Discretization of Flow Diagnostics on Stratigraphic and Unstructured Grids

A. F. Rasmussen* (SINTEF) K.–A. Lie (SINTEF)

Abstract

Flow diagnostics tools yield quantitative information about the flow behaviour of a model, based on controlled numerical flow experiments. We consider a family of flow diagnostic measures that are constructed based on a single pressure solution and can be used to quickly establish flow patterns and well-allocation factors. This offers a means to rank, compare, and validate reservoir models, upscaling procedures, and production scenarios that is significantly less computationally expensive than full-featured multiphase flow simulations.

All flow diagnostic measures considered herein are defined from time-of-flight and tracer partitions. From these basic quantities, one can compute many interesting diagnostics such as: tracer partitions, drainage and swept regions, well-pair connections, well allocation factors, flow and storage capacity (F-Phi) diagrams, sweep efficiency, and Lorenz heterogeneity coefficients. Time-of-flight and delineation of reservoir volumes are often associated with streamlines, but can equally well be computed from a standard Eulerian discretization like the single-point upstream method used in most reservoir simulators. Herein, we discuss two improved discretizations, a truly multi-dimensional method and a higher-order discontinuous Galerkin method, that both are applicable to a large family of general, polyhedral grids. We validate the methods, compare their accuracy, and investigates how the improved accuracy impacts flow-diagnostic measures and to what extent this is important for their use in various workflows.

The flow diagnostic tools outlined herein are available as free and open-source software, through the Matlab Reservoir Simulation Toolbox (http://www.sintef.no/mrst) in the *diagnostics* module or the OPM initiative (http://opm-project.org) in the *opm-core* module.

Introduction

Computational tools for reservoir modelling are used extensively throughout the hydrocarbon industry to validate alternative hypotheses about the reservoir, quantify uncertainty, or systematically explore different strategies for optimal recovery given a set of data, assumptions, and operating constraints. In particular, modern reservoir simulators can assist reservoir management by providing detailed forecasts of hydrocarbon recovery based on a description of the reservoir, the fluid dynamics, well controls, and couplings to surface facilities. However, this capability usually comes at a high computational costs, and in many geomodelling workflows such as production optimisation, there is not time to wait for full dynamic simulations, or there is no call for the complexity of a full simulation to produce the information necessary to make certain reservoir management decision.

The word 'flow diagnostics' as used herein, refers to a set of simple and controlled numerical flow experiments that are run to probe a reservoir model, establish connections and basic volume estimates, and quickly provide a qualitative picture of the flow patterns in the reservoir. The methods can also be used to compute quantitative information about the recovery process in settings somewhat simpler than what would be encountered in an actual field. As such, these methods offer a computationally inexpensive alternative to performing full-featured multiphase simulations in order to rank, compare, and validate reservoir models, upscaling procedures, and production scenarios. So far, flow diagnostics has primarily been associated with streamlines and have been applied e.g., in ranking and upscaling of geostatistical models (Idrobo et al., 2000; Ates et al., 2005), to optimise well rates in waterflooding (Thiele and Batycky, 2003; Park and Datta-Gupta, 2011; Izgec et al., 2011), for flood surveillance on a pattern-by-pattern basis (Batycky et al., 2008), and to optimise fracture stages and horizontal well completions in tight gas reservoirs (Sehbi et al., 2011). Whereas streamline tracing is highly efficient for standard reservoir models composed of hexahedral cells, their extensions to more complex grids are far less efficient. To also be applicable to fully unstructured grids with general polyhedral cells, multicontinuum models, and models containing lower-dimensional objects (discrete-fracture models), it would be an advantage if flow diagnostic tools could be formulated based on finite-volume discretizations that are compatible and straightforward to include in existing simulators.

Herein, we discuss an approach to flow diagnostics that uses standard finite-volume discretizations to compute time-of-flight and stationary tracer partitions (Natvig et al., 2006, 2007; Aarnes et al., 2007; Eikemo et al., 2009), from which one can easily find time lines in the reservoir, extract information along these lines, and partition the reservoir into volumes associated with injector/producer pairs as shown in Figure 1. Shahvali et al. (2012) showed how these basic grid quantities can be used to compute well-allocation factors and flow-capacity/storage-capacity diagrams, from which estimates of sweep efficiency and various heterogeneity measures like the Lorenz coefficient can be computed. In a recent paper, Møyner et al. (2014) also demonstrated how flow diagnostics can be combined with rigorous mathematical optimisation techniques to improve decisions like well location, drilling sequence, and time-varying pressure and rate targets for individual well completions. These production plans will not necessarily satisfy all operation constraints, but by feeding them to a more comprehensive simulator for validation and minor adjustments, the authors were able to demonstrate substantial improvements in simulated net-present value objectives (see the 'optimisation' box in Figure 1).

The current paper presents two new contributions to flow diagnostics. The first consists of two improved discretizations of the basic differential equations: (i) a generalisation of multidimensional upstream weighting suitable for 3D unstructured grids, and (ii) a discontinuous Galerkin method for time-of-flight suitable for 3D unstructured grids. We also discuss slope limiters suitable for solving the time-of-flight equation with higher-order methods. The second contribution is an investigation of the possible benefits of using these more accurate discretization methods for flow diagnostics purposes. However, before introducing these new discretization methods and investigating their effect, we will briefly outline the basic ideas of flow diagnostics in some more detail.

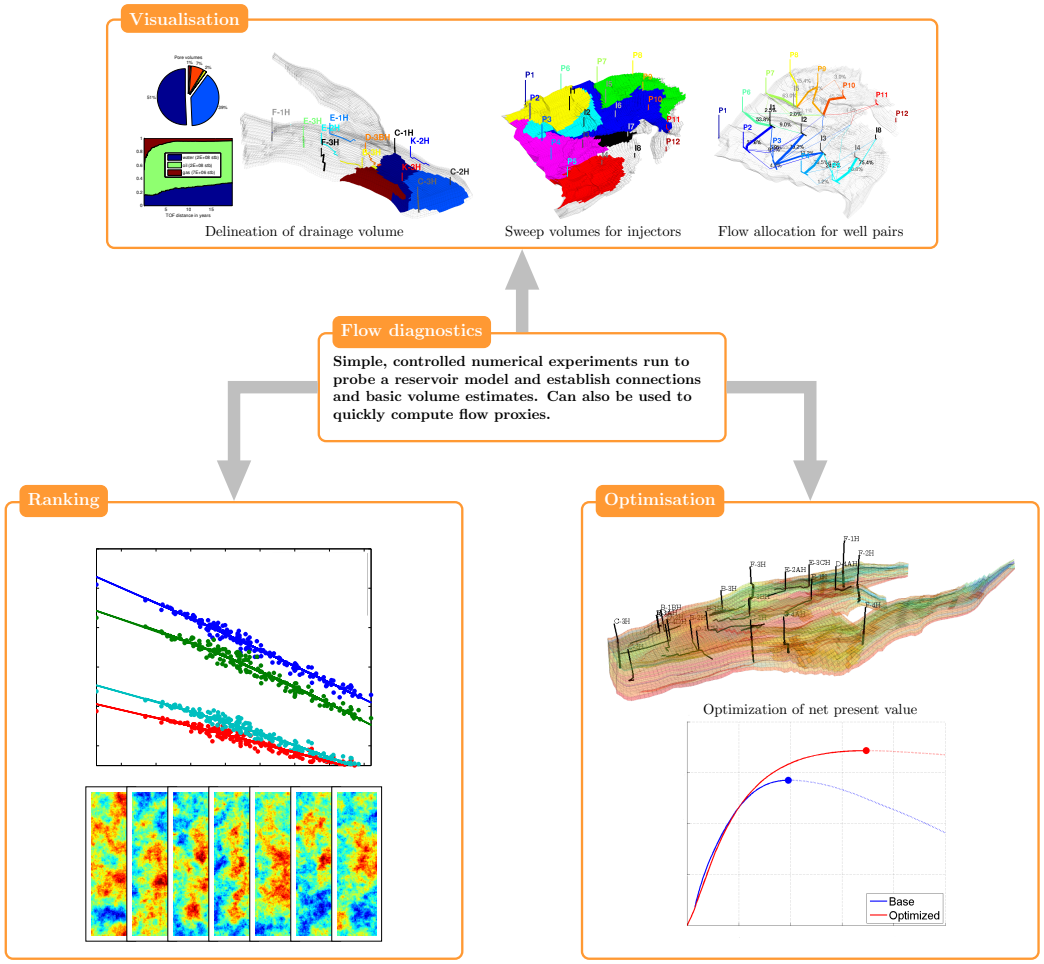

Figure 1 *Examples of flow diagnostics usage.* ***Visualisation:*** *To the left, a drainage volume associated with a producer from the Norne field is delineated into volumes associated with injector-producer pairs. The middle plot shows sweep volumes for an artificial well pattern on the Gullfaks field, whereas the right plot shows volumetric flow associated with each injector-producer pair.* ***Ranking:*** *the upper plot shows correlation between the amount of oil recovered and the Lorenz coefficient for a set of equiprobable permeability realisations and four different fluid models. The lower plots show seven different realisations.* ***Optimisation:*** *an approximate measure of net-present value derived from time-of-flight is used as objective function to optimise the injection strategy for the Norne field.*

