FAST COMPUTATION OF MULTIPHASE FLOW IN POROUS MEDIA BY IMPLICIT DISCONTINUOUS GALERKIN SCHEMES WITH OPTIMAL ORDERING OF ELEMENTS

JOSTEIN R. NATVIG AND KNUT–ANDREAS LIE

Abstract. We present a family of implicit discontinuous Galerkin schemes for purely advective multiphase flow in porous media in the absence of gravity and capillary forces. To advance the solution one time step, one must solve a discrete system of nonlinear equations. By reordering the grid cells, the nonlinear system can be shown to have a lower triangular block structure, where each block corresponds to the degrees-of-freedom in a single or a small number of cells. To reorder the system, we view the grid cells and the fluxes over cell interfaces as vertices and edges in a directed graph and use a standard topological sorting algorithm. Then the global system can be computed by processing the blocks sequentially using a standard Newton–Raphson algorithm for the degrees-of-freedom in each block. Decoupling the system offers greater control over the nonlinear solution procedure and reduces the computational costs, memory requirements, and complexity of the scheme significantly. In particular, the first-order version of the method may be at least as efficient as modern streamline methods when accuracy requirements or the dynamics of the flow allow for large implicit time steps.

1. Introduction

Simulation of flow in porous rock formations is used as a basis for decision-making in many subsurface disciplines, from petroleum production to groundwater protection. Driven by a need for faster and more accurate prediction of fluid behaviour, there is a prevailing requirement for fast flow simulation. This is particularly evident within the petroleum industry, where the current trend is towards using very large and complex models, often consisting of several million grid cells, to describe the underlying subsurface geology. Direct simulation of flow on models with several million cells is a challenging task.

Given the different physical mechanisms behind flow in porous media, the flow simulation can typically be divide in two parts: (i) solution of an elliptic/parabolic pressure equation to compute fluid pressures and (phase) velocities, and (ii) solution of a hyperbolic/parabolic equation to compute the fluid transport. In recent years, there have been several advances towards

Date: Draft August 19, 2008.

more accurate and efficient (elliptic) pressure solvers, including advanced spatial discretisation schemes (multipoint schemes, mixed finite elements, etc), efficient linear algebra solvers like algebraic multigrid, and rescaling and multiscale methods for reducing the number of unknowns to be updated. As a result, the bottleneck in many simulators is often the solution of the (nonlinear) fluid transport equations. Although several advanced high-resolution methods were reported in the literature 10–20 years ago (see e.g., [2, 3, 13, 14]), the most wide-spread discretisation scheme is still the first-order, single-point upwind scheme. One possible reason for this, is the need for implicit time-stepping due to large differences in time constants throughout the spatial domain, which leads to a nonlinear algebraic system that needs to be solved using e.g., a Newton–Raphson method. Assembling and solving a large nonlinear system is often very expensive, even for a simple first-order method, and using a higher-order spatial discretisation introduces extra couplings and increases the nonlinearity of the discretised equations.

In this paper we consider a general class of implicit upwind schemes for solving purely advective transport in the absence of gravity and capillary forces. We present two key ideas to derive efficient, robust and accurate schemes. The first idea is a reordering procedure that will greatly reduce the runtime and the memory requirements needed to compute each time step. For instance, for the implicit single-point upwind scheme, the reordering procedure yields a very simple and efficient nonlinear solver that can be applied to simulate flow on grids with 10^6–10^7 grid cells with an efficiency resembling (and sometimes surpassing) that of modern streamline methods (see Section 4.1). The reordering idea is simple to explain and is applicable to upwind discretisations of pure advective flow on general, unstructured grids.

We note in passing that the reordering ideas presented here are not new. Duff and Reid [12] have applied reordering of linear systems to obtain a block-triangular structure as a general tool in numerical linear algebra. Dennis et al. [11] have previously explored the use of triangular structures in nonlinear systems to construct effective Newton-Raphson-type nonlinear solvers. Wang and Xu [32] have more recently used reordering of elements to build efficient iterative methods for convection-dominated problems. However, as far as we know, these ideas have not previously been applied to transport in porous media and in the current paper we give them a clear motivation based on the underlying physics.

