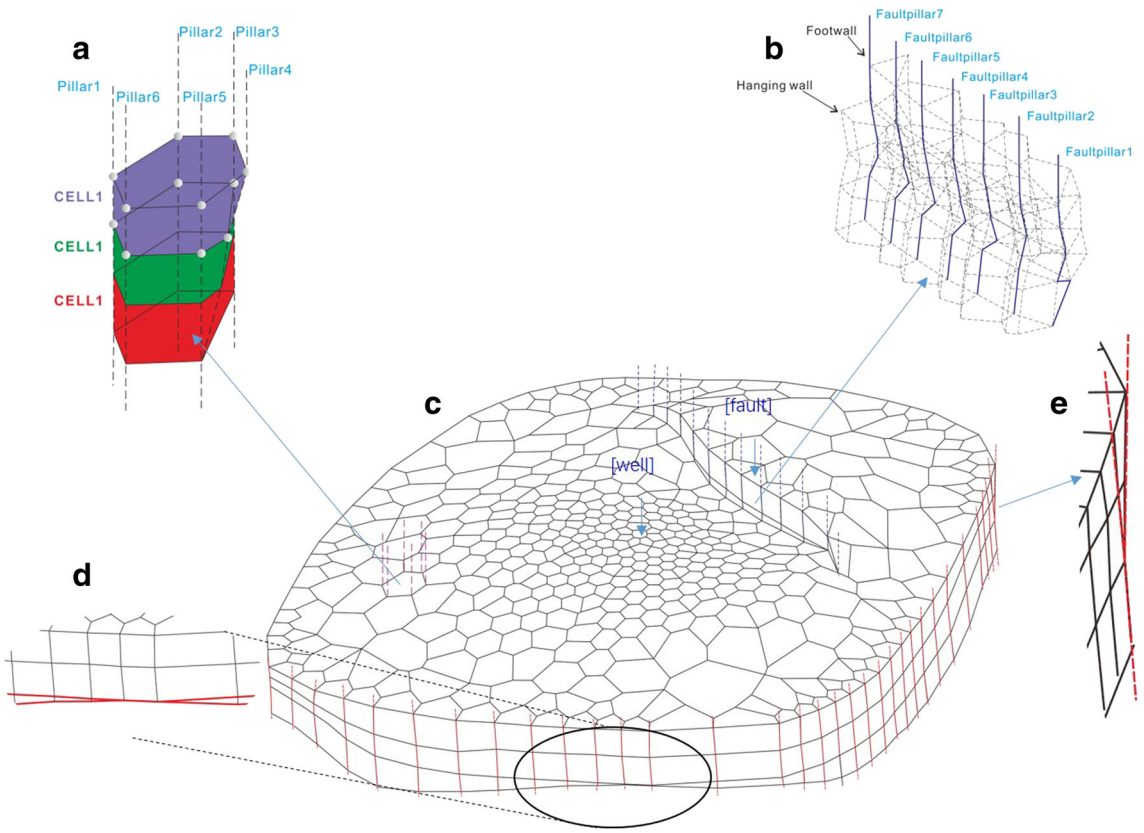

Fig. 1 Schematic diagram of GPG

3 Generation of GPG

GPG can be considered a combination of single-layered meshes. To avoid the breaks of pillars, the topologies of different layers should be identical. However, the pillars are not necessarily straight lines, which means that the absolute positions of mesh points are not fixed, so the single-layered meshes can morph. This flexibility makes GPG different from the 2.5D PEBI grid. To achieve a representation of 3D geological objects with GPG involves mapping the 2D cross-sections of geological objects to single-layered meshes like the tomography. We divide the generation of GPG into five steps (the final step is optional):

1) Setting up 2D control points for the determined geological objects;
2) Generating a 2D mesh as the topology reference;
3) Morphing the 2D mesh to form the mesh of each layer;
4) Assigning depths to the grid points of each layer;
5) (Optional) Smoothing the grid through minor adjustments to the locations of unfixed points [33, 34].

3.1 Representation of fractures

Geometric parameters that describe a fracture include direction, inclination, dip angle, and fracture width. To reflect all of the above parameters of fractures, we designed the fracture grid as shown in Fig. 2. The basic shape of a fracture grid is a rectangle, and the intersection grid is a polygon prism. As the pillars that define prism grids can incline or bend, the fracture can incline or bend, but the fracture grids from different layers still possess the same topological structure. When creating the 2D reference mesh, control points of fractures (black dots in Fig. 3a) are inserted into the set of boundary points, which constrains Delaunay triangulation. The control points of fractures form a series of right triangles, and the fracture edges are just the connections of hypotenuse midpoints. When two fractures intersect, the triangular edges that emit from the intersection point should be equal, e.g., $L_1 = L_2$ in Fig. 3a, to make the resultant dual grid close to axisymmetric. When the intersection angle of two fractures (θ in Fig. 3b) is small, to keep the size of the intersection grid small, two additional control points are placed at the bisector of the obtuse angle. The obtuse angle is formed by the central lines of the two fractures. This point inserting method follows the work of Yang et al. [35].

3.2 Representation of faults

The displacement, inclination, shape, and width of faults bring great challenges to gridding. In GPG, faults are misalignments of grids. The footwall and hanging wall of a fault are defined by two sets of pillars. These pillars are also parts of reservoir grids. In GPG, by bending the pillars, a fault could have volume (Fig. 1b) between the footwall and the hanging wall, thus being more realistic; whereas, in CPG, faults are only volumeless faces. Like the representation of fractures, control points of faults are inserted as fixed points into the 2D reference mesh. In this work, we make the fault a piece of polyline in 2D, as shown in Fig. 4b, and points of the polyline are the control points in Delaunay triangulation. We distribute other points around the polyline using a constrained Voronoi method [36]. When two faults intersect, the control points are placed on the two circles (the inner circle and the outer circle in Fig. 4a) that are centered at the intersection. Both circles are protected areas, which means that no point is allowed to add into the areas. The pair of points on the opposite sides of a fault edge forms a mirror image with respect to the edge. For instance, on the inner circle in Fig. 4a, the pairs of points are P4 and P23, P17 and P22, P11 and P16, and P5 and P10. Until this step, the footwall and hanging wall of the fault are still sealed. After mesh morphing, however, the footwall

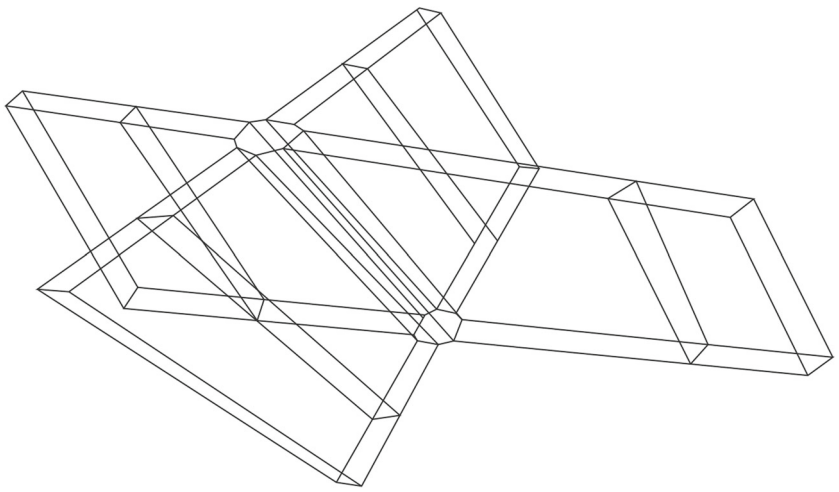

Fig. 2 Illustration of fracture grids

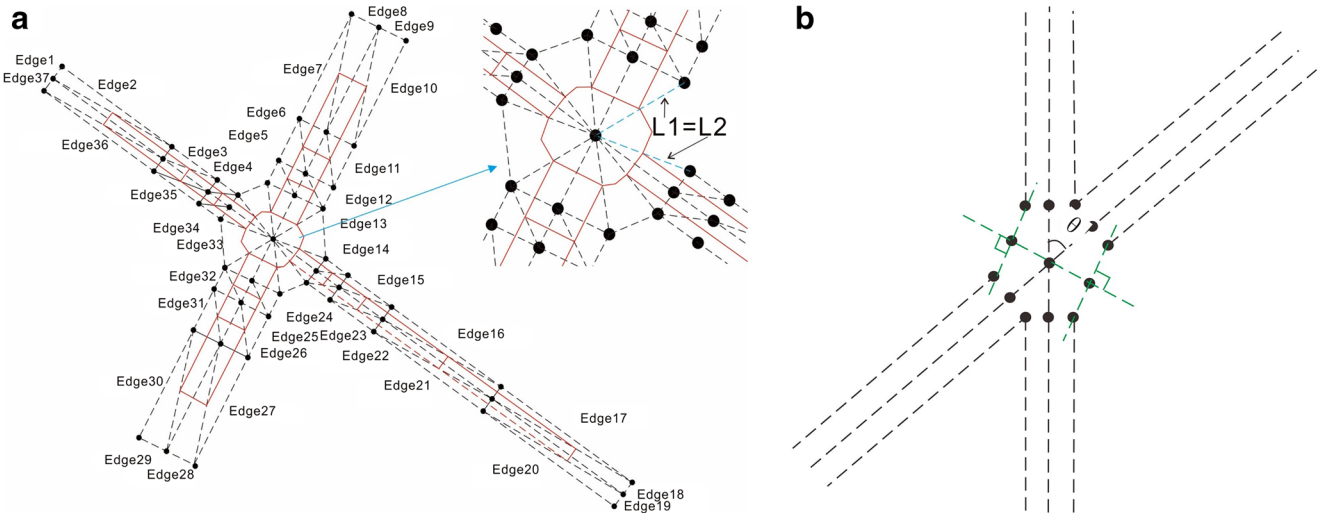

Fig. 3 Fixed-point distribution of fractures

and the hanging wall will split and the fault will become a body (Fig. 4b).

The key to representing x and y faults is the allowance of pillars overlapping and grid degeneration. In GPG, the y fault is two pillars overlapping at the lower half under the inflexion point (Fig. 5a). The x fault is two pillars touching at the inflexion point (Fig. 5b). We setup a synthetic case to show how the GPG generates x and y faults. The synthetic case is a square domain with one x fault and one y fault (Fig. 6).

3.3 Generation of the reference mesh

The reference mesh is a polygon mesh that defines the 2D topologic structure, which includes representations of geological objects and information about grid adaption. Grid adaption is essential for balancing accuracy and efficiency in reservoir simulations. Different from in structured grids, grid adaption in GPG is more natural, by inserting points according to a "density field" prior to the Delaunay triangulation. The construction of the reference mesh includes three steps: (1) calculating the density field; (2) inserting points using the "advancing front method"; and (3) performing Delaunay triangulation to the inserted points. The polygon mesh is the dual-grid of the triangular mesh.

3.3.1 Density field calculation

For the consistency of accuracy, the grid density should be proportional to the pressure gradient, and thus the pressure difference between any two grid blocks is identical. As the reservoir equation is a Poisson-type equation with wells and fractures as sources, the grid density field should also be governed by a Poisson equation with the same sources. The intensities of the sources in the Poisson equation are specified according to some prior knowledge, such as the ratio of fracture permeability to matrix permeability.