Flow diagnostics

The basic flow diagnostics consists of two quantities:

- Time-of-flight denotes the time it takes an imaginary particle released at the inflow to reach a given point in the reservoir. Isocontours of time-of-flight from all injectors define natural time lines in the reservoir. The backward time-of-flight is defined completely analogously as the travel time from a given point to the nearest outflow point.
- Stationary tracer distributions from each injection and production well. For each injector, we freeze the velocity field and conduct a simple tracer test by setting the concentration to unity in that well and to zero in all other injectors. The stationary tracer distribution reveals the regions in the reservoir that are influenced by the injector. For producers, similar tracer tests are conducted by reversing the flux field. Tracer partitions can also easily be computed for individual well completions.

In the following, we will outline the simple differential equations that describe time-of-flight and tracer partitions and briefly discuss the various quantities that can be derived thereof.

Time-of-flight and tracer equations

Time-of-flight τ is a scalar function defined by the equation

$$\mathbf{v} \cdot \nabla \tau = \phi, \tag{1}$$

where $\mathbf{v}$ is the fluid velocity and ϕ the porosity of the medium. Herein, we assume that $\mathbf{v}(\mathbf{x})$ has been precomputed, e.g., by solving an incompressible flow equation with no-flow boundary conditions. In addition we need a boundary condition at the inflow. If all flow is well-to-well, we take $\tau = 0$ at injection wells. This yields the time-of-flight at any point of the domain for tracer particles starting at injectors. For some purposes, the reverse time-of-flight is required: time-of-flight from any point to a producer. This can be obtained by using $-\mathbf{v}$ in (1) and the boundary condition $\tau = 0$ at production wells.

In our case, we make the simplifying assumption of single-phase flow. The flow of a passive tracer component in that phase is then described by

$$\frac{\partial(\phi c \rho)}{\partial t} + \nabla \cdot (c \mathbf{v} \rho) = 0$$

where c is the tracer concentration, ρ is the fluid density and $\mathbf{v}$ is the fluid velocity. Assuming incompressible fluid and porous medium, ρ becomes a constant and ϕ independent of time. Also, $\nabla \cdot \mathbf{v} = 0$. Finally, since we look for a stationary solution, we may simplify the above to

$$\mathbf{v} \cdot \nabla c = 0. \tag{2}$$

Adding a boundary condition specifying $c = c_I$ at some injector I determines c to be constant and equal to c_I in the part of the domain that is reachable from I. Solving for multiple tracers associated with different injectors can yield a partitioning of the domain by flooding region. Like for time-of-flight, reversing the velocity field yields information about the producers, enabling us to compute a partitioning of the domain by drainage region. We note that the similarity of equations (1) and (2) means that they can often be solved simultaneously with little extra effort. Tracer partitions that delineate the reservoir volume into units associated with unique injectors or producers can be defined by thresholding the tracer distributions.

Derived quantities

From from the two basic quantities, time-of-flight and tracer partition, one can derive various quantities. From tracer partitions, one can easily obtain drainage volumes, sweep volumes, well-pair connections, and well allocation factors, i.e., the amount of flow in/out of a segment of a producer/injector that can be attributed to another segment on a injector/producer. Likewise, time-of-flight can be used to extend classical measures of heterogeneity so that they can be used to assess how heterogeneity affects the volumetric sweep efficiency of three-dimensional reservoir models (Shook and Mitchell, 2009). To define this quantity, we first introduce the total travel time $\hat{\tau} = \tau_f + \tau_b$, defined as the sum of the forward and backward time-of-flight. Then, we can define the generalisation of three classical quantities from volumetric sweep theory (Lake, 1989):

- The storage capacity $\Phi(\tau)$ denotes the cumulative pore volume of all cells that have a total travel time $\hat{\tau} \leq \tau$. The flow capacity $F(\tau)$ is the cumulative flux associated with all cells that have $\hat{\tau} \leq \tau$. For incompressible flow, the flux can be determined using the fact that the pore volume equals the product of the flux and the total travel time. In practice, the values are normalized so that both F and Φ are within the unit interval. If we interpret the displacement as a 1D process, the *flow and storage capacity diagram* $F(\Phi)$ would correspond to the 1D fractional flow curve, and hence reduces to a straight line for a homogeneous displacement (for which $\hat{\tau}$ is constant).

- The *sweep efficiency* is measured as a function of dimensionless time in pore volumes using $F(\Phi)$ as flux function. Dimensionless time is defined as $t_D = d\Phi/dF$ and the sweep efficiency is defined as $E_v = \Phi + (1 - F(\Phi))t_D$.
- The *Lorenz coefficient* L_c is defined as the area bounded above by the $F(\Phi)$ curve and below by the line $F = \Phi$, normalized by a factor 0.5, and measures the difference in flow capacity between a heterogeneous and a planar, piston-like displacement. The coefficient has values between zero for homogeneous displacement and unity for infinitely heterogeneous displacement. The Lorenz coefficient is a popular measure of heterogeneity, but can also be used as a simple flow proxy for computing recovery factors, e.g., as discussed in (Møyner et al., 2014).

We refer to (Shook and Mitchell, 2009) for a more in-depth investigation of the Lorenz coefficient in the context of streamlines and to (Shahvali et al., 2012) for the finite-volume interpretation. Several example uses are also illustrated in as shown in Figure 1.

Finite-volume discretizations

The standard choice for solving equation (1) is to use a streamline method. Streamline methods have several advantages: they are fast, yield sub-grid resolution for time-of-flight, are intuitive and easy to visualise. However, there are two major drawbacks. First, tracing streamlines is not trivial in grids with complex geometry. Methods have been developed for example for corner-point grids by Zhang et al. (2011) or for polyhedral grids by Klausen et al. (2012). Still, these methods are quite complex and hard to implement correctly. Second, streamline methods yield answers on the streamlines themselves, so if one needs data such as time-of-flight represented on the computational grid, a mapping procedure is required, which can lead to significant smearing of the data. Moreover, since streamlines cluster in high-flow regions, their resolution may be low in stagnant regions and near flow divides.

We instead choose to solve the time-of-flight and tracer equations directly on the given grid, using finite-volume and discontinuous Galerkin methods, which are fairly simple to implement and give results directly on the computational grid so that we avoid streamline tracing and mapping. This idea is not new, but has been described earlier (Natvig et al., 2007; Natvig and Lie, 2008; Eikemo et al., 2006) and was applied by Shahvali et al. (2012); Møyner et al. (2014) to compute flow diagnostics. As is common also for streamline methods, we will henceforth assume that the velocity field is given as a set of face fluxes, so that for any two adjacent cells C_i and C_j the quantity

$$v_{ij} = \int_{F_{ij}} \mathbf{v} \cdot \mathbf{n}_i \, dS$$

is known. Here, F_{ij} is the common face between the cells, and $\mathbf{n}_i$ is the outer unit normal of ∂C_i. With this definition, v_{ij} is skew-symmetric: $v_{ij} = -v_{ji}$.