The second idea is to use a discontinuous Galerkin formulation in combination with an implicit temporal discretisation to derive conservative schemes that allow for large time steps and high-order spatial discretisation using compact stencils. Applying the reordering idea to the corresponding discrete equations makes it possible to use second- or higher-order spatial discretisations in implicit schemes with moderate cost in terms of computation time and memory requirements. A high-order spatial discretisation generally reduces the numerical smearing and is essential to get stable solutions of miscible flows (as we will see in Section 4.3). Moreover, the flexibility of the discontinuous Galerkin discretization easily allows for local and adaptive *hp*-refinement, but this will not be explored in the current paper.

The outline of the paper is as follows. In Section 2 we present the basic mathematical models and motivate the two key ideas. In Section 3 we present the ingredients of our schemes: the variational formulation and the discontinuous Galerkin discretisations are presented in Section 3.1; the optimal ordering and the element-wise solution procedure are presented in Section 3.2; and finally, all the pieces are put together in Section 3.4. The efficiency, accuracy and robustness of the corresponding class of implicit upwind schemes are discussed in Section 4, where we discuss three different flow models: a two-phase model, a three-phase model, and a three-component miscible flow model. Finally, we sum up our observations and make some concluding remarks in Section 5.

2. Mathematical Models

In this paper we study advective flow in a porous medium, that is, we neglect gravity and capillary forces and consider viscous flow driven only by pressure forces. The fact that viscous flow is unidirectional along streamlines will be the key property used to derive our highly efficient upwind solvers. As for streamlines, more general flows can be computed using operator splitting to include gravity and capillary forces [4, 17].

To obtain a model for the nonlinear transport of ℓ phases or components through a porous medium, one usually starts with the conservation of mass for each phase and/or component. By simple arguments, this can be rewritten as a system of nonlinear conservation laws for the saturations, or volume fractions, $u = (u_1, \dots, u_{\ell-1})$:

$$\phi \partial_t u_i + \mathbf{v} \cdot \nabla f_i(u) = q_i, \quad i = 1, \dots, \ell - 1. \tag{1}$$

Here ϕ is the porosity of the medium and f is the fractional flow function modelling the speed of each phase or component relative to the mean (or total) velocity $\mathbf{v}$. The volume fraction of the last fluid is given by $u_\ell = 1 - \sum_i u_i$. In the absence of gravity, most flow models are unidirectional in the sense that $f \;\; > 0$ for scalar equations or that the Jacobian of f has distinct, positive eigenvalues. This property will be essential for our development of efficient solvers based on reordering.

The total Darcy velocity $\mathbf{v}$ is an average velocity for the bulk flow of fluids and is given by a conservation equation for the total mass. For an incompressible mixture of fluids, this equation can be written as

$$-\nabla \cdot \mathbf{v} = Q, \qquad \mathbf{v} = -K \lambda_T(u) \nabla p, \tag{2}$$

where p is the total pressure and K is the absolute permeability of the medium. The total mobility $\lambda_T(u)$ is a nonlinear function that models the reduced mobility of each fluid due to the presence of other fluids.

To solve the coupled system (1)–(2) we will use operator splitting. We are then (relatively) free to choose well-suited methods for each equation separately. Equation (2) is called the “pressure equation” and is usually solved using either a finite-difference or a finite-element method. Henceforth, we will assume that such a solver is available and can produce a velocity field $\mathbf{v}$ given in the form of fluxes that are constant on each element interface.