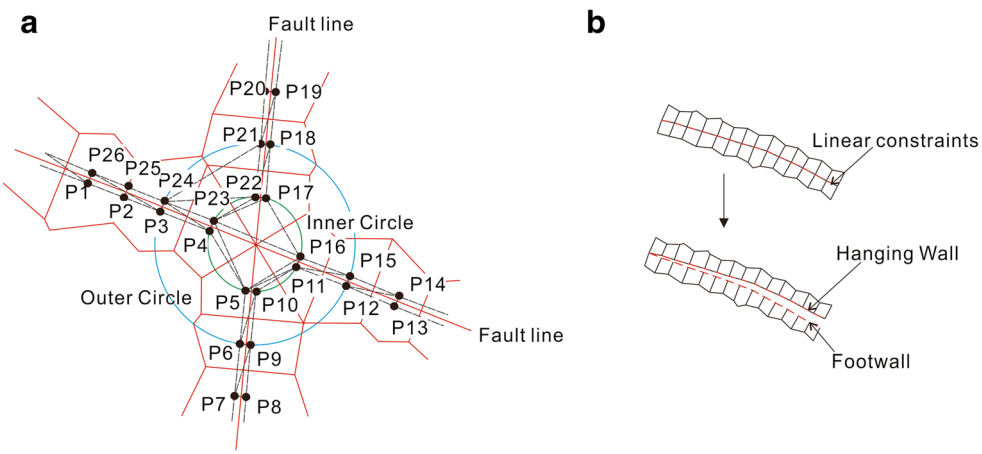

Fig. 4 Representation of faults. **a** The control points of two intersected faults; **b** the split of hanging wall and footwall after mesh morphing

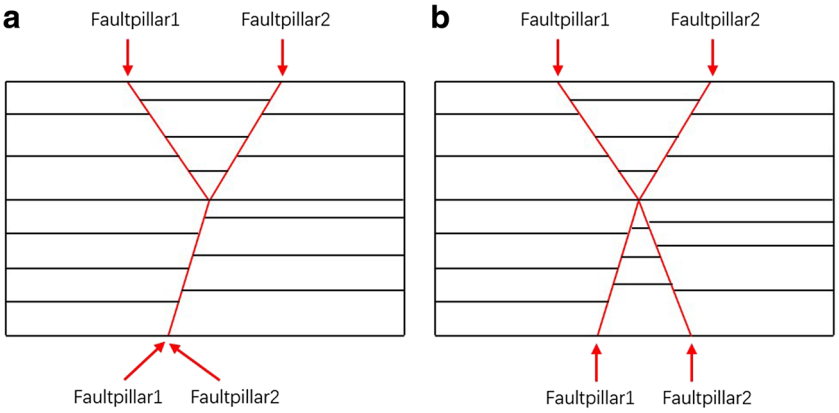

Fig. 5 Representation of x and y faults, with the red lines being the pillars: **a** the y fault; **b** the x fault

The governing equation of density field follows the work of Pirzadeh [37] and is written as:

$$\nabla^2 S = G\,(\mathbf{x}) \tag{1}$$

where G is the sum of source terms; and S is the reciprocal of grid spacing. There are two kinds of source terms, i.e., the point source terms (for vertical wells) and the linear source terms (for horizontal wells and fractures). G is a Green function written as:

$$G\,(\mathbf{x}) = \sum_{n=1}^{N} \psi_n (S \times J_n - I_n) \tag{2}$$

where ψ_n is the priori known intensity factor of the nth source term; and J_n and I_n are the spacing parameters of the nth source term. For point sources, J_n and I_n are, respectively:

$$I_n = \frac{S_n}{r_n^2} \tag{3}$$

$$J_n = \frac{1}{r_n^2} \tag{4}$$

where r_n is the distance between the source and point $\mathbf{x}$; and S_n is the prescribed value of the spacing parameter of the nth source. $1/S_n$ equals the desired diameter of the source grid.

For line source, I_n and J_n are defined as line-averaged spacing parameters, respectively:

$$I_n = \frac{1}{l_n} \int_0^{l_n} \frac{f(l)}{r(l)^2} \mathrm{d}l \tag{5}$$

$$J_n = \frac{1}{l_n} \int_0^{l_n} \frac{\mathrm{d}l}{r(l)^2} \tag{6}$$

where $r\,(l)$ is the distance from a point of the line to point $\mathbf{x}$; and $f\,(l)$ is the spacing parameter varying along the line. $l_n/f\,(l)$ equals the desired length of the discrete source segment. For segments with equal length, $f\,(l)$ is constant.

In this work, Eq. 2 is discretized and solved on a $N_x \times N_y$ square mesh. The discrete equation is written as:

$$\mathbf{L} \times \mathbf{s} = \mathbf{b} \tag{7}$$

where $\mathbf{s}$ and $\mathbf{b}$ are column vectors of length $N = N_x \times N_y$. The nth value of $\mathbf{b}$ is $-\dfrac{D^2 \sum_{n=1}^{N} \psi_n I_n}{4 + D^2 \sum_{n=1}^{N} \psi_n J_n}$, where D is the length of the mesh square. The coefficient matrix $\mathbf{L}$ is a

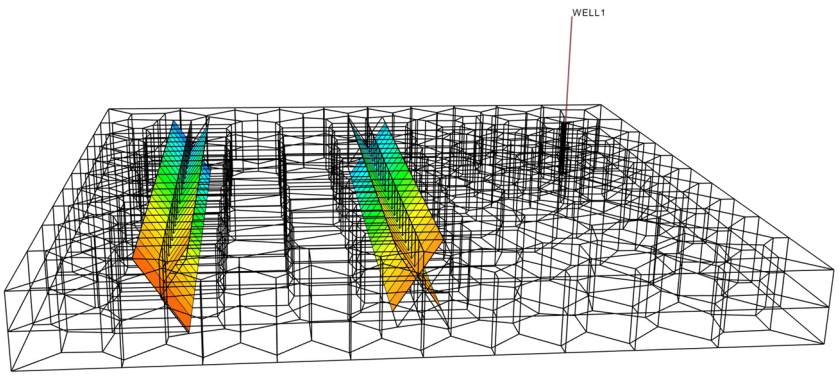

Fig. 6 An example of x and y faults

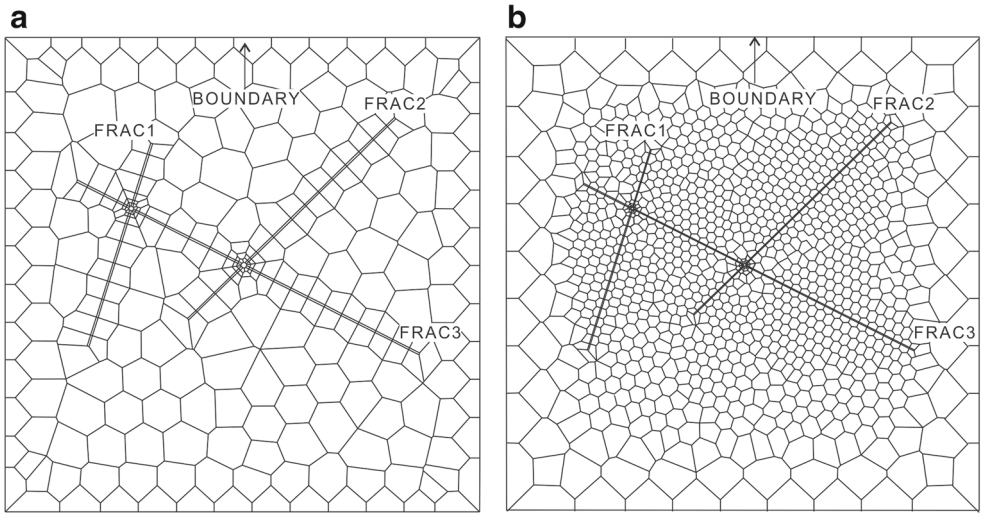

Fig. 7 Effect of source intensity on the polygon mesh

$N \times N$ diagonally dominant pentadiagonal matrix, in which the (i, j) element is:

$$\begin{cases} 1 & \text{if} \quad j = i \\ \dfrac{1}{4+h^2 \sum\limits_{n=1}^{N} \psi_n J_n} & \text{if} \quad j = i+1, i-1, i+N_x, \text{or } i-N_x \\ 0 & \text{others} \end{cases} \tag{8}$$

Linear (8) is solved by Pardiso [38, 39], which is a direct solver based on sparse Gaussian elimination. Parameter ψ_n controls the impact area of the nth source. The effect of ψ_n is demonstrated in Fig. 7. The figure shows a rectangular domain with three fractures. Essentially, the larger the ψ_n, the larger the impacted area.

3.3.2 Advancing front method

With the control points being fixed, the blank area is filled with points through the advancing front method. The "front" refers to a series of connected points holding an area; the "advancing" of the front is to insert some points outside of the area, connect the new points with some old points, and expand the area. The ideal situation is that the newly inserted point forms an equilateral triangle with an edge of the front. The general way to insert a point is to place a trial point on the perpendicular bisector (the red line in Fig. 8) of the edge and adjust the distance between the trial point and the edge. The initial distance is interpolated from the density field. Previous advancing front methods [26, 40] need to keep the integrity of the front to distinguish interior and outer areas. A trial point is accepted if it falls into the outer area, and also has no front point lying in the searching region. The searching region is a circle centered at the trial point. Its radius is comparable with the initial distance between the trial point and the edge. If the trial point falls into the interior area, it is abandoned and replaced with the closest front point, and thus the new triangle is still forming and the front is still advancing. An extreme case occurs when the closest front point lies out of the searching region and expanding the searching radius would make the triangle obtuse. This case could occur frequently in applications of GPG. As shown in Fig. 9, when two fractures intersect in a small angle, the edge near the intersection may fail to find a valid trial point, because the trial point may cross the front of another fracture and the searching radius is too small to find an alternate point on the front. In Fig. 9, T is the trial point of front edge CH, ST crosses another front edge CG, and there is no alternate front point inside the searching region. The

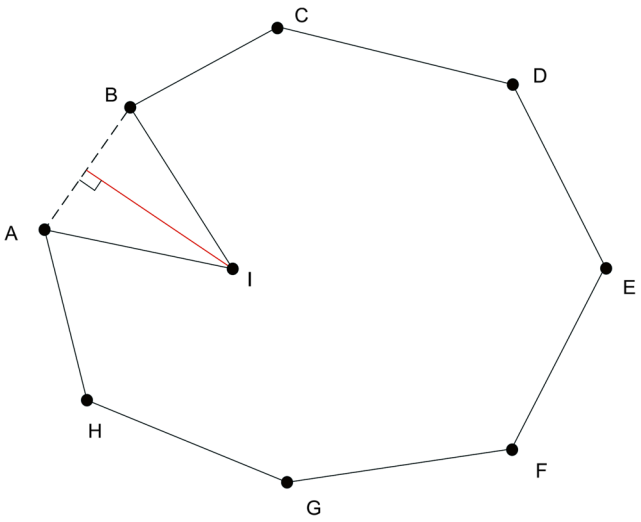

Fig. 8 Illustration of the advancing front method. The initial front is A-B-C-D-E-F-G-H-A. After inserting point I, front edge AB is updated with edges AI and IB