Single-point upstream method

Finite-volume methods are often the method of choice for problems given on stratigraphic grids (corner-point, PEBI, etc), since they are robust and have few requirements with respect to grid properties. A finite-volume formulation of (1) can be obtained as follows. Consider a partition of Ω into cells. For each cell C_i, (1) holds. We integrate over C_i to get

$$\int_{C_i} \mathbf{v} \cdot \nabla \tau \, dx = \int_{C_i} \nabla \cdot (\tau \mathbf{v}) \, dx = \int_{\partial C_i} \tau (\mathbf{v} \cdot \mathbf{n}_i), dS = \int_{C_i} \phi(\mathbf{x}) \, dx,$$

where we have used the divergence theorem and the fact that $\nabla \cdot \mathbf{v} = 0$ for incompressible flow. Approximating τ by a single value τ_i for each cell, and taking τ at cell boundaries to be from the upwind cell of each face, we can write

$$\sum_{j \in U(i)} v_{ij} \tau_j + \tau_i \sum_{j \in D(i)} v_{ij} = \phi_i |C_i|. \tag{3}$$

The sets $U(i)$ and $D(i)$ are given by $U(i) = \{j | v_{ij} < 0\}$ and $D(i) = \{j | v_{ij} > 0\}$, and $|C_i|$ is the volume of C_i. For the stationary tracer equation (2), the finite-volume formulation is

$$\sum_{j \in U(i)} v_{ij} c_j + c_i \sum_{j \in D(i)} v_{ij} = 0. \tag{4}$$

These equations are quite simple, linear, and can easily be solved for large numbers of cells. In the following, we will apply a flow-based reordering technique (Natvig et al., 2007; Natvig and Lie, 2008) that will make the linear systems of (3) and (4) triangular (or block triangular), and hence quick to solve. It is worth noting that for a real-world implementation using floating-point arithmetic, special care may have to be taken in stagnant regions, where inaccuracies in the given face fluxes can result in cells with only inflow or only outflow.

Multidimensional upstream weighting

The single-point upwind choice (SPU) made for the values of τ at cell interfaces in (3) can cause grid-orientation effects. Therefore, many authors have suggested extending the stencil to include information from more neighbouring cells, often called multidimensional upstream (MDU) weighting. Keilegavlen et al. (2012) describe such a scheme suitable for multiphase flow on general 2D grids. We briefly describe the scheme, in the context of time-of-flight computations, and our extension to 3D.

In 2D, each upstream edge is divided in two half-edges, and an interaction region is formed by the set of half-edges adjacent to a vertex of the grid. For such a region, we let $\overline{\tau}_{ij,k}$ be the time-of-flight associated with a particular half-edge, where k indicates the associated vertex. The τ_j-value in (3) would then be replaced with the average of the two half-edge values for the edge ij. Let $v_{ij,k}$ be the flux across a half-edge, and assume that the flux from i to j is positive. Then we define:

$$\overline{\tau}_{ij,k} = (1-\omega)\tau_i + \omega \overline{\tau}_{ri,k}, \qquad \omega = L(\omega^*), \quad \omega^* = v_{ij,k}/v_{ri,k}.$$

In the interaction region around k, cell i is adjacent to two other cells, j and r, and $v_{ri,k}$ is the flux into cell i from r across their mutual half-edge. The function L is a limiter function, that for monotonicity of the scheme must satisfy

$$0 \le L(\omega^*) \le \min(1, \omega^*).$$

The choice $L(\omega^*) = 0$ corresponds to the SPU method. We have used the TMU limiter, given by

$$L(\omega^*) = \max(\min(\omega^*, 1), 0).$$

For more details, see (Keilegavlen et al., 2012).

In 3D, we use face-parts instead of half-edges. As with the half-edges, we associate each face-part with a vertex k, and indicate fluxes across them by $v_{ij,k}$. Now there can be multiple face-parts adjacent to a cell i and vertex k, that need not even share an edge with the ij face. We only want those face-parts with inflow into cell i to contribute to the (outflow) τ on face ij. We therefore define the set Q_{ijk} to help us indicate that set of face-parts:

$$Q_{ijk} = \{r : r \in N(i), r \neq j, k \in V(i,r), v_{ri,k} > 0\}.$$

Here, $N(i)$ are the indices of cells that share a common face with C_i, whereas the set $V(i,r)$ contains the vertices adjacent to face F_{ir}. Now, we can write

$$\overline{\tau}_{ij,k} = (1 - \sum_{q \in Q_{ijk}} \omega_q)\tau_i + \sum_{q \in Q_{ijk}} \omega_q \overline{\tau}_{qi,k}, \qquad \omega_q = \frac{v_{qi,k}}{\sum_{r \in Q_{ijk}} v_{ri,k}} L\left(\frac{\sum_{r \in Q_{ijk}} v_{ri,k}}{v_{ij,k}}\right).$$

This method reproduces the 2D scheme for extruded grids, and provides an appropriate MDU weighting for 3D.

Discontinuous Galerkin discretization

For higher-order accuracy, we have chosen to use the discontinuous Galerkin (DG) method. As for the finite-volume method, DG is well-suited for complex grids. We will show the method for the time-of-flight equation only, since the tracer equation is a simplification. This follows the approach used in (Natvig et al., 2007), we will however give some details pertaining to basis, quadrature, and limiters used in our specific implementation.

Weak form on cells

As a first step, we must write (1) in weak form, by multiplying the equation with an arbitrary smooth function ψ and integrating over some domain $K \subseteq \Omega$. After integration by parts, we have the following equation:

$$-\int_K \tau(\mathbf{v}\cdot\nabla\psi)\,dx + \int_{\partial K} \tau\psi(\mathbf{v}\cdot\mathbf{n})\,dS = \int_K \phi\,\psi\,dx, \tag{5}$$

that must hold for arbitrary K and ψ that are smooth on K. We now choose a polynomial degree d and a basis for polynomials of degree d on each cell C_i which we call $\{\psi_{i,k}\}$. We will then write τ in terms of these basis functions:

$$\tau|_{C_i} = \sum_{k=1}^{k_d} \tau_{i,k}\psi_{i,k} \equiv \tau_i,$$

where k_d is the number of basis functions, which only depends on the degree d and the number of spatial dimensions, which we assume to be three herein. Since (5) must hold for the domain C_i and each basis function for that cell, we have that

$$-\int_{C_i} \tau(\mathbf{v}\cdot\nabla\psi_{i,k})\,dx + \int_{\partial C_i} \tau\psi_{i,k}(\mathbf{v}\cdot\mathbf{n})\,dS = \int_{C_i} \phi\,\psi_{i,k}\,dx$$

must hold for all cells C_i and all basis functions $\psi_{i,k}$. The τ in the first term is evaluated in cell C_i, but the second term is ambiguous as written. Since the value of τ can be discontinuous on cell boundaries, we need to choose how to evaluate τ on ∂C_i. Using upwind evaluation is consistent with the finite volume method described earlier. That yields the following for cell C_i:

$$\sum_{j\in U(i)} \int_{F_{ij}} \tau_j\psi_{i,k}v_{ij}\,dS + \sum_{j\in D(i)} \int_{F_{ij}} \tau_i\psi_{i,k}v_{ij}\,dS - \int_{C_i} \tau_i(\mathbf{v}\cdot\nabla\psi_{i,k})\,dx = \int_{C_i} \phi\,\psi_{i,k}\,dx. \tag{6}$$

Choice of basis

Ideally, the basis functions would form an orthogonal set for each cell C_i, since that would give maximal sparsity for the linear systems. This can be hard to provide since we want to allow arbitrary cell shapes. However, we do exploit some orthogonality, by letting the single degree-zero basis function on C_i, denoted $\psi_{C_i,0}$, be the characteristic function on C_i, and making the degree-one basis functions, denoted $\psi_{C_i,1,k}$ for $k = \{0,1,2\}$, orthogonal to it:

$$\psi_{C_i,0}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x}\in C_i, \\ 0, & \mathbf{x}\notin C_i, \end{cases} \qquad \psi_{C_i,1,k}(\mathbf{x}) = \begin{cases} x_k - \bar{x}_k, & \mathbf{x}\in C_i, \\ 0, & \mathbf{x}\notin C_i, \end{cases}$$

where $\mathbf{x} = (x_0, x_1, x_2)$ and the centroid of C_i is $\bar{\mathbf{x}} = 1/|C_i| \int_{C_i} \mathbf{x}\,dx$.

The SPU finite-volume method described previously is equivalent to the DG method with a degree-zero basis. So far we have not investigated what basis function set to use for degrees higher than one, we should look for functions that are easily implemented yet have some orthogonality and locality. Note that we do not define our basis on any reference domain, but directly on the physical domain. This is a consequence of allowing arbitrary cell geometries, since it is not feasible to define reference domains for arbitrary cell shapes in 3D.