Solution of the transport equations (1) is in many cases the most time consuming step in an operator splitting scheme since saturations typically must be evolved through many time steps in order to accurately predict the transport of fluid. For explicit temporal discretisations, stability of each time step is ensured by a CFL condition stating that waves can only pass through a single cell in one time step. Unfortunately, this stability restriction may enforce prohibitively small time steps due to large spatial variations in the mean velocity $\mathbf{v}$ and in the porosity ϕ (in particular for real-life models). Implicit discretisations capable of taking large(r) time steps are therefore often preferred in practical computations. Although implicit schemes are more diffusive than their explicit counterparts, they yield better stability. Moreover, for cases with strong heterogeneity, our experience is that the (most severe) time-step restrictions arise in the near-well area or in cells with very small porosity that fill up almost instantly and have little effect on the global flow pattern. Choosing the time step to obey a CFL restriction in such cells means that many other cells are updated with an effective CFL number that is orders of magnitude smaller than one. Using an implicit scheme, one can instead choose time steps that give reasonable effective CFL number in these cells. Although this will lead to very high CFL numbers and excessive diffusion in cells with small porosity, the spatial accuracy in the majority of the domain will not be significantly affected.

The downside of implicit schemes is that the computational cost of these schemes is larger because each time step involves the solution of a system of nonlinear equations. Therefore, the efficiency of implicit schemes depends on an efficient linearization method in combination with efficient linear solvers or on a fast nonlinear solver as will be developed later in the paper.

Implicit temporal discretisation of (1) yields a boundary-value problem on the form

$$\alpha u + \mathbf{v}\cdot\nabla f(u) = \beta \text{ in } \Omega, \qquad u = h(\mathbf{x}) \text{ on } \partial\Omega^-, \tag{3}$$

where α and β are bounded functions and $\partial\Omega^-$ denotes the inflow boundary. In [24] we presented a family of efficient discontinuous Galerkin schemes for stationary linear transport, i.e., for the special case with $f(u)=u$ and $\alpha=0$. In the next section, we will extend these schemes to include (3). The key to an efficient scheme is the observation that all transport in the domain is directed, i.e., that all waves in the conservation law (1) travel forward along integral curves of the mean velocity field $\mathbf{v}$. By using a discontinuous Galerkin scheme combined with an upwind flux approximation on cell interfaces, the directedness of (1) is preserved in the discrete system in the sense that the solution in an element only depends on the solution in its immediate neighbours on the upwind side.

3. A Family of Implicit Discontinuous Galerkin Schemes

Accurate solution of hyperbolic conservation laws has been the focus of research for decades. Modern numerical schemes for hyperbolic conservation laws often combine an explicit temporal discretisation with a complex spatial discretisation and a limiter to avoid spurious oscillations. One such method is the Runge-Kutta discontinuous Galerkin method [8, 9, 22, 28], which has been successfully applied to different systems of hyperbolic conservation laws. The discontinuous Galerkin (dG) scheme was first proposed by Reed and Hill [28] for an equation modelling neutron transport. Lasaint and Raviart [22] analysed the method for the neutron transport problem and proved an $\mathcal{O}(\Delta\mathbf{x}^n)$ convergence rate. Later, Cockburn and Shu [8, 9] extended the method to general systems of hyperbolic conservation laws and convection-dominated problems by using stable, explicit Runge-Kutta schemes and limiters. The advantage of the discontinuous Galerkin method in the present context is that we can get arbitrarily high, formal order of accuracy with a compact stencil. We refer the reader to Epshteyn and Rivière [16] for an overview of discontinuous Galerkin methods applied to flow in porous media. To the best of our knowledge, this is the first paper to consider *implicit* dG methods, which, unlike our work, are applied to the coupled pressure–saturation system.