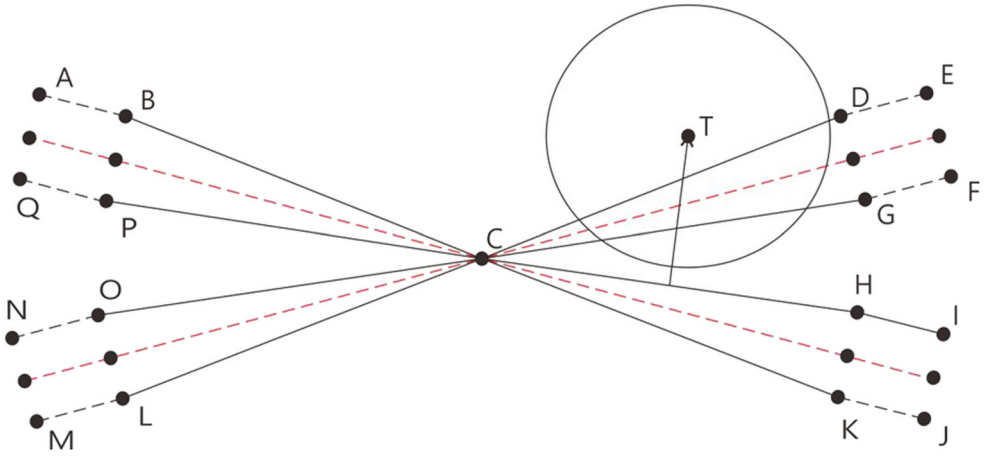

Fig. 9 A failing situation of the traditional advancing front method. Red dashed lines are central lines of two fractures. Black solid segments are front edges. ST crosses CG and CD, are there is no alterate front point in the searching region (the circle)

so-called combined Delaunay-advancing method [41, 42] can avoid this situation, but the dual grid of the triangular mesh may violate the outlines of fractures. In this work, we proposed a modified advancing front method, using multiple independent edges instead of one integral front. Each edge has a unit normal vector pointing outward. The edges do not necessarily connect head-to-tail. With this method, the outlines of the geological objects are still represented faithfully, and the front advancing process can finish unconditionally. This new method is more suitable for building reservoir grids, especially for cases with many fractures.

We use a list to store the front edges that are to be expanded. These front edges are named "active front edges." Initially, the list contains all of the edges made of the control points. Each active front edge is used only once. Whether or not the trial point is valid, the corresponding front edge is removed from the head of the list. The validity of the trial point is checked locally as soon as it is created. The local searching algorithm utilized here is the "background chessboard" method [37]. The whole region is divided as a $N_x \times N_y$ chessboard. The lattice to which any point or edge belongs is easy to identify. The density information is stored on the vertices of the lattice. The detailed algorithm is listed as follows:

1) Initializing the front. The initial front is a series of closed connecting lines of the control points of the geological objects, including faults, boundaries, fractures, horizontal wells, and vertical wells. The control points of boundaries and horizontal wells are set up in the same way as the control points of faults and fractures, respectively. The control points of vertical wells are vertices of nested equilateral polygons. An example of the initial front is shown in Fig. 10a.
2) Selecting the edge at the head of the list (S_1 in Fig. 10b), assuming its length is l.
3) Finding the background lattice that contains the midpoint of S_1. The density value m is obtained by inverse distance weighted interpolation of the four vertices of the lattice.

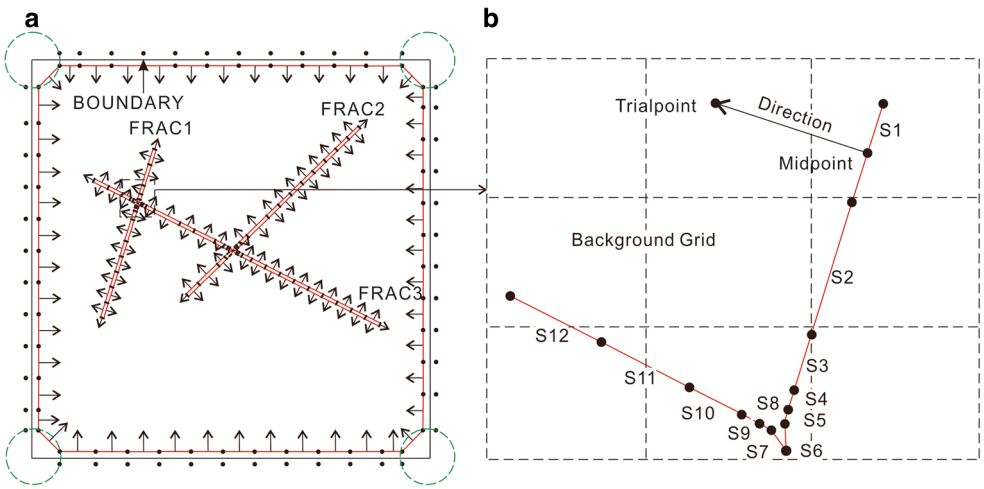

Fig. 10 The modified advancing front method. **a** The initial front; **b** finding an ideal trial point for an edge with the outward direction being defined

4) Computing the position of ideal trial point P_i on the perpendicular bisector of S_1. The distance is according to Farrashkhalvat's criterion [26]:

$$d = \begin{cases} 0.55\,\mathrm{l} & \text{if } 1/\mathrm{m} < 0.55\,\mathrm{l} \\ 1/\mathrm{m} & \text{if } 0.55\,\mathrm{l} < 1/\mathrm{m} < 2.0\,\mathrm{l} \\ 2\,\mathrm{l} & \text{if } 1/\mathrm{m} > 2.0\,\mathrm{l} \end{cases} \tag{9}$$

This formula keeps the triangle acute by constraint d.

5) Searching the nearby front points. A recommendation for the searching radius is $r = 0.8\,d$ [26]. If any front points lie inside the circle, choose the closest one to replace the ideal trial point.
6) If the following conditions are all satisfied: (1) there are no front points near the trial point, (2) the trial point falls into the outer region, and (3) neither of the new edges crosses existing front edges; the trial point is reserved. Otherwise, the trial point is abandoned.
7) Pop the edge from the list of active front edges. If step 6 succeeds, two new active front edges are formed and pushed back to the list. Return to step 2.
8) Repeat steps 2 to 7 until the list of active edges is empty.

In steps 5 and 6, the searching of nearby points and intersected edges is not performed globally. As the whole region is divided as a $N_x \times N_y$ chessboard, searching only occurs in the lattices that contain the searching region or the new edges.

Another benefit of the modified front advancing method is the potential for parallelization. As the point insertion operation only involves nearby front edges, edges that are sufficiently far away can advance simultaneously, and the whole algorithm can be accelerated with the divide-and-conquer method.

3.3.3 Delaunay triangulation

The modified advancing front method inserts points into the domain but does not create triangles. The control points and the inserted points form a point set suitable for Delaunay triangulation. Delaunay triangulation is a classical and well-developed technique. In this work, we use the program "Triangle" [43] to triangulate the point set. "Triangle" is a fast, robust, and open-source program for Delaunay triangulation.

The final polygon mesh is the dual grid of the Delaunay triangular mesh. The dual grid is created by connecting the circumcenters of the adjacent triangles. In the Delaunay triangular mesh, since no extra points fall inside of the circumcircle of any triangle, the circumcenter connecting line of two adjacent triangles is always the perpendicular bisecting their common edge. Therefore, the final polygon mesh is a PEBI grid. The orthogonality of the PEBI grid is a favorable property for the CVFD scheme. Figure 11 shows the reference mesh of a case with three fractures. The red dotted lines are the Delaunay triangular mesh; the black solid lines are the reference mesh.

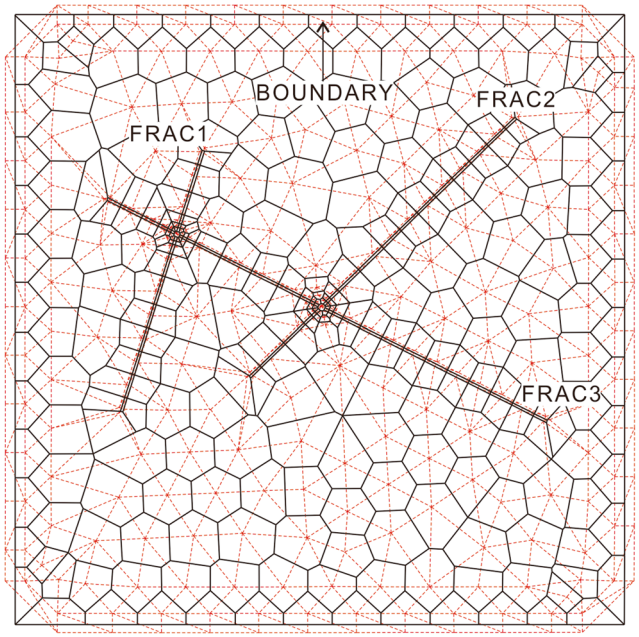

Fig. 11 Generation of the reference mesh. Red dotted lines are the Delaunay triangular mesh; black solid lines are the reference mesh

The steps of constructing the reference mesh are listed in Table 1.

3.4 Mesh morphing

In GPG, all 3D layers have the same topology. This feature simplifies the flow calculation for reservoir simulation, but from a pragmatic perspective, the polygons of a same column should deform to adapt to geological objects inclining from verticality. By definition, the pillars of GPG can bend. This feature offers the possibility to create deformed, but homeomorphic, 3D layers. An example is presented in Fig. 12, in which the absolute positions of the fracture elements (red lines) are distinctly different on two layers, while the topology does not change. The first layer (Fig. 12a) is the 2D reference mesh. The second

Table 1 The steps of constructing the reference mesh

1. Place control points of the determined geological objects.
2. Construct the $N_x \times N_y$ background chessboard, discrete the density field equation, and compute the density values on the vertices.
3. Distribute the remaining points using the modified advancing front method.
4. Perform Delaunay triangulation and obtain the dual mesh.

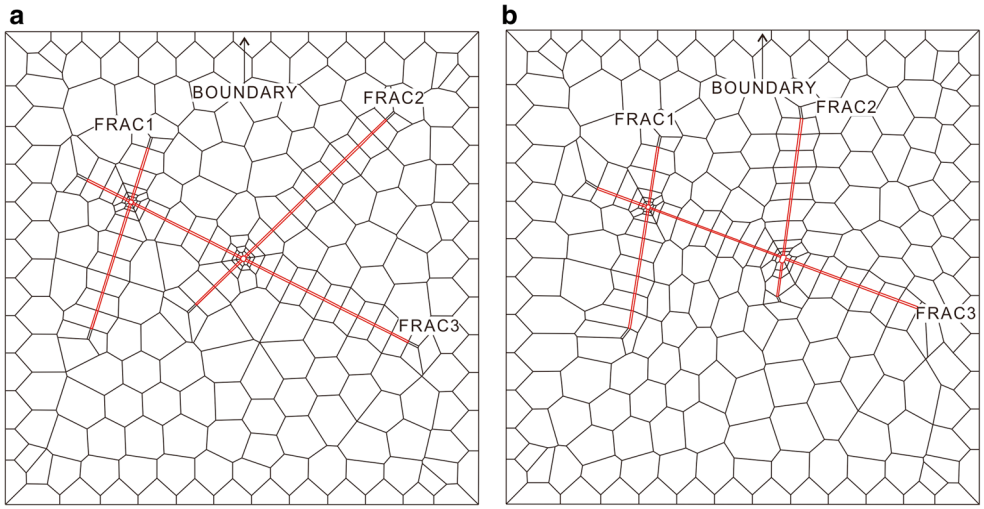

Fig. 12 Mesh morphing. **a** The reference layer; **b** the deformed layer, which is a homeopmorphic mapping of the reference layer

layer (Fig. 12b) is a homeomorphic mapping of the first layer.