Quadrature

Evaluating the integrals in (6) requires numerical integration over cells and faces. For arbitrary cell shapes this is a nontrivial task, that is easiest done by subdividing complex geometries into elementary shapes. We handle the general case by subdividing each face (that is, each cell-cell intersection) into triangles, each triangle having two vertices being the endpoints of a boundary edge and the third vertex being the face centroid. Similarly, we subdivide cells into tetrahedra by adding the cell centroid as a vertex to each of the boundary face triangles. This is fairly robust, but quite expensive, resulting in a simple cube being subdivided into 24 tetrahedra. It may be well worth the effort to use specialised quadrature rules for simple, known cell shapes, such as using tensor-product Gauss-Legendre rules for hexahedra. We have not applied this optimisation to our software at this time.

The basic quadrature rules used for elementary shapes may often require faces to be planar and edges to be linear. Quadrature is therefore one of the areas that may be sensitive to curved cell geometries. For quadratures requiring degree-one precision (i.e., exact for degree-one polynomials), we use the simple rule:

$$\int_E f(\mathbf{x})\,dx \approx |E| f(\bar{\mathbf{x}}_E),$$

where E is an arbitrary entity (cell or face), $|E|$ is its measure (volume or area), and $\bar{\mathbf{x}}_E$ is its centroid. This rule is applicable to arbitrary shapes, as long as we are able to compute the centroid.

For quadratures requiring degree-two precision, we split the domains into simplices. For triangles we apply the following edge-midpoint rule (given in barycentric coordinates):

$$\int_T f(\mathbf{x})\,dx \approx \frac{|T|}{3}\left(f(0,\tfrac{1}{2},\tfrac{1}{2}) + f(\tfrac{1}{2},0,\tfrac{1}{2}) + f(\tfrac{1}{2},\tfrac{1}{2},0)\right),$$

and for tetrahedra the following symmetric rule:

$$\int_T f(\mathbf{x})\,dx \approx \frac{|T|}{4}\left(f(b,a,a,a) + f(a,b,a,a) + f(a,a,b,a) + f(a,a,a,b)\right)$$

where $a = 0.138196601\ldots$ and $b = 1-3a$. These rules and more may be found in (Cools, 2003).

An important question is, what degree of precision do we need in the various integrations to ensure that the overall error has the expected order? Although we currently do not have strict theoretical bounds for this, our numerical experiments indicate that degree-two precision for cells seems to only be necessary for source and sink terms. This reduces significantly the amount of computations needed, as only a single point evaluation is required for degree-one precision.

Velocity interpolation

In general, we assume that the velocity field is only given as a set of discrete normal fluxes for each face in the grid. For DG1 and higher order, we need to integrate an expression that includes the velocity over each cell, thus requiring values for the velocity at quadrature points.

We have implemented two different velocity-interpolation schemes. The most basic scheme gives a constant approximation to $\mathbf{v}|C_i$. If we let v_{ij} be the normal flux of the face between cells C_i and C_j, as before oriented to have a positive sign if the flow is from C_i to C_j, the constant velocity for cell C_i is given by

$$\mathbf{v}_{C_i} = \frac{1}{|C_i|}\sum_{j\in N(i)} v_{ij}(\bar{\mathbf{x}}_{ij} - \bar{\mathbf{x}}_i),$$

in which $\bar{\mathbf{x}}_{ij}$ and $\bar{\mathbf{x}}_i$ are face and cell centroids. This formula is motivated from the mimetic finite difference method, and reproduces constant velocities.

The second scheme is the extended corner velocity interpolation scheme (ECVI). The basic CVI scheme was introduced for quadrangular cells by Hægland et al. (2007), and subsequently extended by Klausen et al. (2012) to any convex 2D cell shape and a large class of convex 3D cell shapes. It is based on two stages. In the first stage, we estimate velocity vectors for each corner of every cell independently, i.e., if a corner is shared by multiple cells, it will have an associated velocity vector for each of those cells. In the second stage, we use generalised barycentric interpolation to interpolate the corner velocities to arbitrary points.

The corner velocity $\mathbf{v}_{i,k}$ for a cell C_i and a corner k is chosen so that

$$\mathbf{v}_{i,k} \cdot \mathbf{n}_{ij} = v_{ij}$$

holds for all face fluxes v_{ij} whose associated face is adjacent to the corner k. For this to uniquely determine $\mathbf{v}_{i,k}$, there must be exactly three such adjacent faces to each corner (two in 2D), and their normals must be linearly independent. This is satisfied by many cell shapes, such as tetrahedra, hexahedra, prisms, extruded polygons, and (under a certain uniqueness assumption) Voronoi diagrams. Pyramids are an example of an unacceptable shape (four faces adjacent to the top corner). Another failing example would be a Cartesian cell with a subdivided face, since the subdivided faces would have the same normal and thus not be linearly independent.

Given corner velocities, we interpolate in a cell C_i by

$$\mathbf{v}(\mathbf{x}) = \sum_{k \in K(i)} \lambda_k(\mathbf{x}) \mathbf{v}_{i,k},$$

where $K(i)$ is the set of corners of C_i and the functions λ_k are generalised barycentric coordinates on C_i.

The particular family of barycentric coordinates we have implemented is an extension of the Wachspress coordinates to 3D. They are computed by the following formula, which is a modified form of Eqs. (4) and (5) in (Meyer et al., 2002):

$$\lambda_k(\mathbf{x}) = \frac{w_k(\mathbf{x})}{\sum_k w_k(\mathbf{x})}, \qquad w_k(\mathbf{x}) = V_k \prod_{j \in X(i,k)} \Big(\mathbf{n}_{ij} \cdot (\bar{\mathbf{x}}_{ij} - \mathbf{x}) \Big), \qquad V_k = |\det\{\mathbf{n}_{ij}\}_{j \in A(i,k)}|.$$

In the above, $A(i,k)$ and $X(i,k)$ are sets of neighbour cells (of C_i) for which the common face with C_i is adjacent to the corner k (for $A(i,k)$) or non-adjacent (for $X(i,k)$), respectively. Moreover, $\bar{\mathbf{x}}_{ij}$ is the centroid of face F_{ij}, and $\mathbf{n}_{ij}$ is its outward-pointing unit normal.

Slope limiters

Higher-order methods often exhibit unwanted behaviour in the form of oscillations, and that is also the case for our DG method with degree-one basis. Also, unfeasible values can appear in the solutions for time-of-flight and tracers. To handle this, we must apply a slope limiting procedure.

We have created two different slope limiting methods that are suitable for the DG(1) method on unstructured grids. The limiters enforce the following two properties: that $\tau \geq 0$, and that $\tau|C_i \geq m_i$ for some m_i depending on upstream values of τ. The difference between the two methods is in the choice of m_i. We first describe the method that is our current choice for the time-of-flight equation. This method consists of two steps: In the first step, we correct the mean value of τ:

$$\tau(\bar{\mathbf{x}}_i) = \max\big(m_i, \tau(\bar{\mathbf{x}}_i)\big), \qquad m_i = \max\big(\min_{j \in U(i)} \tau(\bar{\mathbf{x}}_j), 0\big).$$

Then, in the second step, we limit the slope of the linear polynomial. That is, if M_i is the minimum of $\tau(\mathbf{x})$ on C_i and $M_i < m_i$, we replace τ on C_i with $\hat{\tau}$ given by

$$\hat{\tau}_i(\mathbf{x}) = \tau(\bar{\mathbf{x}}_i) + (\tau(\mathbf{x}) - \tau(\bar{\mathbf{x}}_i))L_i, \qquad L_i = \frac{\tau(\bar{\mathbf{x}}_i) - m_i}{\tau(\bar{\mathbf{x}}_i) - M_i}.$$

We note that M_i can be found by evaluating τ at all corners of C_i, since we assume that cells have planar faces. In our preferred basis, the transformation from τ to $\hat{\tau}$ can be easily accomplished by multiplying the degree-one coefficients (that is, all but the first coefficient) with L_i.

In the second variant, we use upstream face values instead of upstream cell values to define m_i; that is, we take m_i to be the minimum of $\tau_j(\mathbf{x})$ with $\mathbf{x} \in F_{ij}$ for all $j \in U(i)$, bounded below by zero. These minima can again be found by evaluating τ in the corners adjacent to the upstream faces using the value on the *upstream* side, i.e., τ defined in cell j as signified by the notation $\tau_j(\mathbf{x})$. Numerical experiments indicate that this limiter does not preserve second order for smooth solutions (see Figure 3), and we have therefore used the first variant in all cases, unless explicitly noted.