In the field of porous media flow, explicit temporal discretisations are usually considered impractical due to large magnitude variances of the Darcy velocity and the cell pore volumes. We therefore propose to combine the discontinuous Galerkin spatial discretization with an implicit time stepping procedure and use a limiter function to avoid (or reduce) spurious oscillations. To this end we consider (3) and assume that the equation is a temporal semi-discretised version of (1). For instance, using the θ-rule in time, the coefficients of (3) read

$$\alpha^n = \frac{1}{\theta\Delta t}, \qquad \beta^n = \frac{U^{n-1}}{\theta\Delta t} - \frac{1-\theta}{\theta}\,\mathbf{v}\cdot\nabla f(U^{n-1}), \tag{4}$$

where U^{n-1} and U^n are approximations of $u(\cdot, t^{n-1})$ and $u(\cdot, t^n)$, respectively. Throughout this paper, we use (4) with $\theta = 1$, corresponding to a fully implicit discretisation, to allow arbitrarily large time steps, at least in principle.

Notice that implicit time-discretisations do not have the *strong stability preserving* property [18] for orders higher than one. (Strong stability is stability in a nonlinear rather than in the usual linear sense, here meaning that the discretization is stable with respect to total variation). Higher-order implicit temporal discretisations may therefore cause spurious oscillations in the presence of discontinuities or kinks in the solution, even for a stable spatial discretisation. The onset of oscillations makes the discrete nonlinear system harder to solve, and effectively imposes a limit on the magnitude of the time step. Here we do not look into these issues, but tacitly assume that our time steps are within the stability limit henceforth. In our implementation, we have a simple and pragmatic mechanism that reduces the time step if the nonlinear iterations do not converge.

3.1. **Spatial Discretisation.** The discretisation in a discontinuous Galerkin method starts with a variational formulation, as in a standard Galerkin method. The difference is that discontinuous Galerkin methods allow discontinuities at element interfaces. To get a variational formulation of (3), we partition the domain into non-overlapping elements $\{K\}$. Let V be a space of arbitrarily smooth functions. By multiplying (3) with a function $\varphi \in V$ and integrating by parts over each element K, we get

$$\int_K (\alpha u - \beta)\varphi - \int_K f(u)\ \nabla\cdot(\mathbf{v}\varphi) + \int_{\partial K} [\varphi f(u)]\ \mathbf{v}\cdot\mathbf{n} = 0, \quad \forall \varphi \in V,$$

where $\mathbf{n}$ is the outward pointing normal to ∂K and the bracket in the last integral signifies that the integrand in principle may be discontinuous at all points along the integration path. We seek a solution in a finite-dimensional subspace $V_h \subset V$. To this end, we replace the exact solution u and the test function φ with $u_h \in V_h$ and $\varphi_h \in V_h$, respectively. For V_h we choose a space of element-wise smooth functions that may be discontinuous over element boundaries. This means that we must replace the flux $f(u)\mathbf{v}\cdot\mathbf{n}$ with a consistent and conservative numerical flux $\widehat{f}(u^+, u^-, \mathbf{v}\cdot\mathbf{n})$, involving the one-sided values $u^\pm$. This gives the following discrete variational formulation; let

$$a_K(u,\varphi) = \int_K (\alpha u - \beta)\varphi\, dx - \int_K f(u)\ \nabla\cdot(\mathbf{v}\varphi)\, dx + \int_{\partial K} \widehat{f}(u^+, u^-, \mathbf{v}\cdot\mathbf{n})\varphi\, ds,$$

and find u_h such that

$$a_K(u_h, \varphi_h) = 0 \qquad \forall K \text{ and } \forall \varphi_h \in V_h. \tag{5}$$

Here $\widehat{f}$ is the upwind flux given by

$$\widehat{f}(p^+, p^-) = f(p^+)\max(\mathbf{v}\cdot\mathbf{n}, 0) + f(p^-)\min(\mathbf{v}\cdot\mathbf{n}, 0), \tag{6}$$

for inner and outer approximations p^+ and p^- at element boundaries. Although other flux approximations are consistent, they may not preserve the directional dependency we rely on to later reorder the equations. For instance, the well-known Lax-Friedrichs flux yields a consistent approximation of the inter-element fluxes, but creates a bidirectional dependence between all elements.