In the 2D reference mesh, the control points of geological objects are predetermined. However, as the geological objects may incline, their vertical projections onto a 3D layer may shift from the reference layer. The shifting of a predetermined point may exceed the size of the polygon to which it belongs. Locally adjusting the x–y coordinates of the inserted points could save the convexity of one polygon, but may destroy others' or, even worse, cause the overlapping of polygons. In this work, we introduced a method that adjusts the x–y coordinates of inserted points globally. This guarantees both convexity and topological equality. This method is performed once for each 3D layer.

The problem of mesh morphing is to fix some points and re-distribute the others. In computational graphics, this problem is called mesh manipulation. Some methods are readily available, including: (1) the free-form deformation method (FFD) [44–46]; (2) the differential domain deformation method [47–49]; and (3) the equilibrium spring method [30, 50, 51]. FFD is the most direct method but only modifies the grid locally. FFD is based on absolute Euclidean coordinates, while the differential domain deformation method is based on differential coordinates. The differential coordinate is defined as the difference between a point and the average of its neighbors. The differential domain deformation method is a global method, but in our problems, it usually produces overlapped polygons. The equilibrium spring method simulates edges as springs and predetermined points as anchors. The displacements of inserted points are solved by minimizing the overall elastic potential energy, which is a quadric form. This method is also global. In this work, we found that it rarely produces overlapped polygons.

The overall elastic potential energy is written as:

$$E=\frac{1}{2}\sum_{i}\sum_{j\in N(i)}k_{ij}\left\{\left[\left(x_i-x_i^0\right)-\left(x_j-x_j^0\right)\right]^2+\left[\left(y_i-y_i^0\right)+\left(y_j-y_j^0\right)\right]^2\right\} \tag{10}$$

where $N(i)$ is the set of neighboring points of i; both $i$and $j$include all fixed points; (x_*, y_*) and $\left(x_*^0, y_*^0\right)$ are the new and original x–y coordinates of point "*," respectively; the subscript "*" stands for i or j, and k_{ij} is the Hook's coefficient of spring i–j.

The original x–y coordinate of point i—$\left(x_i^0, y_i^0\right)$ is its coordinate on the reference layer. Vector $\left(x_i - x_i^0, y_i - y_i^0\right)$ is the 2D displacement of point$i$ after mesh morphing. As proposed by Batina [30], the Hook's coefficient is taken as the inverse of the initial spring length (11):

$$k_{ij}=\frac{1}{\sqrt{\left(x_i^0-x_j^0\right)^2+\left(y_i^0-y_j^0\right)^2}} \tag{11}$$

Minimizing the elastic potential energy is to let function (9) have zero first-order derivatives, which is equivalent to solving the x- and y-direction static equilibrium equations, respectively. These equations are two sparse linear equations with variable vectors of **x** and **y**, respectively. In this work, the two equations are solved using Pardiso. Figure 13 shows the example that has already appeared in Figs. 10, 11, and 12. The 2nd and 3rd layers are produced using the equilibrium spring method. As Fig. 13 shows, all polygons are convex, the layer morphing is smooth, and the topology is constant.

The equilibrium spring method solves the issue of determining x- and y-coordinates for points on 3D layers. For z-coordinate evaluation, we use discrete smoothing

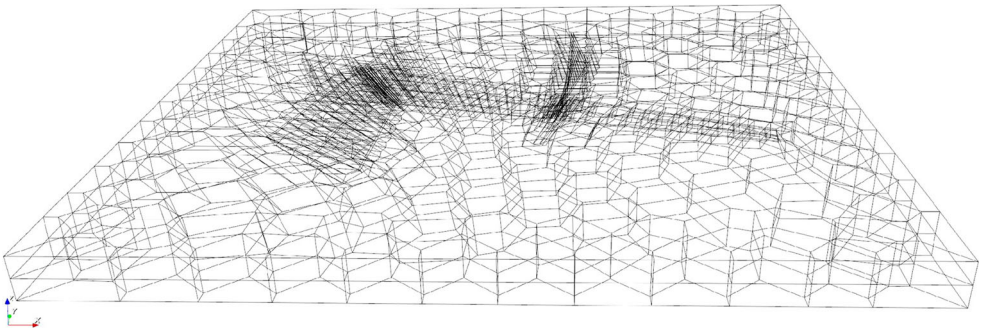

Fig. 13 A 3D GPG example with inclined fractures

interpolation, which is a classic method to assign properties to scattered points.

4 Data format of GPG

In the workflow of a reservoir simulation, GPG is created by the gridding program, saved as a file, and loaded by the simulator. The general way to store polygon grids is to store vertex coordinates and the connection list. The connection list defines which vertices form faces and which faces form bodies. However, as GPG has a layered structure, the storage of GPG can be greatly simplified. In this work, we define a data format that contains the essential geometric information of GPG and is also friendly to the simulator. The data consist of pillar points, pillar indices of columns, and z-coordinates of vertices. This format consumes much less space than the general format. Instead of storing all vertices and the 3D connection list, this format only stores pillar points and the 2D connection list. The total number

Table 2 The text formats of GPG

Keywords	Contents
GPG	5 integers, indicating the size of memory space: 1st integer: the total number of grids (N_G) 2nd integer: the number of grid vertices (N_V). The overlapped vertex of 2 adjacent grids is counted as 2 vertices. The pinch-out vertex of the upper and lower faces of a grid is also counted as 2 vertices. 3rd integer: the number of layers (N_{ly}) 4th integer: the number of pillars (N_P) 5th integer: the total number of points that constitutes pillars (N_{PP})
PCOORD	Integers and real numbers, specifying pillar points. Each pillar data are 1 line of data. In each line of data, the first datum is an integer (n_{PP}), which specifies the number of points that constitutes the pillar. The next $n_{PP} \times 3$ real numbers are the $x-$, $y-$, and $d-$coordinates (d is depth) of the n_{PP} points.
PIND	Integer arrays, the 2D connection list. Each array defines a polygon in 2D, by specifying the pillar indices counter-clockwise. The first entry of each array is the length of the array. As all layers are aligned in GPG, PIND only contains the data for 1 layer.
PZCORN	Real number array with the length N_V, specifying the depths of grid vertices. Each grid block's data are a group with length n_V (the number of vertices of the grid). Each grid block has its upper and lower faces. The pinch-out vertex of the top and bottom faces is counted as 2 vertices, and thus n_V must be an even number. In each group, the first $n_V/2$ numbers are the vertex depths of the upper face and the last $n_V/2$ numbers are the vertex depths of the lower face. The arrangement order of each $n_V/2$ vertices is consistent with the order of pillars in PIND.
PFAULTS	Tables, each of which defines a fault. 1st column: the name of the fault 2nd column: the starting layer of the fault (k_1) 3rd column: the terminal layer of the fault (k_2) 4th column: the flow conductivity multiplier (m_1) 5th column: the heat conductivity multiplier (m_2) 6th column: the number of pillars that constitute the fault (N_{FP}) 7th to ($N_{FP} + 6$)th columns: the pillar indices of the fault The N_{FP} pillar indices are arranged from left to right or right to left in the top-down view. For 2 adjacent grids separated by the fault surface, if 1 of them is located between layers k_1 and k_2 (including k_1 and k_2 layer), the flow or heat conductivity between the two grids is multiplied by m_1 or m_2, respectively.

Table 3 Finding horizontal connections for GPG

1. From the "PIND" data, collect all 2D edges as data structures $[(p_1, p_2); i]$, where p_1 and p_2 are pillar indices and i is the grid index. i refers to the grid in the reference layer. Make sure that $p_1 < p_2$. Push the edges into a vector.
2. Arrange the vector in ascending order by comparing the (p_1, p_2) tuples.
3. If 2 adjacent nodes in the vector have the same (p_1, p_2) components but different "i" components, the 2 grids contact through a side face.
4. Push back $[(p_1, p_2); (i_1, i_2)]$ to a list. The list is the connection table of the reference layer or the "reference connection table."
5. In each 3D layer, the grid connections follow the reference connection table, unless a fault appears. When a fault appears, grid blocks on different layers should be examined to determine whether they contact.

of pillar points is less than or equal to the total number of polygon vertices. A straight vertical pillar can be defined by only one point. The 2D connection list is also shorter than the 3D connection list. Another advantage of this format is the simplification of fault definition. Instead of enumerating all fault faces, the fault is defined directly with the corresponding pillars and the z-coordinates of fault faces.

The GPG data can be saved as readable text. The text is divided into five sections, each of which is headed by a title. The titles and their contents are listed in Table 2. An example that illustrates the data format is provided in Appendix 1.

5 Calculating transmissivities for GPG

5.1 Finding grid connections

The data format of GPG does not explicitly define any connections or transmissivities. These questions are left to the simulator to answer. The simulator reads in the data file of GPG detects all grid connections and calculates the transmissivities. A benefit of GPG is the simple neighboring relationship, which can be summarized as follows:

1. Any two grids possess at most one interface.
2. In the same column of grid blocks, the top face of a grid in the lower layer and the bottom face of a grid in the upper layer are always aligned, and if the depths of the two faces are identical, the two grids are connected vertically.
3. Two adjacent grids within the same layer may contact through a side face, and the shape of the interface depends on the coordinates of the face nodes.

Finding vertical connections for GPG is direct, while finding horizontal connections requires an algorithm, which is presented in Table 3.

In step 5 of Table 3, the contacted area of a cross-layer grid couple may not be quadrilateral. Determining the contacted area constitutes a special case of finding the overlapped area of two polygons, where the two polygons are quadrilaterals. For two quadrilaterals with a pair of opposite edges being collinear, there are a total of 35 types of overlapping, some of which are presented in Fig. 14.

5.2 Calculating transmissivity factors and well indices

We use two-point flux approximation (TPFA), which is a CVFD scheme, to calculate transmissivity factors. TPFA allows unaligned grid couples, e.g., two grids on different sides of a fault. In GPG, the permeability can have vertical-horizontal anisotropy, i.e., $k_z \neq k_h$, where k_z is vertical permeability and k_h is horizontal permeability. GPG has good orthogonality, because the reference layer is a PEBI grid and the 3D layers are topologically matched. Near the inclined fractures, the geometric orthogonality may break. However, as the dip angles of fractures are usually small, the introduced error is not significant.