An issue that is orthogonal to the choice of limiter is when to apply it. We have investigated three distinct approaches:

1. Apply limiter as a post-process, replacing τ with $\hat{\tau}$ simultaneously.
2. Apply limiter as a post-process, replacing τ with $\hat{\tau}$ cell-by-cell in flow order.
3. Replace τ with $\hat{\tau}$ cell-by-cell in flow order, as we solve the equations (in the same order).

The best choice seems to be the third approach in terms of preserving accuracy and avoiding oscillations.

Numerical experiments

In this section, we will verify and validate the three different discretization methods outlined above. For brevity, we will refer to them as follows: the single-point, finite-volume method is denoted FV(SPU); the multidimensional, upstream weighting method is denoted FV(MDU); and the discontinuous Galerkin method with degree-one polynomials is referred to as DG(1). The basic flow diagnostic tools are available as open-source software, implemented as a separate module of the Matlab Reservoir Simulation Toolbox (Lie et al., 2012a; MRST), which also contains a graphical user interface that can be used for enhanced, interactive visualisation and pre- and post-processing of high-resolution simulation models.

Case 1. Numerical verification of method order

To verify our implementations, we consider a 2D rotational velocity field $\mathbf{v}(\mathbf{x}) = (y, -x)$, for which the analytical solution of τ is known (Eikemo et al., 2009). If we choose our computational domain to be $[1,2] \times [1,2]$, the time-of-flight is given by

$$\tau(\mathbf{x}) = -\arctan\left(\frac{y}{x}\right) + \arctan\left(\frac{\min(\sqrt{x^2+y^2-1}, 2)}{\max(\sqrt{\max(x^2+y^2-4, 0)}, 1)}\right)$$

The fluxes for each face of the computational grids are approximated using the velocity at the face centroids: $v_{ij} = (\mathbf{v}(\overline{\mathbf{x}}_{ij}) \cdot \mathbf{n}_{ij})|F_{ij}|$.

The computed τ fields for a coarse and a fine grid are shown in Figure 2, for the three methods we consider. The grids used are twisted Cartesian grids, the twisting effect can be most clearly seen in the coarse-grid solutions. As expected, the DG(1) method produces the most accurate solution with the sharpest reproduction of the kink, followed by FV(MDU) and finally FV(SPU).

Table 1 reports the results of a grid-refinement study for all three methods, in which we have computed errors by evaluating the numerical solutions in all cell centroids $\overline{\mathbf{x}}_i$ and comparing to the value of the true solution in the same points for a series of $N \times N$ grids. While the methods do not achieve the maximum order of convergence due to the kink in the solution, nonetheless we see that the more advanced methods

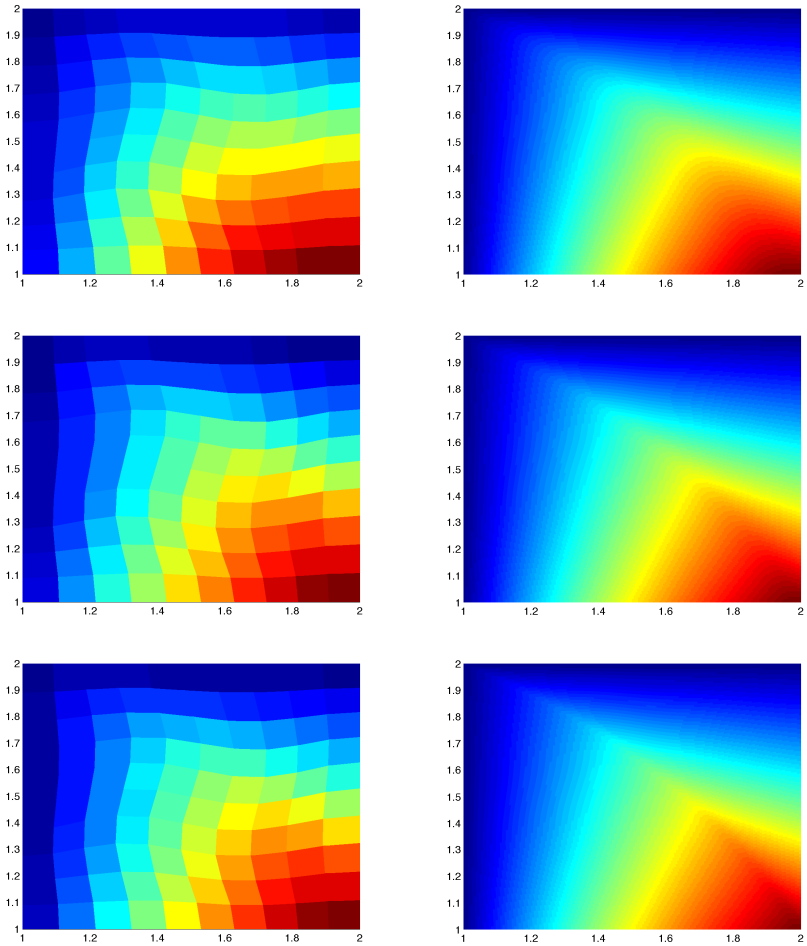

Figure 2 *TOF for rotational velocity field. The left column contains results for* 10×10 *grids and the right column for* 160×160 *grids. The top row shows FV(SPU) results, the middle row FV(MDU) results and the bottom row DG(1) results.*

Table 1 *L_2 errors and convergence rates for full domain.*

N	FV(SPU) error	rate	FV(MDU) error	rate	DG(1) error	rate
10	3.1e-02		3.1e-02		6.8e-03	
20	2.1e-02	0.59	1.6e-02	0.97	3.1e-03	1.16
40	1.3e-02	0.64	8.3e-03	0.95	1.4e-03	1.14
80	8.2e-03	0.67	4.3e-03	0.93	6.3e-04	1.13
160	5.1e-03	0.69	2.3e-03	0.91	2.9e-04	1.13

FV(MDU) and DG(1) give a significant improvement in the convergence behaviour. In Table 2 we have given the same error estimates, but this time only for a smooth subdomain, $[1, 1.3] \times [1, 1.3]$, on which the methods seem to approach the expected order-one convergence for the FV methods and order-two convergence for the DG(1) method. Plots of L_2 errors versus N can be found in Figure 3. As noted in the section on slope limiters, the alternative limiter is not able to preserve second order, and will not be further investigated.

Case 2. Variation across SPE10 layers.

For each of the 85 layers of the SPE10 benchmark model (Christie and Blunt, 2001), we have computed a single-phase pressure solution for a standard five-spot well configuration, i.e., a single injector in the

Table 2 *L_2 errors and convergence rates for smooth subdomain.*

N	FV(SPU) error	rate	FV(MDU) error	rate	DG(1) error	rate
10	1.1e-02		1.3e-02		1.0e-04	
20	5.5e-03	0.97	6.5e-03	1.00	2.9e-05	1.78
40	2.8e-03	1.00	3.2e-03	1.00	8.0e-06	1.87
80	1.4e-03	1.00	1.6e-03	1.00	2.1e-06	1.93
160	6.9e-04	1.00	8.1e-04	1.00	5.4e-07	1.96

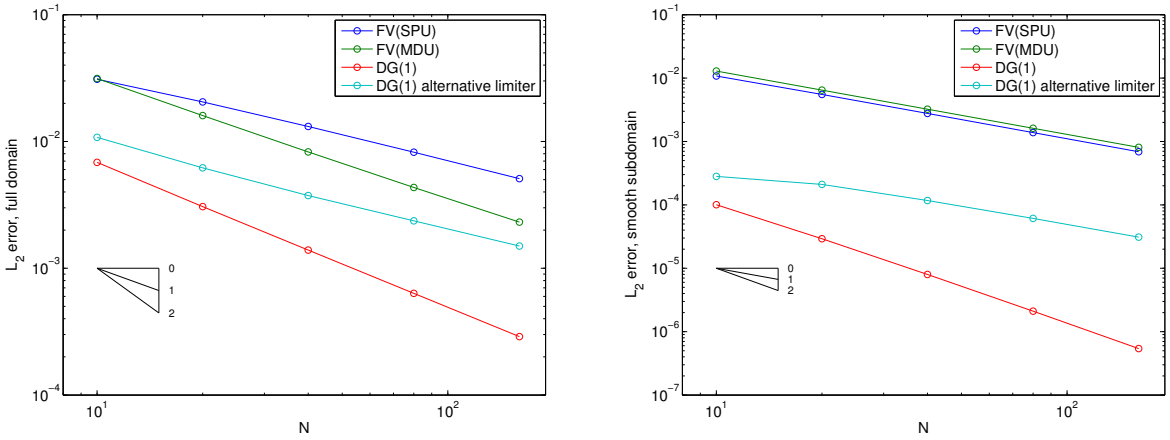

Figure 3 *L_2 errors for full domain (left) and smooth subdomain (right).*

middle and a producer in each corner. From the pressure solution we have obtained face fluxes, that were used as input for the time-of-flight and tracer computations. For each layer, we have then computed F-ϕ curves, Lorenz coefficients, and sweep efficiencies. Figure 4 shows the computed Lorenz coefficients for all layers, for all three methods on the left, and the increase given by using FV(MDU) or DG(1) compared to FV(SPU) on the right. Similarly, Figure 5 shows the estimated sweep efficiencies after one pore volume has injected in all layers. While the differences are not dramatic for most layers, the more accurate methods universally give higher Lorenz coefficients and lower sweep efficiency than the FV(SPU) method. This is not surprising, since both the FV(MDU) and the DG(1) method are designed to have lower numerical dissipation than FV(SPU) and hence should introduce less smearing of local heterogeneity effects than the FV(SPU) scheme. For some layers, the differences are quite large, and the