To fix ideas, we assume for simplicity of presentation that $\Omega \subset \mathbb{R}^3$ and that the elements K are hexahedra in a regular Cartesian grid. We will use the polynomial spaces $\mathbb{Q}^n = \text{span}\{x^p y^q z^r : 0 \le p, q, r \le n\}$ and $\mathbb{P}^n = \text{span}\{x^p y^q z^r : 0 \le p+q+r \le n\}$). The dimension of $\mathbb{Q}^n$ is $(n+1)^3$, whereas the dimension of $\mathbb{P}^n$ is $(n+1)(n+2)(n+3)/6$.

Let $V_h^{(n)} = \{\varphi : \varphi|_K \in \mathbb{P}^n\}$. Products of Legendre polynomials $L_k(\xi,\eta,\zeta) = \ell_r(\xi)\ell_s(\eta)\ell_t(\zeta)$ form a convenient basis for these spaces. The approximate solution in an element K_i can then be written as

$$u_h^i(x,y,z) = \sum_{k=0}^{N} U_{ik}\, L_k\Big(\frac{2(x-x^i)}{\Delta x^i}, \frac{2(y-y^i)}{\Delta y^i}, \frac{2(z-z^i)}{\Delta z^i}\Big), \tag{7}$$

where N is the number of basis functions, (x^i, y^i, z^i) is the centre of element K_i, and $\{U_{ik}\}_k$ are the unknown coefficients to be determined. Thus, $V_h^{(0)}$ is the space of element-wise constant functions and yields a formally first-order accurate scheme, $V_h^{(1)}$ is the space of element-wise trilinear approximations and yields a formally second-order accurate scheme, etc. In the following, we use dG(n) to denote the discontinuous Galerkin approximation of formal order $n+1$. In other words, the error of a dG(n)-method will decay with order $n+1$ for smooth solutions. On non-smooth solutions, slower convergence is to be expected. Notice also that dG(0) coincides with the single-point upwind (SPU) method.

We remark that although the polynomial spaces $V_h^{(n)}$ are sufficiently rich to yield a scheme that has formal accuracy of order $(n+1)$, we have for simplicity used the tensor-product space $W_h^{(n)} = \{\varphi : \varphi|_K \in \mathbb{Q}^n\}$ for purely two-dimensional simulations. Thus, the number of unknowns per element in a dG(n)-method is $m = (n+1)^2$ in two spatial dimensions and $m = (n+1)(n+2)(n+3)/6$ for three spatial dimensions.

By substituting the tensor-product Legendre polynomials in the variational formulation (5), we get a system of nonlinear equations for the unknowns in the domain. Let U be the vector of unknowns $\{U_{ik}\}$ and let

$$G_K(U)_j = a_K^h(u_h, L_j),$$

where a_K^h is an approximation of a_K using Gaussian quadrature. If U_K denotes the unknowns in element K, we can write the equations for element K as

$$0 = G_K(U) = M_K U_K - B_K - F_K(U_K) + \widehat{F}_K(U),$$

where we for convenience have split the form G_K in four parts: three integrals over K plus the element coupling given by the flux $\widehat{F}_K(U)$ integrated over ∂K. Due to the directional dependence of the upwind flux function (6), the flux can be split in two parts

$$\widehat{F}_K(U) = \widehat{F}_K^{+}(U_K) + \widehat{F}_K^{-}\big(U_{\mathcal{U}(K)}\big)$$

describing flow out of and into element K, respectively. Here $\mathcal{U}(K) = \{E \in \Omega : \partial E \cap \partial K^- \neq \emptyset\}$ denotes the neighbouring elements of K in the upwind direction. Thus, the nonlinear system for element K can be written as

$$\begin{aligned} G_K(U) &= M_K U_K - B_K - F_K(U_K) + \widehat{F}_K^{+}(U_K) + \widehat{F}_K^{-}\big(U_{\mathcal{U}(K)}\big), \\ &= G_K^{+}(U_K) + G_K^{-}\big(U_{\mathcal{U}(K)}\big) = 0. \end{aligned} \tag{8}$$