Calculating well indices in GPG is also simple, because the grid blocks near the wellbore are specially treated during grid generation. The grid blocks surrounding vertical wells are like radial grids; whereas, the grid blocks surrounding horizontal wells are like rectangular grids. In this work, we use Palagi and Aziz's method .....[22, 28] to calculate vertical wells' indices and Peaceman's formula ....[52] to calculate horizontal wells' indices.

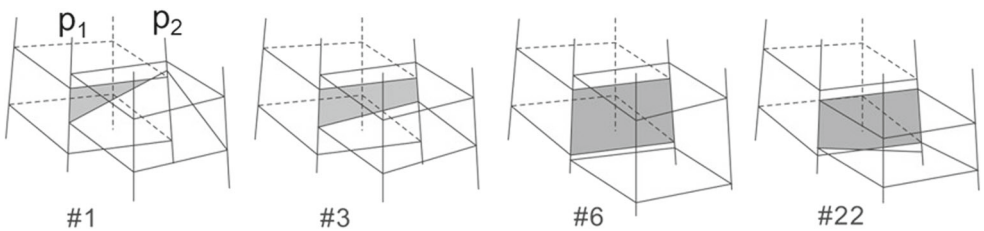

Fig. 14 Some types of fault connections

Table 4 Computational performance of the GPG generation algorithm (time elapsed in seconds)

Case number	Vertex number	GPG cell number	Point distribution (s)	Delaunay triangulation (s)	Mesh morphing (s)	Total time (s)
1	81396	6859	0.267	0.015	0.800	1.082
2	1318980	112100	2.669	0.022	1.000	12.691
3	376428	31452	2.420	0.022	0.720	3.202
4-1	2015248	216428	77.295	0.084	4.044	81.420
4-2	2720424	276024	133.305	0.096	6.664	134.065
4-3	3604272	350364	201.413	0.115	10.508	212.036
4-4	5762528	530048	394.932	0.154	22.604	417.69
4-5	8254640	737056	679.070	0.217	36.198	715.485
4-6	10263792	904280	953.579	0.254	44.840	998.693

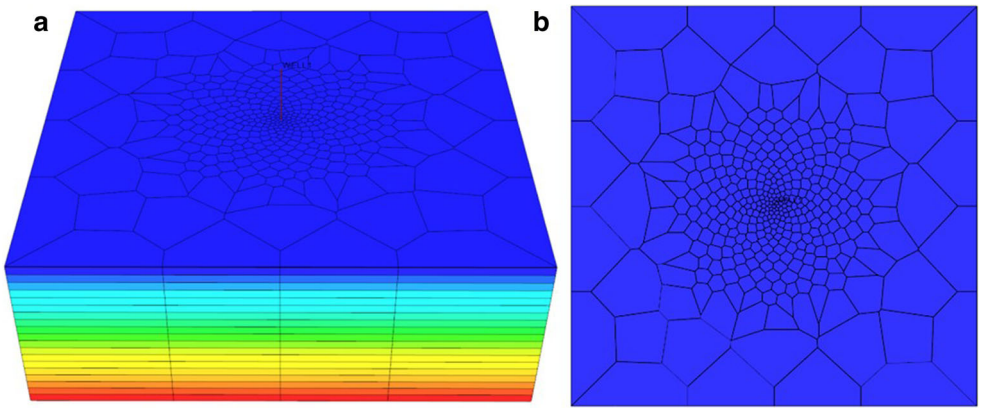

Fig. 15 A view of case 1. **a** The 3D view; **b** the top view

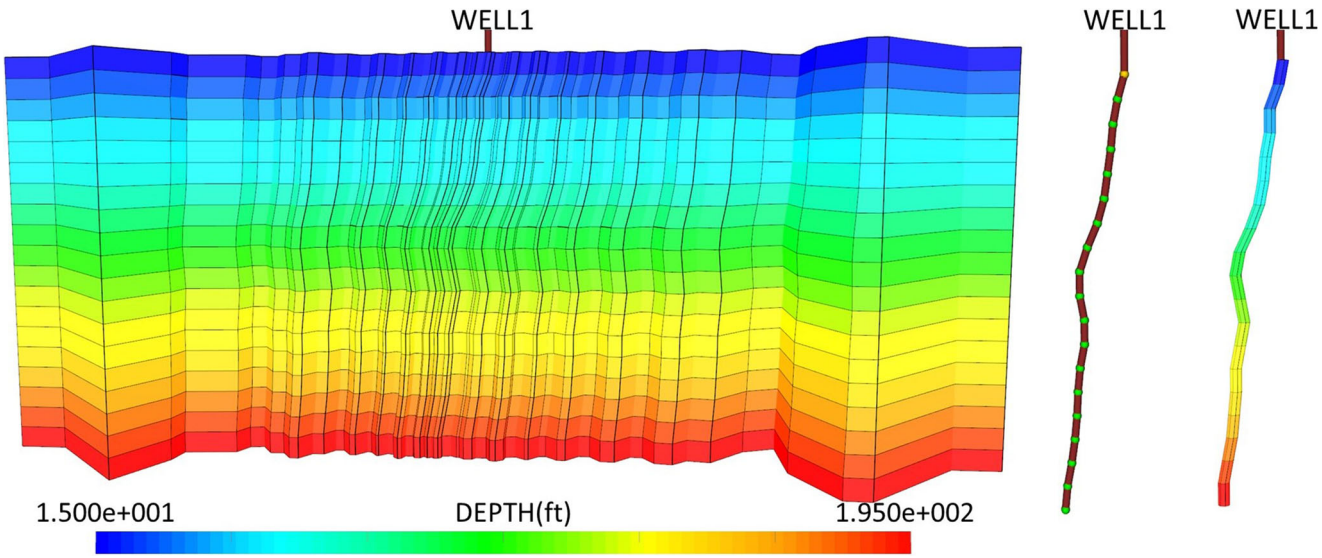

Fig. 16 Another view of case 1. **a** The $x - z$ cross section of the mesh; **b** the well trajectory; **c** and the isolated well grid blocks. The color scale represents depth

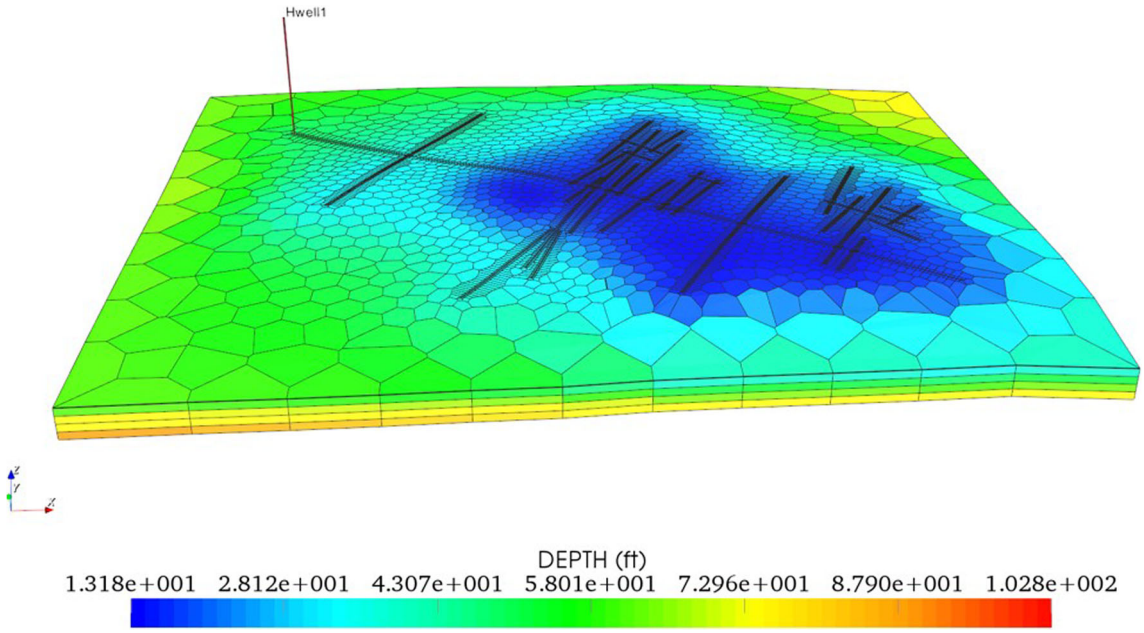

Fig. 17 The 3D view of case 3. The color scale represents depth

6 Examples

In this section, we present four cases to illustrate the features of GPG. These cases cover the major situations that are difficult for CPG. They include twisted well trajectory, declining fractures, 3D faults, and horizontal fracturing wells. All cases are simulated using a reservoir simulator. We integrate the code of GPG data format interpretation and transmissivity calculation to a reservoir simulator named the unconventional oil and gas simulator (UNCONG) ...[53, 54], so that the simulator connects the GPG gridding program seamlessly. UNCONG is a compositional reservoir simulator for shale oil, shale gas, CBM reservoirs, etc. A key development in UNCONG is to integrate state-of-the-art methods to model fractured reservoirs. We compared the simulation results of case 3 with the results of CPG to cross-validate the two types of grids in the simulator. In case 4, we compared the simulation results of the versions