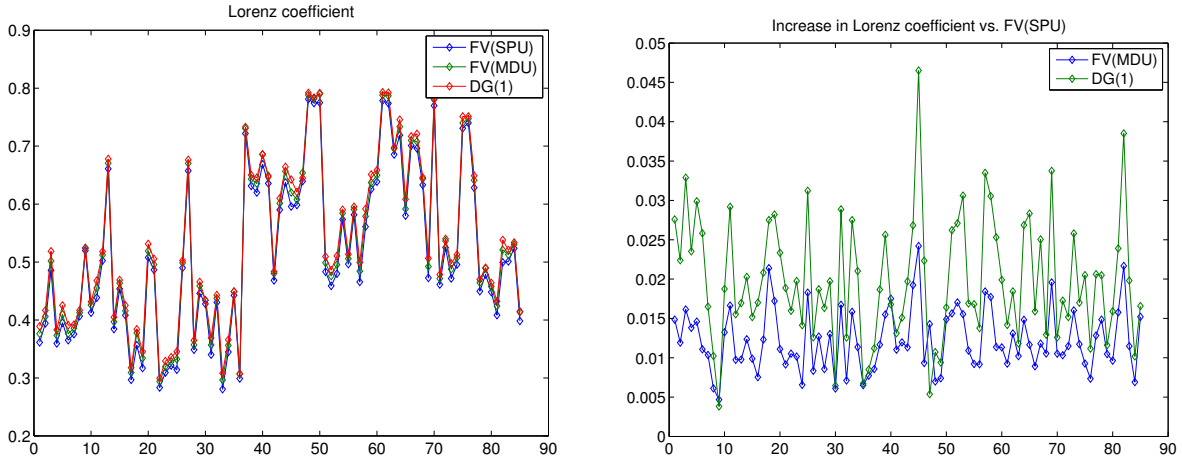

Figure 4 *Lorenz coefficients (left) and increase in Lorenz coefficient relative to the FV(SPU) method (right) for all layers of SPE10.*

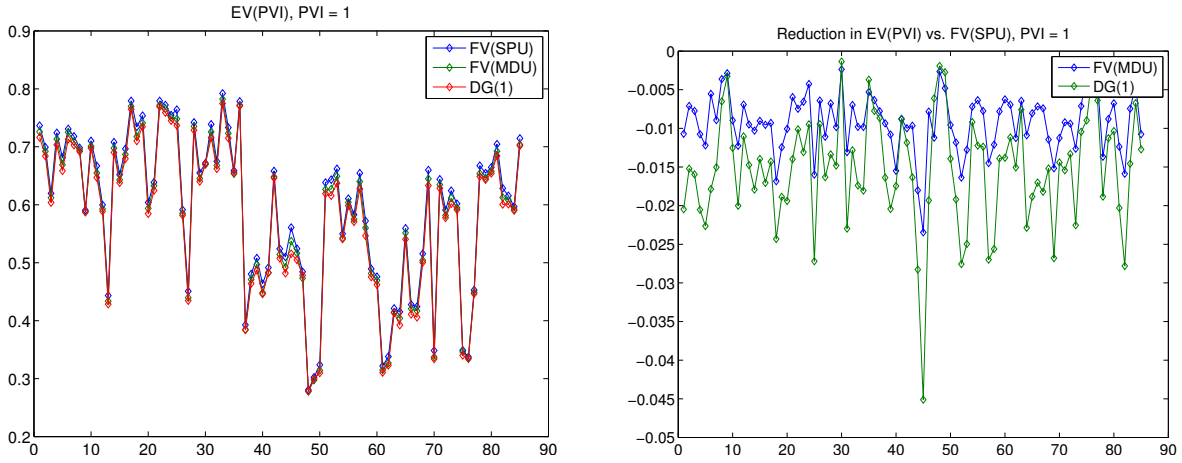

Figure 5 *Sweep efficiency (left) and decrease in sweep efficiency relative to the FV(SPU) method (right) for all layers of SPE10.*

Table 3 *Correlation coefficients between Lorenz coefficient and recovery factor.*

Coefficient	Fluid	FV(SPU)	FV(MDU)	DG(1)
Pearson	equal viscosity	-0.983711	-0.984116	-0.981108
Pearson	5:1 contrast	-0.974777	-0.976211	-0.973356
Spearman	equal viscosity	-0.982763	-0.983838	-0.982763
Spearman	5:1 contrast	-0.972367	-0.976021	-0.973285

more accurate methods should be preferred.

Case 3. Correlation with simulated recovery.

In this example, which is an extension of a numerical experiment shown in (Møyner et al., 2014), we have computed oil recovery fractions for a five-spot water-flooding scenario using a two-phase simulation for all layers of the SPE10 benchmark case. This has been done by simulating the injection of one pore volume of water over 100 time steps. For each layer, the flux field from the first time step has been used to compute time-of-flight values and the corresponding Lorenz coefficient. We have then correlated the Lorenz coefficients and the simulated recovery fractions. Finally, the experiment has been conducted twice using different fluid properties: with linear relative permeability and no viscosity contrast, and with quadratic relative permeability and a 5:1 viscosity contrast (oil-water).

The time-of-flight fields have been computed with all three discretization methods, and Figure 6 shows the results, with best-fit linear regression models. The correlation is in all cases strong enough to justify the use of the Lorenz coefficient as a metric for optimisation of well placement and rates. The question we ask here is: what is the effect of using the more accurate FV(MDU) or DG(1) methods. From Case 2, we would expect the more accurate methods to produce higher Lorenz coefficients given the same flux fields, and the plots confirm this: the linear best fit is found at higher Lorenz values for FV(MDU) and even higher for DG(1), for both fluid property models tested.

In Table 3 we find Pearson coefficients indicating linear correspondence and Spearman coefficients indicating rank correspondence. We observe that both sets of coefficients seem to be little changed by changing discretization method.

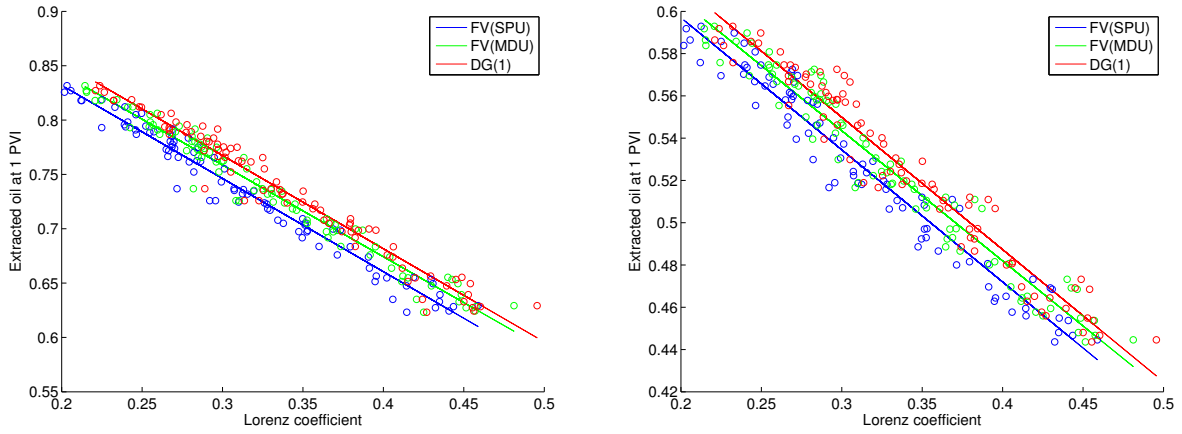

Figure 6 *Correlation between Lorenz coefficients and two-phase recovery for linear relative permeability and no viscosity contrast (left) and for quadratic relative permeability and a 5:1 viscosity contrast (right).*

Case 4. The Norne field model.