Assembling all the local nonlinear systems gives a global nonlinear system $G(U) = 0$. In the linear case (with $f(u) = u$, see [24]), this results in a *reducible* block-structured linear system $Ax = b$. Furthermore, there exists a symmetric permutation P that maps the global coefficient matrix A to a lower block-triangular matrix $L = PAP^T$. If we are able to determine P, we can therefore solve the linear system by solving a sequence of smaller linear problems corresponding to each *irreducible* diagonal block of L. The size of each diagonal block is given by the number of degrees–of–freedom in the corresponding single element or group of mutually dependent elements. The resulting computational complexity is close to optimal in the linear case: if LU-factorisation is used for each subsystem, the asymptotic complexity is linear in the number of irreducible blocks (which in the best case coincides with the number of elements) and roughly cubed in the degrees-of-freedom per element.

In the nonlinear case, the Jacobian matrix J_G must have the same block-structure as A. Thus, $J_F = PJ_GP^T$ will have a lower block-triangular structure as in the linear case. Alternatively, one can apply the symmetric permutation directly to the nonlinear equations and unknowns to get a system $F(\tilde{u}) = PG(\tilde{u})$, $\tilde{u} = P^T u$ with the irreducible lower block-triangular form

$$F(\tilde{u}) = \begin{pmatrix} F_1(\tilde{u}_1) \\ F_2(\tilde{u}_1, \tilde{u}_2) \\ \vdots \\ F_M(\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_M) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{9}$$

Here $\tilde{u}_j$ is the vector of unknowns corresponding to one or a few elements, and F_j are the equations that govern these degrees-of-freedom. With this structure, $\tilde{u}_i$ is independent of $\tilde{u}_j$ for $j > i$. In other words, we can compute the solution of the discrete nonlinear boundary-value problem (9) by solving a sequence of M smaller nonlinear systems using e.g., Newton-Raphson method.

In the next subsection we will demonstrate that an optimal permutation P can be found by considering the interface fluxes. However, rather than working on the linear system directly, we will present a method for reordering the elements and a corresponding solution procedure that visits the elements sequentially and solves (8) locally. Thus, neither the global Jacobi matrix J_G nor the nonlinear systems G need to be assembled. Instead, the solution procedure guarantees that $U_{\mathcal{U}(K)}$ is a vector of known variables when element K is processed and all one needs to assemble is the local Jacobi matrix J_{G_K}.

3.2. **Sequential Solution Procedure.** Using the upwind flux approximation in the discontinuous Galerkin schemes above preserves the unidirectional flow property of (1) exactly: The solution in element K depends on the solution at the inflow boundary ∂K^- and is independent of the solution elsewhere in the domain. This property implies that there is an inherent directional dependency between (collections of) elements. Under certain assumptions on the interface fluxes $v = \mathbf{v} \cdot \mathbf{n}$ (to be discussed below), this directional dependency permits us to compute the solution sequentially, one element at a time.

Consider the directed graph defined by assigning a vertex to each element K_i and an edge for the flux v_{ij} between elements K_i and K_j, see Figure 1. An edge from vertex i to vertex j in the graph implies that the solution in element j depends on the solution in element i. This dependency graph is directed and is given entirely by the interface fluxes v_{ij}. Suppose that we can find a sequence of element numbers $\Pi = (p_1, p_2, \dots, p_M)$ such that if U_{p_j} depends on U_{p_i}, then $i < j$ such that p_i appears before p_j in Π. Then, by definition, we can compute the solution in element p_1 independently of the rest of the elements using only values on the inflow boundary. Likewise, the sequence guarantees that U_{p_i} is either given on the inflow boundary or can be computed from $U_{p_1}, \dots, U_{p_{i-1}}$, once these are known. This means that the global nonlinear system reduces to a sequence of local systems,

$$
\begin{array}{lll}
G^+_{p_1}(U_{p_1}) & = 0, & \\
G^+_{p_2}(U_{p_2}) & = -G^-_{p_2}(U_{p_1}), & \\
\vdots & \vdots \quad \vdots & \\
G^+_{p_M}(U_{p_M}) & = -G^-_{p_M}(U_{p_1}, \dots, U_{p_{M-1}}). &
\end{array}
\tag{10}
$$