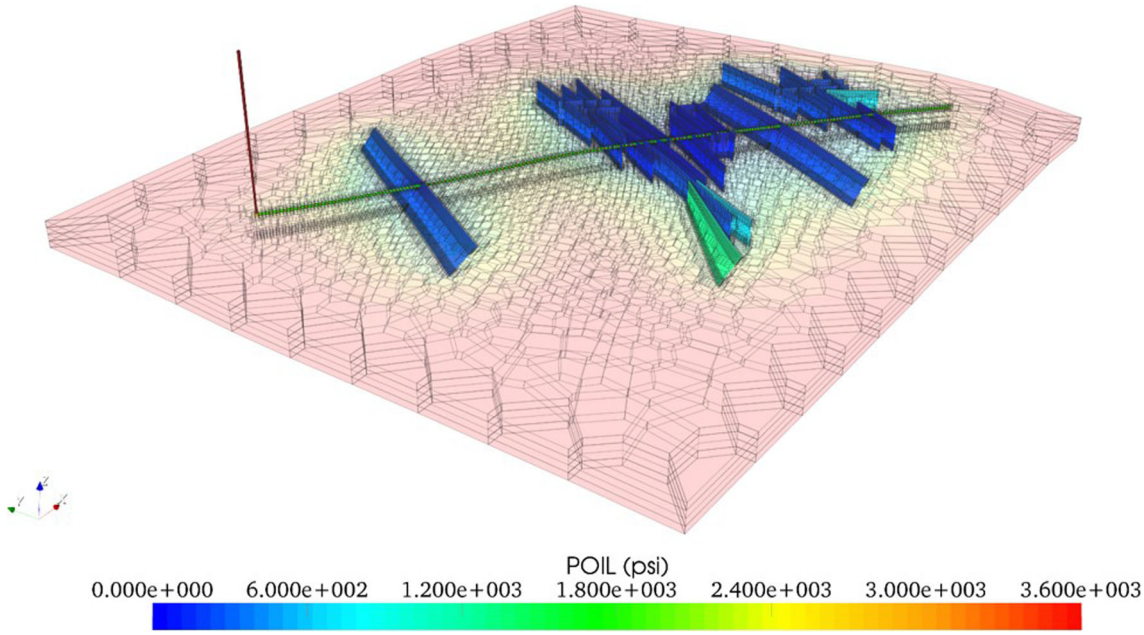

Fig. 18 View of the fracture grids. The color scale represents depth

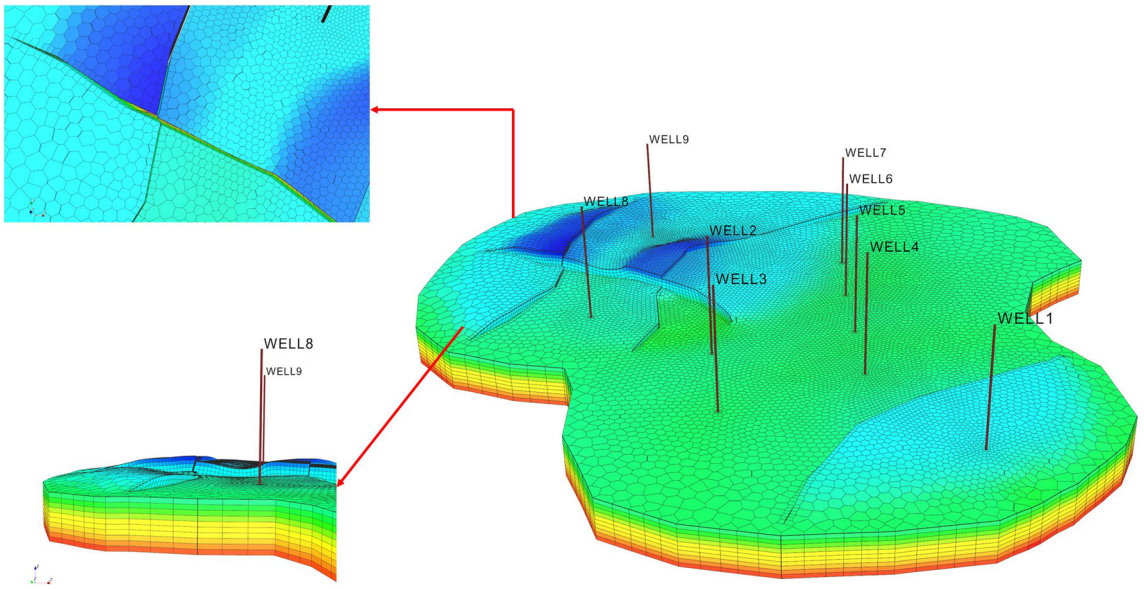

Fig. 19 The resultant GPG for case 3. The color scale represents depth

with different grid densities to verify the convergence of precision. The time costs of the four cases are summarized in Table 4.

Case 1: twisted well trajectory Case 1 is a vertical well with a twisted trajectory in a square domain. Figure 15 shows the resultant grid, which has 6,859 grid blocks. The grid blocks near the well are refined smoothly. The density of grid blocks near the well can be adjusted by changing the value of the intensity factor (ψ in Eq. (2)). The top layer is the reference layer. Other layers are generated by deforming the reference layer with the boundaries being fixed and the central grid blocks shifting with the well trajectory. The pillars of the central prism are bent to conform to the well trajectory, so that the borehole is represented faithfully. The cut view of the mesh and the isolated well grid blocks are presented in Fig. 16.

Case 2: declining fractures Case 2 is a rectangular reservoir with a horizontal fracturing well and some large fractures. Figure 17 shows the resultant grid, which has 37,064 grid blocks. The grid blocks are refined near the horizontal well and the fractures naturally. In case 2, the top layer is also the reference layer. Other layers are homeomorphic mappings of this layer with the boundaries being fixed and the fracture grid blocks morphing with depth. Figure 18 presents the fracture grids by making the matrix grids transparent. The grid faithfully represents the fractures. The pillars of the prisms are bent to conform to the fracture shapes.

Case 3: 3D faults Case 3 is a faulted reservoir with nine vertical wells. The reservoir is modeled with both GPG and CPG. Figure 19 shows the resultant GPG, which has 112,100 grid blocks. The zoomed-in views reveal that the fault pillars are bent according to the fault shapes, and

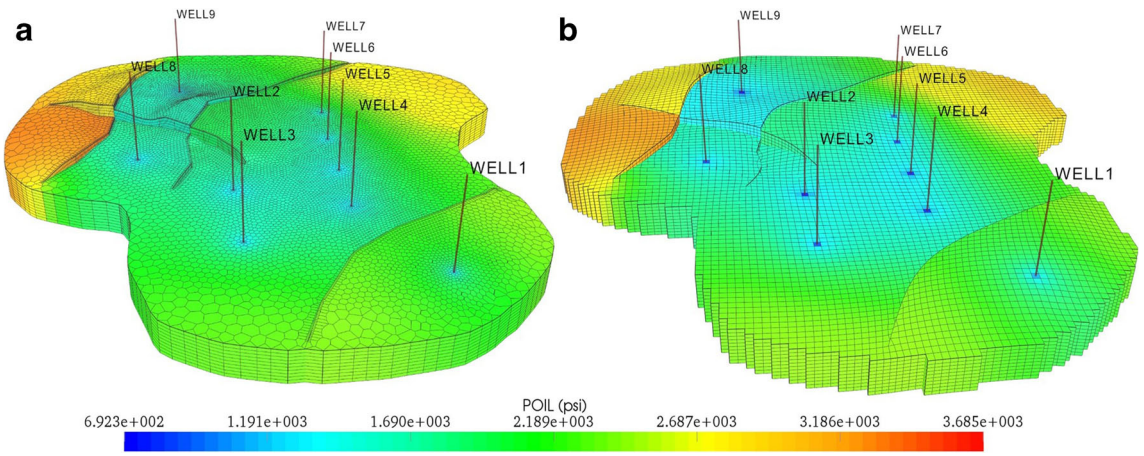

Fig. 20 The pressure distribution at day 400. The color scale represents depth

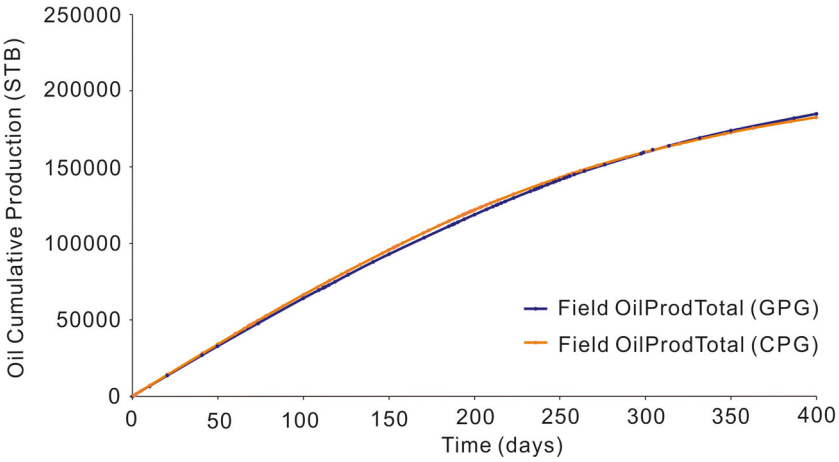

Fig. 21 Comparison of field oil total production of GPG and CPG

the faults are represented as 3D bodies. The reservoir is simulated dynamically using UNCONG. The effect of faults can be observed in Fig. 20, which presents the pressure distribution of GPG and CPG cases at day 400. Figure 21 is the comparison of field oil total production of GPG and CPG cases. Good agreement between GPG and CPG is achieved.

Case 4: horizontal fractured wells Case 4 is a reservoir with three horizontal fractured wells and many fractures. We generate six models (Fig. 22) with different grid intensities. The grid numbers are 216,428, 276,024, 350,364, 530,048, 737,056, and 904,280. The transparent view (Fig. 23) of the grid, whose grid number is 216,428, shows that the gridding algorithm can constrain the intersection grids to sizes that are similar to those of the non-intersected fracture grids. We simulate these models using UNCONG. The pressure distributions of the six models at the 1100th day are presented in Fig. 22. The curves of total gas production are compared in Fig. 24. From these results, we can confirm the convergence of precision with the growing number of grids, and an asymptote exists. When the number of grids is larger than a threshold and continues growing, the simulation

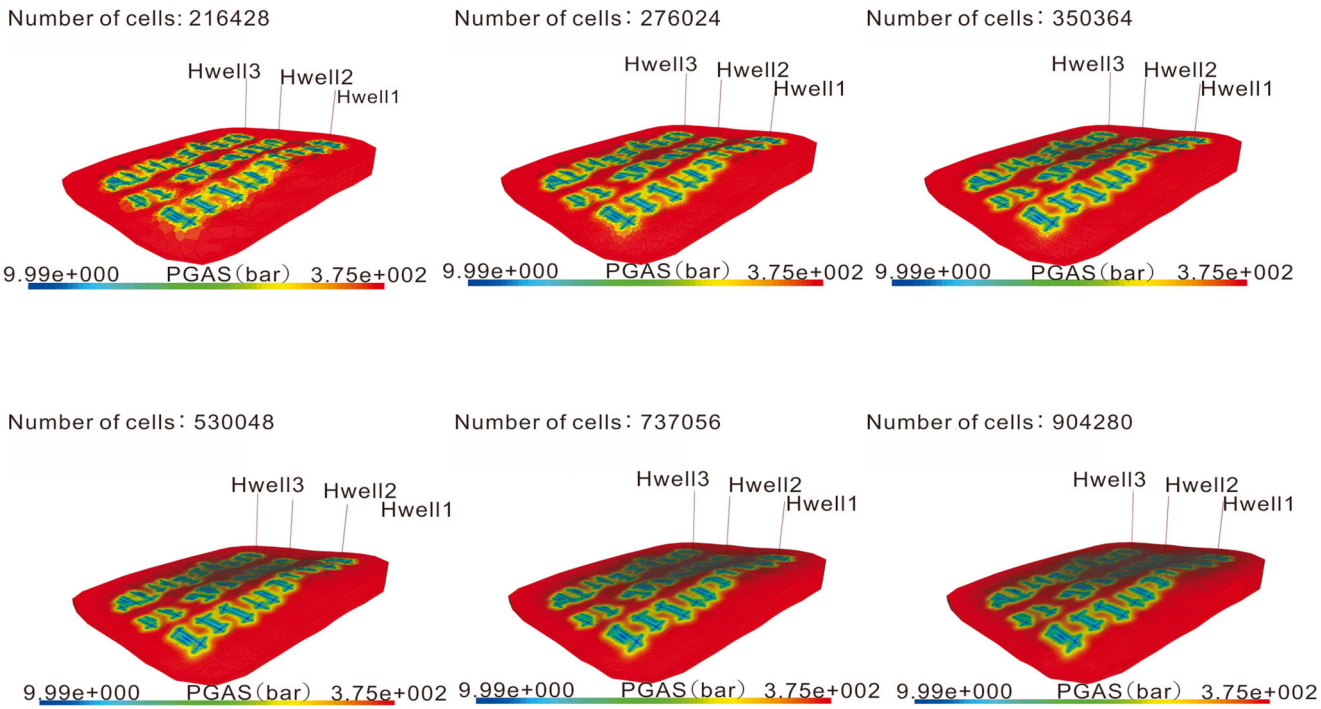

Fig. 22 The GPG of the six models, which have different grid densities. The color scale represents pressure