The Norne benchmark case (IO Center, NTNU, 2012) is a real field case that has been made available by Statoil and license partners. It contains data from a Norwegian Sea oil field, and we have used the grid and well placements from the model. The reservoir is faulted, and the corner-point grid is fairly complex. It has approximately 45 000 cells.

We have computed a single-phase pressure solution using the mimetic finite difference method (Lie et al., 2012b), and then solved the reverse tracer equations to determine well drainage regions, as shown in Figure 7. The sum of tracers is normalised to one, and gray color is used to show regions in which the maximum tracer value is less than one, i.e., regions near flow-divided where multiple tracers overlap. We can see that the gray areas are smaller for FV(MDU) and DG(1), which means that the boundaries between drainage regions are better defined than for the FV(SPU) method.

Case 5. A Voronoi-cell (PEBI) grid.

Voronoi-cell or PEBI grids are sometimes used for reservoir simulation. In such grids, cells are general convex polygons or polyhedra. As discussed earlier, streamline tracing on such geometries is challenging, and the FV and DG methods are a good alternative. We have used a simple quarter-five-spot setup on a 2D Voronoi grid created from random Voronoi sites (and some additional sites on the boundary). The resulting grid has a large variation in cell shapes and sizes, and is not really representative of grids that would be used in a real case, which would use a smoother distribution of Voronoi sites to get more smoothly varying cell sizes, or an iterative procedure to improve grid quality, such as used in (Persson and Strang, 2004). To generate a heterogenous case, we have simply applied the permeability and porosity fields from layer 45 of the SPE10 benchmark to the cells of our grid. We make no claim that this represent a geostatistically valid case. The flux field has been computed with a TPFA approximation, which is valid for Voronoi grids. The resulting forward time-of-flight and total travel times can be seen in Figure 8.

Conclusions

We have shown several methods for computing flow diagnostics such as time-of-flight and tracers including extension of a multidimensional upstream-weighting, finite-volume method and a discontinuous Galerkin method to arbitrary unstructured grids in 3D. We have then investigated the effect of applying the different discretizations to some common flow diagnostic scenarios. The more accurate schemes do

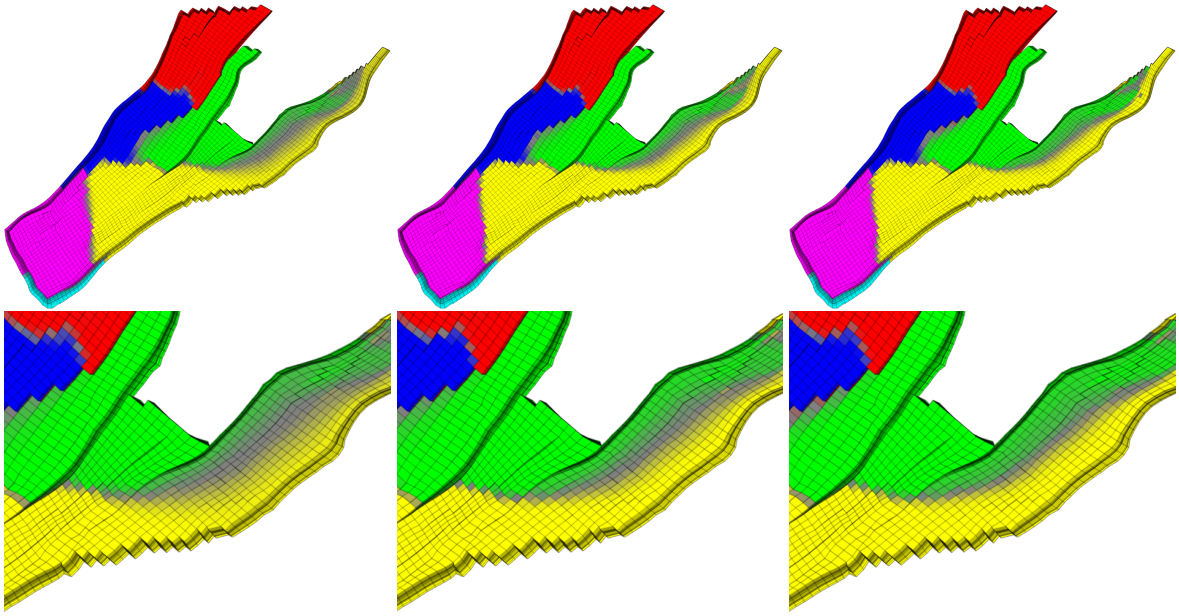

Figure 7 *Drainage regions computed with FV(SPU) method (left), FV(MDU) method (middle) and DG(1) method (right). Whole grid on top, zoomed in on stagnant region on bottom.*

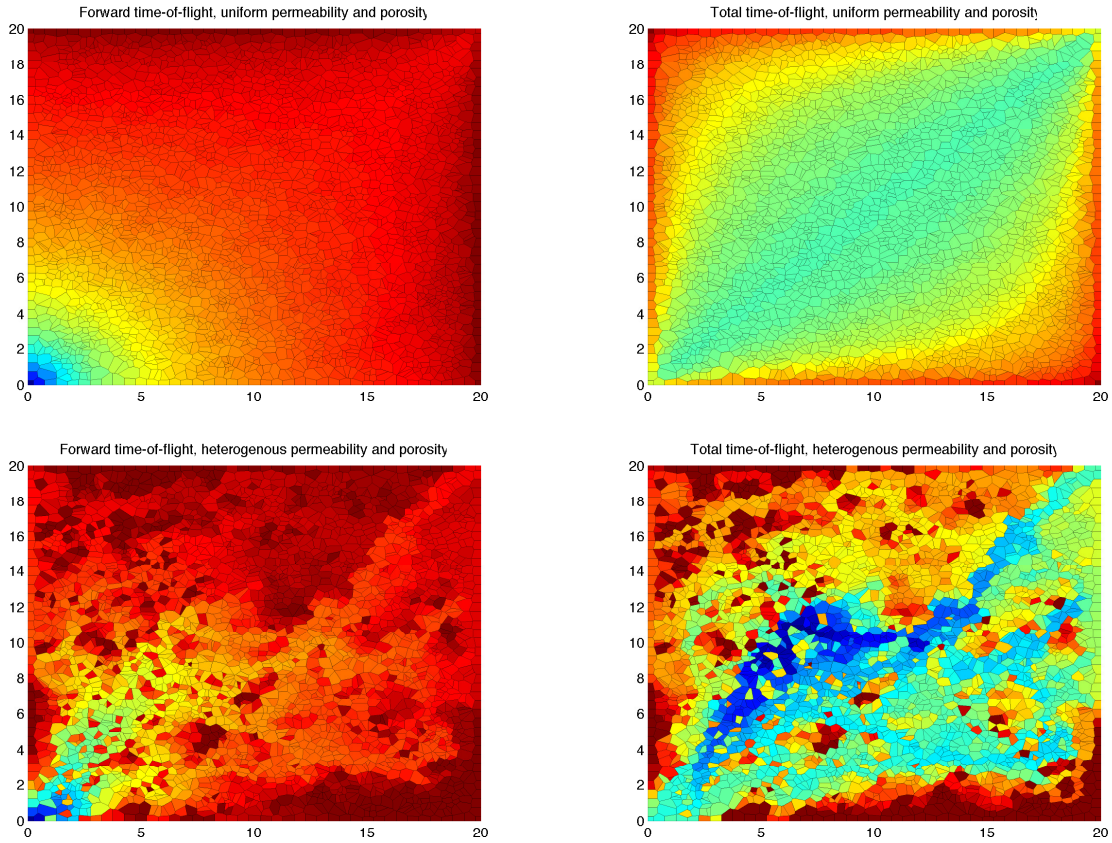

Figure 8 *Logarithm of forward time-of-flight (left) and total travel time (right) for uniform (top) and heterogenous (bottom) permeability field with Voronoi grid using DG(1) method.*

have an effect on the computed Lorenz coefficients and identification of flow divides, as shown in Cases 2 and 4, and can therefore be worth the extra investment in implementation effort, in particular since the additional computational effort would usually be small compared to the cost of computing the initial pressure solution used as basis for all flow-diagnostic calculations.