Here, each local system corresponds to a irreducible diagonal block in the nonlinear system and involves the degrees-of-freedom in a single element, or more generally, as we will discuss below, in a collection of mutually dependent elements. In other words, we recover the solution of the full nonlinear system $G(U) = 0$ by solving one sub-system at a time.

If there exists a sequence Π of single elements, it can easily be obtained by a classical graph algorithm. The task of arranging vertices in a sequence Π according to their position in a directed graph is called a topological sort [29]. To see how a suitable sequence can be constructed, note that $i < j$ for any vertex p_i that can be reached from vertex p_j by going backwards in the graph. In Figure 1, Element 5 can be reached going backward from Element 2 and similarly Element 8 is reached from Element 5. Hence, in the reordered sequence, they end up in the order $\{8, 5, 2\}$. A topological sort can easily be obtained by looping through the vertices in the graph in any order and initiate a depth-first search of the reversed graph from each vertex that has not been previously visited. During the depth-first search, we add vertex p_j to the reordered sequence when the search backwards from p_j has been completed. In Figure 1 we start at Element 9 and go down through

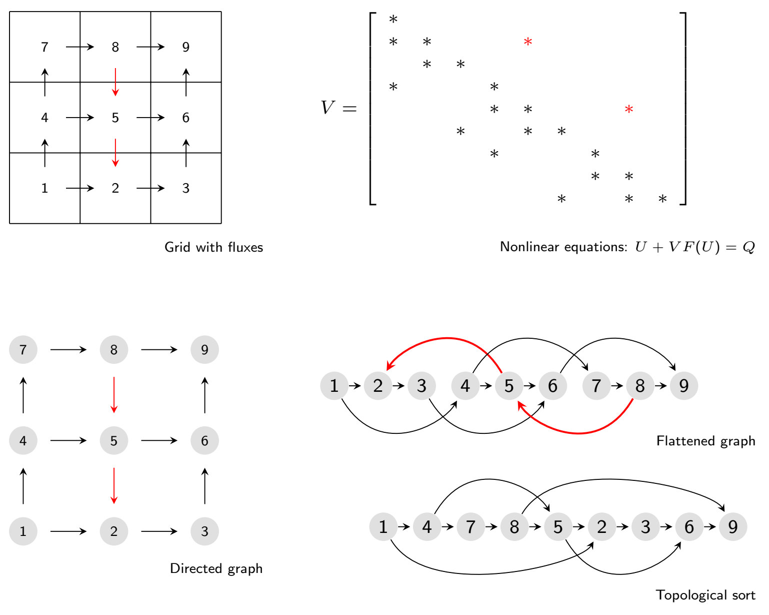

FIGURE 1. Illustration of the reordering procedure for a simple 3×3 grid.

Elements 8, 7, and 4 until we reach Element 1, which has no outgoing edges in the reversed graph and is therefore added as the first element in the reordered sequence. We then backtrack, adding Elements 4, 7, and 8, for which all outgoing edges point to vertices that have been visited during the backtracking phase. Next we visit Element 6 and then Element 5, for which the only outgoing edges point to Elements 4 and 8 that have already been added. We then add Element 5 and backtrack to Element 6, and so on.

Since a depth-first search only visits each vertex once, the topological sort of a directed graph can be obtained in $\mathcal{O}(n)$ operations for n vertices. The directed graphs that can be sorted topologically are characterised by being acyclic. If there are cycles in a graph, a forward (backward) traversal can reach a vertex an arbitrary number of times, and a sequence that fulfils our requirement does not exist. If a sequence Π cannot be found by a single depth-first traversal, the graph has at least one cycle (or strongly connected component). Cycles correspond to groups of elements that are made mutually dependent by nonzero circulation in the discrete set of fluxes $\{v_{ij}\}$. This mutual dependence makes a topological sort impossible. For the discrete equations, this implies that the degrees-of-freedom in such a collection of cells form an irreducible diagonal block in the Jacobian matrix of PJ_GP^T and must be solved for simultaneously.