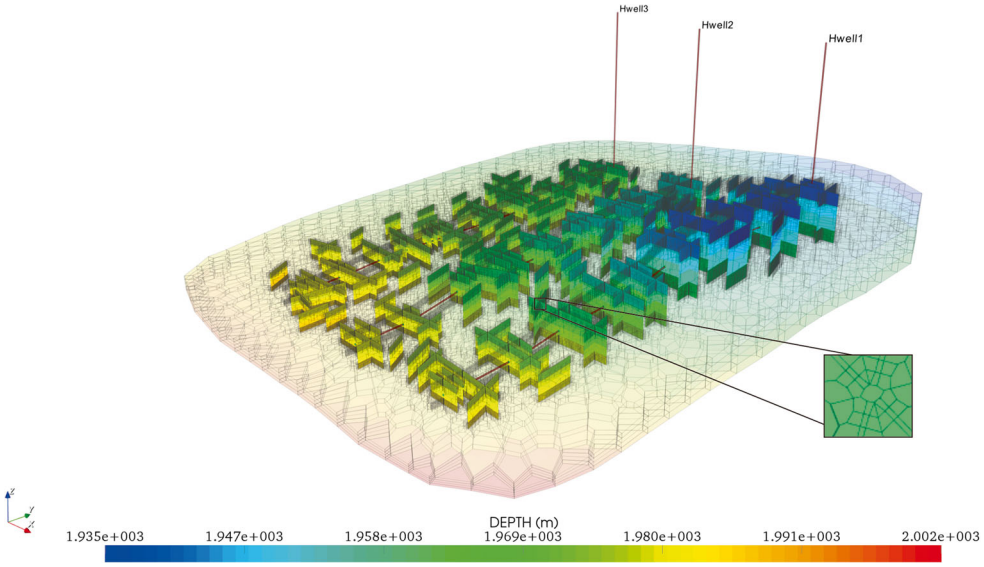

Fig. 23 The transparent view of the GPG with 216,428 cells. The color scale represents depth

results tend to be stable. In this case, the threshold is approximately 300,000.

Computational performance We evaluate the performance of the gridding algorithm via the four cases above. The results are listed in Table 4. All cases are tested on a single CPU running at 2.60 GHz. The gridding program is not parallelized. The total CPU time equals the summation of time spent in point distribution, Delaunay triangulation, and mesh morphing. The time of height interpolation is not included. The time costs of point distribution, Delaunay triangulation, and each layer's mesh morphing almost grow linearly with the vertex number per layer. The point distribution is always the most time-consuming part of the whole algorithm. As we described in Section 3.3, the point distribution includes density field calculation and point insertion (the advancing front method). The former requires solving large sparse linear equations, and the latter involves a large amount of distance and intersection calculations. Fortunately, both steps can be accelerated by parallelization. The solving of the pentadiagonal matrix could easily have a near-linear acceleration ratio. In fact, it is already quite optimized in Pardiso. Since the modified advancing front method can be parallelized with the divide-and-conquer strategy, it could also have a near-linear acceleration ratio. The mesh-morphing step takes a minor part of the total time. Although the equilibrium spring method also requires solving large sparse linear equations, in these cases, the time cost is much less than point insertion. Moreover, solving an unstructured, sparse, and symmetric positive definite matrix is already quite optimized in Pardiso. The Delaunay triangulation step comprises a very small part of the total time, because after point distribution, we already have a very good point set for Delaunay triangulation. Consequently, accelerating the point insertion step is critical to improve the speed of the current algorithm. It also constitutes an important part of our future work.

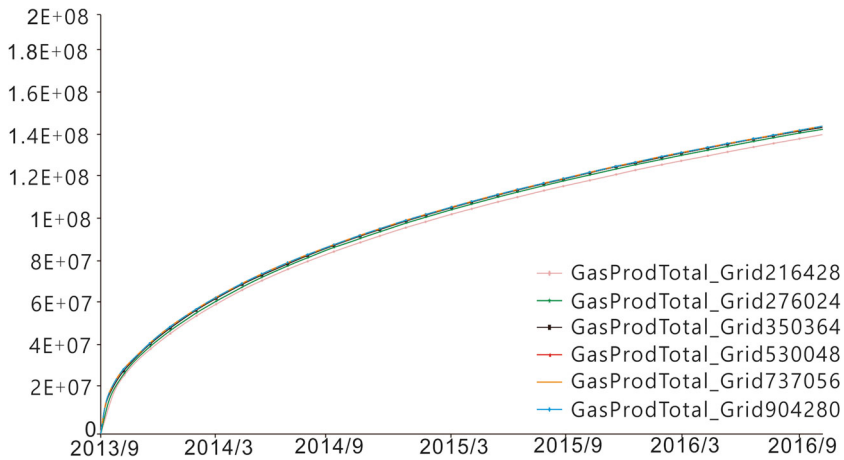

Fig. 24 Comparing gas total production curves of the six models of case 4

7 Conclusions

This paper presents a novel 3D unstructured grid, the GPG, for reservoir simulations. GPG is a pillar-based layered grid. In the design of GPG, the pillars can overlap or bend, the layers can slit or degenerate, and the cells can deform through depth. These flexibilities enable GPG to model complicated geological structures, including horizons, pinch-outs, faults, fractures, and boreholes, with geometric details preserved. We propose an effective gridding algorithm that fulfils the features of GPG.

The implementation of GPG is a systematic work. We designed a file format that contains only the necessary information of GPG but is still easy to interpret. We also integrated the ability of modeling with GPG into a reservoir simulator. The methods that we utilized to calculate the transmissivities and well indices are presented.

Cases are presented to demonstrate the flexibility, accuracy, and practicability of GPG, as well as the efficiency of the proposed gridding algorithm. GPG is capable of representing both high permeability zones and barrier zones with a reasonable number of grids and rather simple topology, which benefits its applications in reservoir simulations. The final example highlights the capabilities of GPG. With GPG, very complicated fracture networks are represented faithfully, while the geometric topology remains simple. The error converges quickly with the increasing of grid number. Compared with the traditional CPG, GPG is much more flexible yet still pragmatic. GPG performs better in capturing geological structures while minimizing grid orientation effects. The conciseness and flexibility of GPG make it a potential new standard grid format that can replace CPG.

A challenge with GPG is the difficulty to model anisotropic permeability in the x–y plane. If using the simple two-point flux approximation (TPFA), the "K-orthogonal" condition would be violated and the flow calculation would lose accuracy. The NTPFA or MFD method may be employed to keep the accuracy in the non-K-orthogonal grid, which constitutes a topic of our ongoing work.

Funding Information This work is partially funded by the National Natural Science Foundation of China (Grant Nos. U1663208 and 51520105005), the Beijing Municipal Science and Technology Commission (Z171100002317022), and the National Science and Technology Major Project of China (Grant Nos. 2016ZX05037-003 and 2017ZX05049-003). The data used are available upon request from the corresponding author.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

Appendix 1: A simple example to illustrate the data format of GPG

In the following script, each "#" leads a comment line. The grid is shown in Fig. 25.

GPG

N_G N_V N_{ly} N_P N_{PP}

8 72 2 11 26

PCOORD

n_{PP} x_1 y_1 d_1 x_2 y_2 d_2 x_3 y_3 d_3

2 41.3 54.9 0.0 41.3 54.9 8.0

3 41.7 41.7 0.0 41.7 41.2 6.0 41.7 40.7 8.0

2 52.0 54.8 0.0 52.0 54.8 8.0

2 14.7 53.8 0.0 14.7 53.8 8.0

3 20.8 40.4 0.0 20.8 39.4 6.0 20.8 38.4 8.0

3 29.9 34.3 0.0 29.9 33.8 6.0 29.9 33.3 8.0

2 41.3 70.1 0.0 41.3 70.1 8.0

2 59.9 63.8 0.0 59.8 58.0 5.0 59.8 58.0 8.0

2 59.8 58 0.0 59.8 58.0 8.0

2 28.0 75.6 0.0 28.0 75.6 8.0

2 14.2 66.2 0.0 14.2 66.2 8.0

PIND

Length and Pillar indices

3 1 2 3

5 4 5 6 2 1

5 1 7 8 9 3

5 1 7 10 11 4

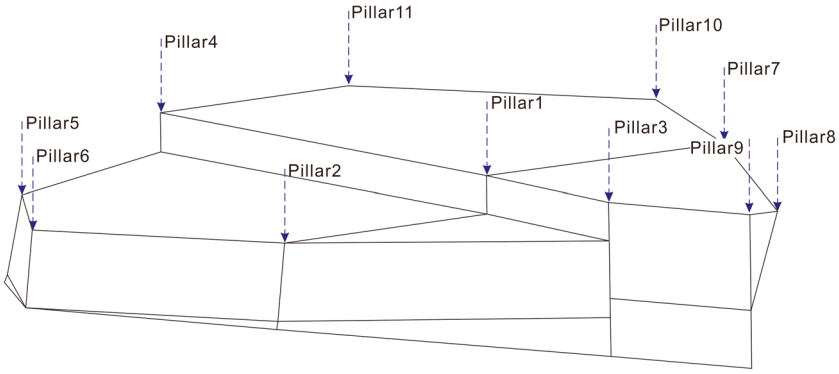

Fig. 25 The 3D view of the simple GPG example

PZCORN

Every two lines are to define the depths of the top and the

bottom faces of a prism grid, respectively. The order of

numbers is coincided with the order of pillars defined in

PIND.