The effect on rankings seems to be low, however. As indicated by the results from Case 3, the correlation coefficients do not markedly improve, therefore the lowest-order FV(SPU) scheme seems to be sufficient for ranking and optimisation purposes. We caution that it should be investigated if this also holds for more complicated cases, for example with more wells, compressible flow, or three-phase effects.

The methods used in this paper are available as free software. The most recent version of the MATLAB Reservoir Simulation Toolbox (MRST) contains time-of-flight computations with the FV(SPU) method, and a *diagnostics* module with multiple utility functions and GUI utilities for flow diagnostics. The *opm-core* module from the Open Porous Media framework (OPM) contains classes and programs that compute tracer and time-of-flight with the FV(SPU), FV(MDU) and DG(1) methods.

Acknowledgements

The research is funded in part by Chevron ETC and the Research Council of Norway under grant no. 215665. The authors thank Statoil (operator of the Norne field) and its license partners ENI and Petoro for the release of the Norne data. Further, the authors acknowledge the IO Center for cooperation and coordination of the Norne cases.

References

Aarnes, J.E., Krogstad, S. and Lie, K.A. [2007] Application of multiscale modelling concepts on the reservoir model for the Norne field. Tech. Rep. F4102, SINTEF ICT.

Ates, H., Bahar, A., El-Abd, S., Charfeddine, M., Kelkar, M. and Datta-Gupta, A. [2005] Ranking and upscaling of geostatistical reservoir models using streamline simulation: A field case study. *SPE Res. Eval. Eng.*, **8**(1), 22–32, doi:10.2118/81497-PA.

Batycky, R.P., Thieles, M.R., Baker, R.O. and Chugh, S.H. [2008] Revisiting reservoir flood-surveillance methods using streamlines. *SPE Res. Eval. Eng.*, **11**(2), 387–394, doi:10.2118/95402-PA.

Christie, M.A. and Blunt, M.J. [2001] Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Eval. Eng.*, **4**, 308–317, doi:10.2118/72469-PA, url: `http://www.spe.org/csp/`.

Cools, R. [2003] An encyclopedia of cubature formulas. *J. Compl.*, **19**, 445–453.

Eikemo, B., Berre, I., Dahle, H.K., Lie, K.A. and Natvig, J.R. [2006] A discontinuous Galerkin method for computing time-of-flight in discrete-fracture models. *Proceedings of the XVI International Conference on Computational Methods in Water Resources*, Copenhagen, Denmark.

Eikemo, B., Lie, K.A., Dahle, H.K. and Eigestad, G.T. [2009] Discontinuous Galerkin methods for transport in advective transport in single-continuum models of fractured media. *Adv. Water Resour.*, **32**(4), 493–506, doi: 10.1016/j.advwatres.2008.12.010.

Hægland, H., Dahle, H.K., Eigestad, G.T., Lie, K.A. and Aavatsmark, I. [2007] Improved streamlines and time-of-flight for streamline simulation on irregular grids. *Adv. Water Resour.*, **30**(4), 1027–1045, doi: 10.1016/j.advwatres.2006.09.00.

Idrobo, E., Choudhary, M. and Datta-Gupta, A. [2000] Swept volume calculations and ranking of geostatistical reservoir models using streamline simulation. *SPE/AAPG Western Regional Meeting*, Long Beach, California, USA, sPE 62557.

IO Center, NTNU [2012] The Norne benchmark case. url: `http://www.ipt.ntnu.no/~norne/wiki/doku.php`.

Izgec, O., Sayarpour, M. and Shook, G.M. [2011] Maximizing volumetric sweep efficiency in waterfloods with hydrocarbon f-ϕ curves. *Journal of Petroleum Science and Engineering*, **78**(1), 54–64, doi: 10.1016/j.petrol.2011.05.003.

Keilegavlen, E., Kozdon, J.E. and Mallison, B.T. [2012] Multidimensional upstream weighting for multiphase transport on general grids. *Comput. Geosci.*, **16**, 1021–1042, doi:10.1007/s10596-012-9301-7.

Klausen, R.A., Rasmussen, A.F. and Stephansen, A. [2012] Velocity interpolation and streamline tracing on irregular geometries. *Comput. Geosci.*, **16**, 261–276, doi:10.1007/s10596-011-9256-0.

Lake, L.W. [1989] *Enhanced Oil Recovery*. Prentice-Hall.

Lie, K.A., Krogstad, S., Ligaarden, I.S., Natvig, J.R., Nilsen, H.M. and Skaflestad, B. [2012a] Open source MATLAB implementation of consistent discretisations on complex grids. *Comput. Geosci.*, **16**, 297–322, ISSN 1420-0597, doi:10.1007/s10596-011-9244-4.

Lie, K.A., Krogstad, S., Ligaarden, I.S., Natvig, J.R., Nilsen, H. and Skaflestad, B. [2012b] Open-source MATLAB implementation of consistent discretisations on complex grids. *Comput. Geosci.*, **16**, 297–322, doi: 10.1007/s10596-011-9244-4.

Meyer, M., Barr, A., Lee, H. and Desbrun, M. [2002] Generalized barycentric coordinates on irregular polygons. *J. Graphics Tools*, **7**(1), 13–22.

Møyner, O., Krogstad, S. and Lie, K.A. [2014] The application of flow diagnostics for reservoir management. *SPE J.*, accepted.

MRST [2014] The MATLAB Reservoir Simulation Toolbox, version 2014a. `http://www.sintef.no/MRST/`.

Natvig, J.R., Lie, K.A. and Eikemo, B. [2006] Fast solvers for flow in porous media based on discontinuous Galerkin methods and optimal reordering. *Proceedings of the XVI International Conference on Computational Methods in Water Resources*, Copenhagen, Denmark.

Natvig, J.R. and Lie, K.A. [2008] Fast computation of multiphase flow in porous media by implicit discontinuous Galerkin schemes with optimal ordering of elements. *J. Comput. Phys.*, **227**(24), 10108–10124, doi: 10.1016/j.jcp.2008.08.024.

Natvig, J.R., Lie, K.A., Eikemo, B. and Berre, I. [2007] An efficient discontinuous Galerkin method for advective transport in porous media. *Adv. Water Resour.*, **30**(12), 2424–2438, doi:10.1016/j.advwatres.2007.05.015.

OPM [2013] The Open Porous Media initiative. `http://wwww.opm-project.org`.

Park, H.Y. and Datta-Gupta, A. [2011] Reservoir management using streamline-based flood efficiency maps and application to rate optimization. *Proceedings of the SPE Western North American Region Meeting*, 7-11 May 2011, Anchorage, Alaska, USA, doi:10.2118/144580-MS.

Persson, P.O. and Strang, G. [2004] A simple mesh generator in matlab. *SIAM Review*, **46**(2), 329–345.

Sehbi, B.S., Kang, S., Datta-Gupta, A. and Lee, W.J. [2011] Optimizing fracture stages and completions in horizontal wells in tight gas reservoirs using drainage volume calculations. *Proceedings of the North American Unconventional Gas Conference and Exhibition*, 14-16 June 2011, The Woodlands, Texas, USA, doi: 10.2118/144365-MS.

Shahvali, M., Mallison, B., Wei, K. and Gross, H. [2012] An alternative to streamlines for flow diagnostics on structured and unstructured grids. *SPE J.*, **17**(3), 768–778, doi:10.2118/146446-PA.

Shook, G. and Mitchell, K. [2009] A robust measure of heterogeneity for ranking earth models: The F-Phi curve and dynamic Lorenz coefficient. *SPE Annual Technical Conference and Exhibition*.

Thiele, M.R. and Batycky, R.P. [2003] Water injection optimization using a streamline-based workflow. *Proceedings of the SPE Annual Technical Conference and Exhibition*, 5-8 October 2003, Denver, Colorado, doi: 10.2118/84080-MS.

Zhang, Y., King, M. and Datta-Gupta, A. [2011] Robust streamline tracing using intercell fluxes in locally refined and unstructured grids. *SPE Reservoir Simulation Symposium, 21-23 February 2011, The Woodlands, Texas, USA*, SPE 140695.