2.0 2.0 2.0

6.0 6.0 6.0

2.0 2.0 2.0 2.0 2.0

6.0 6.0 6.0 6.0 6.0

0.0 0.0 0.0 0.0 0.0

5.0 5.0 5.0 5.0 5.0

0.0 0.0 0.0 0.0 0.0

5.0 5.0 5.0 5.0 5.0

6.0 6.0 6.0

8.0 6.4 8.0

6.0 6.0 6.0 6.0 6.0

8.0 6.4 6.0 6.4 8.0

5.0 5.0 5.0 5.0 5.0

8.0 8.0 8.0 8.0 8.0

5.0 5.0 5.0 5.0 5.0

8.0 8.0 8.0 8.0 8.0

PFAULTS

name k_1 k_2 $m_1 m_2$ N_{FP} p_1 p_2 p_3

'Fault1' 1 2 0 0 3 4 1 3

References

1. Aavatsmark, I., Eigestad, G.T., Mallison, B.T., Nordbotten, J.M.: A compact multipoint flux approximation method with improved robustness. Numerical Methods for Partial Differential Equations **24**(5), 1329–1360 (2008). https://doi.org/10.1002/num.20320
2. Aavatsmark, I., Eigestad, G.T., Heimsund, B.-O., Mallison, B.T., Nordbotten, J.M., Cian, E.: A New Finite-Volume Approach to Efficient Discretization on Challenging Grids. Paper Presented at the SPE Reservoir Simulation Symposium, Houston, Texas, USA (2007)
3. Aavatsmark, I.: An introduction to multipoint flux approximations for quadrilateral grids. Comput. Geosci. **6**(3), 405–432 (2002). https://doi.org/10.1023/A:1021291114475
4. Brezzi, F., Lipnikov, K., Shashkov, M.: Convergence of the mimetic finite difference method for diffusion problems on polyhedral meshes. SIAM J. Numer. Anal. **16**(02), 275–297 (2005). https://doi.org/10.1137/040613950
5. Brezzi, F., Lipnlkov, K., Simoncini, V.: A family of mimetic finite difference methods on polygonal and polyhedral meshes. Mathematical Models & Methods in Applied Sciences **15**(10), 1533–1551 (2005). https://doi.org/10.1142/S0218202505000832
6. Hyman, J., Morel, J., Shashkov, M., Steinberg, S.: Mimetic finite difference methods for diffusion equations. Comput. Geosci. **6**(3), 333–352 (2002). https://doi.org/10.1023/A:1021282912658
7. Lipnikov, K., Manzini, G., Shashkov, M.: Mimetic finite difference method. J. Comput. Phys. **257**, 1163–1227 (2014). https://doi.org/https://doi.org/10.1016/j.jcp.2013.07.031
8. Nikitin, K., Terekhov, K., Vassilevski, Y.: A monotone nonlinear finite volume method for diffusion equations and multiphase flows. Comput. Geosci. **18**(3), 311–324 (2014). https://doi.org/10.1007/s10596-013-9387-6
9. Klemetsdal, Ø.S., Berge, R.L., Lie, K.A., Nilsen, H.M., Møyner, O.: Unstructured Gridding and Consistent Discretizations for Reservoirs with Faults and Complex Wells. Paper Presented at the

SPE Reservoir Simulation Conference, Montgomery, Texas, USA (2017)
10. Nelson, R.A.: 1. Evaluating Fractured Reservoirs: Introduction. In: Geologic analysis of naturally fractured reservoirs. 2nd edn., pp. 1-100. Gulf Professional Publishing, Woburn (2001)
11. Ding, Y., Lemonnier, P.: Use of corner point geometry in reservoir simulation. Paper presented at the International Meeting on Petroleum Engineering, Beijing China (1995)
12. Aarnes, J.E., Krogstad, S., Lie, K.-A.: Multiscale mixed/mimetic methods on corner-point grids. Comput. Geosci. **12**(3), 297–315 (2008). https://doi.org/10.1007/s10596-007-9072-8
13. Ponting, D.K.: Corner point grid geometry in reservoir simulation. Paper presented at the Proc., First European Conference Math Oil Recovery (1989)
14. Wadsley, W.A.: Modelling reservoir geometry with non-rectangular coordinate grids Paper presented at the SPE Annual Technical Conference and Exhibition (1980)
15. Goldthorpe, W.H., Chow, Y.S.: Unconventional Modelling of Faulted Reservoirs: a Case Study. Paper Presented at the SPE Reservoir Simulation Symposium, Dallas, Texas (1985)
16. Bennis, C., Borouchaki, H., Flandrin, N.: 3D conforming power diagrams for radial LGR in CPG reservoir grids. Engineering with Computers **24**(3), 253–265 (2008). https://doi.org/10.1007/s00366-008-0098-x
17. Quandalle, P.: Eighth SPE Comparative Solution Project: Gridding Techniques in Reservoir Simulation. Paper Presented at the 1 Symposium on Reservoir Simulation, New Orleans, Louisiana (1993)
18. Pedrosa Jr., O.A., Aziz, K.: Use of a hybrid grid in reservoir simulation SPE Reservoir Engineering. https://doi.org/10.2118/13507-PA (1986)
19. Borouchaki, H., George, P.L., Lo, S.H.: Optimal delaunay point insertion. Int. J. Numer. Methods Eng. **39**(20), 3407–3437 (2015). https://doi.org/10.1002/(SICI)1097-0207(19961030)39:203407::AID-NME53.0.CO;2-C
20. George, P.L., Borouchaki, H.: Delaunay triangulation and meshing hermès (1998)
21. Mlacnik, M.J., Durlofsky, L.J., Heinemann, Z.E.: Dynamic Flow-Based PEBI Grids for Reservoir Simulation. Paper Presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas (2004)
22. Palagi, C.L., Aziz, K.: Use of Voronoi grid in reservoir simulation SPE-22889-PA. https://doi.org/10.2118/22889-PA (1994)
23. Fung, L.S.K., Buchanan, L., Ravi, S.: Hybrid-CVFE method for flexible-grid reservoir simulation SPE Reservoir Engineering. https://doi.org/10.2118/25266-PA (1994)
24. Jackson, M.D., Gomes, J.L.M.A., Mostaghimi, P., Percival, J.R., Tollit, B.S., Pavlidis, D., Pain, C.C., El-Sheikh, A.H., Muggeridge, A.H., Blunt, M.J.: Reservoir Modeling for Flow Simulation Using Surfaces, Adaptive Unstructured Meshes, and Control-Volume-Finite-Element Methods. Paper Presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA (2013)
25. Vestergaard, H., Olsen, H., Sikandar, A.S., Abdulla, I.A., Noman, R.: The application of unstructured gridding techniques for full field simulation of a giant carbonate reservoir developed with long horizontal wells Paper presented at the International Petroleum Technology Conference (2007)
26. Farrashkhalvat, M., Miles, J.: Basic structured grid generation: with an introduction to unstructured grid generation Butterworth-Heinemann (2003)
27. Thompson, J.F., Soni, B.K., Weatherill, N.P.: Handbook of grid generation CRC Press (1998)
28. Palagi, C.L., Aziz, K.: Modeling vertical and horizontal wells with Voronoi grid SPE-24072-PA. https://doi.org/10.2118/24072-PA (1994)
29. Pirzadeh, S.: Structured background grids for generation of unstructured grids by advancing-front method. Aiaa J. **31**(2), 285–289 (1991). https://doi.org/10.2514/3.11662
30. Batina, J.T.: Unsteady Euler airfoil solutions using unstructured dynamic meshes. AIAA J. **28**(8), 1381–1388 (1989). https://doi.org/10.2514/3.25229
31. Mallet, J.L.: Discrete smooth interpolation. ACM Trans Graph. **8**(2), 121–144 (1989). https://doi.org/10.1145/62054.62057
32. Mallet, J.L.: Discrete smooth interpolation in geometric modelling. Comput. Aided Des. **24**(4), 178–191 (1992). https://doi.org/10.1016/0010-4485(92)90054-E
33. Freitag, L.A.: On combining Laplacian and optimization-based mesh smoothing techniques. ASME applied mechanics division-publications-amd **220**, 37–44 (1997)
34. Field, D.A.: Laplacian smoothing and delaunay triangulations. J. Communications in Applied Numerical Methods **4**, 709–712 (1998). https://ci.nii.ac.jp/naid/80004258763/en/
35. Yang, C., Xue, X., King, M.J., Datta-Gupta, A.: Flow simulation of complex fracture systems with unstructured grids using the fast marching method. Paper presented at the SPE/AAPG/SEG Unconventional Resources Technology Conference, Austin, Texas USA (2017)
36. Branets, L., Ghai, S.S., Lyons, S.L., Wu, X.-H.: Efficient and Accurate Reservoir Modeling Using Adaptive Gridding with Global Scale Up. Paper Presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas (2009)
37. Pirzadeh, S.: Unstructured grid generation by advancing front method using structured background grids. In: Napolitano, M., Sabetta, F. (eds.) 13th international conference on numerical methods in fluid dynamics: proceedings of the conference held at the Consiglio Nazionale delle Ricerche Rome, Italy, 6–10 July 1992, pp. 285-289. Springer, Berlin (1993)
38. Schenk, O., Gärtner, K.: Solving unsymmetric sparse systems of linear equations with PARDISO. Futur. Gener. Comput. Syst. **20**(3), 475–487 (2004). https://doi.org/10.1016/j.future.2003.07.011
39. Schenk, O., Gärtner, K., Fichtner, W., Stricker, A.: PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. Futur. Gener. Comput. Syst. **18**(1), 69–78 (2001). https://doi.org/10.1016/S0167-739X(00)00076-5
40. George, J.A.: Computer implementation of the finite element method. Department of Computer Science, Standford University, CA USA (1971)
41. Mavriplis, D.J.: An advancing front Delaunay triangulation algorithm designed for robustness. J. Comput. Phys. **117**(1), 90–101 (1995). https://doi.org/10.1006/jcph.1995.1047
42. Müller, J.D., Roe, P.L., Deconinck, H.: A frontal approach for internal node generation in Delaunay triangulations. Int. J. Numer. Methods Fluids **17**(3), 241–255 (1993). https://doi.org/10.1002/fld.1650170305
43. Shewchuk, J.R.: Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In: Lin, M.C., Manocha, D. (eds.) Applied Computational Geometry Towards Geometric Engineering: FCRC'96 Workshop, WACG'96 Philadelphia, PA, May 27–28, 1996 Selected Papers, pp. 203-222. Springer, Berlin (1996)
44. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. SIGGRAPH Comput Graph. **20**(4), 151–160 (1986). https://doi.org/10.1145/15886.15903
45. Coquillart, S.: Extended free-form deformation: a sculpturing tool for 3D geometric modeling. SIGGRAPH Comput Graph. **24**(4), 187–196 (1990). https://doi.org/10.1145/97880.97900
46. Modat, M., Ridgway, G.R., Taylor, Z.A., Lehmann, M., Barnes, J., Hawkes, D.J., Fox, N.C., Ourselin, S.: Fast free-form deformation

using graphics processing units. Comput. Methods Prog. Biomed. **98**(3), 278–284 (2010). https://doi.org/10.1016/j.cmpb.2009.09.002

47. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Seidel, H.-P.: Laplacian Surface Editing Paper Presented at the Proceedings of The 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Nice, France (2004)
48. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.-Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans Graph. **23**(3), 644–651 (2004). https://doi.org/10.1145/1015706.1015774
49. Alexa, M.: Differential coordinates for local mesh morphing and deformation. Vis. Comput. **19**(2), 105–114 (2003). https://doi.org/10.1007/s00371-002-0180-0
50. Farhat, C., Degand, C., Koobus, B., Lesoinne, M.: Torsional springs for two-dimensional dynamic unstructured fluid meshes. Comput. Methods Appl. Mech. Eng. **163**(1), 231–245 (1998). https://doi.org/10.1016/S0045-7825(98)00016-4
51. Murayama, M., Nakahashi, K., Matsushima, K.: Unstructured dynamic mesh for large movement and deformation. AIAA paper **122**, 2002 (2002)
52. Peaceman, D.W.: Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. Soc. Pet. Eng. J. **18**(3), 183–194 (1983). https://doi.org/10.2118/10528-PA
53. Li, X., Zhang, D., Li, S.: A multi-continuum multiple flow mechanism simulator for unconventional oil and gas recovery. Journal of Natural Gas Science and Engineering **26**, 652–669 (2015). https://doi.org/10.1016/j.jngse.2015.07.005
54. Li, X., Zhang, D.: A backward automatic differentiation framework for reservoir simulation. Comput. Geosci. **18**(6), 1009–1022 (2014). https://doi.org/10.1007/s10596-014-9441-z