Fortunately, there is an easy solution to this problem. By grouping together elements corresponding to strongly connected components in the dependency graph, we obtain an acyclic graph, where each vertex corresponds to either a single element or a single strongly connected component (i.e., a group of elements). Strongly connected components can easily be detected by one extra depth-first traversal of the graph. Alternatively, Tarjan's ordering algorithm [31] can be used. Thus, if our solver can compute the solution in groups of elements, we can still apply the sequential solution procedure.

In passing, we remark that the reordering idea can also be applied when $\mathbf{v}$ is computed by a higher-order method, in which case there may be a few cell faces where the flux changes sign. Given a proper algorithm to detect these faces, the neighbouring cells can easily be treated as on a cycle and be lumped together.

3.3. **Treatment of Cycles.** The number and size of cycles in the discrete fluxes depend on the heterogeneity and on the numerical method used to discretize the pressure equation. For instance, if one uses a two-point flux-approximation scheme for the pressure equation (2), a simple monotonicity argument on the pressure shows that the discrete velocity field is guaranteed to contain no cyclic dependencies, see [24]. More general schemes like mixed finite elements, multipoint flux-approximation schemes [1], or mimetic finite differences [5, 6] may produce velocity fields or fluxes with cyclic dependencies. Theoretically, all cells in the domain can be coupled and reordering the cells will not give any speedup for the solution of the global nonlinear system. However, this is a worst case scenario that we have so far never encountered, despite extensive testing on both synthetic and real-field cases with strong anisotropy. In our experience, large cycles rarely occur and if they occur it is mainly for rough grids and permeability fields with large-scale structures that dominate flow, such as fluvial or layered permeability fields with sharp contrasts. For cases with shorter correlation lengths or lack of strong directional trends, the fraction of cells in cycles is typically small compared with the total number of cells in the domain. See also the discussion in [25] of how to avoid nonmonotone solutions (and thereby cycles) in multipoint schemes.

To obtain an efficient solver for cases containing cycles, some care should be taken when choosing linear solver(s) for the block problems. For small-sized problems corresponding to a single or a few connected cells, a direct solver is the best choice. For medium-sized cycles, we are currently using the UMFPACK library [10].

Alternatively, or as a complement, one may preprocess the discrete velocity field and cut edges with small flow, e.g., by setting all fluxes with magnitude below a certain user-given threshold to zero. This will not only reduce the number of cycles, but also take away cells with small flow. Cutting small fluxes will introduce small errors in the local mass balance. The thresholding of small fluxes may therefore be followed by a repair algorithm, in which one seeks to get the mass balance right again. That is, in each cell C_i, one introduces a scaling α_i of the outflow fluxes determined by

$$\alpha_i \sum_{j\in U(i)} \max(v_{ij},0) = q_i - \sum_{j\in U(i)} \min(v_{ij},0). \tag{11}$$

If the inflow fluxes are known, each outflow flux is scaled by α_i, and mass conservation is retained. In the case of cycles, (11) gives a linear system for the unknown scaling factors of all cells involved in the cycle. By solving this system, the mass balance in the whole cycle is cured.

3.4. **The Complete Scheme.** The upwind discontinuous Galerkin scheme (5) yields an approximation of the solution in terms of a polynomial $u_h^i(x,y,z)$ for each element K_i in the domain. For higher-order approximations we need to remove or reduce any spurious oscillations that may appear near discontinuities and kinks in the solution to obtain a stable scheme for (1). To this end, we apply a variant of Cockburn and Shu's TVB limiter [8] to postprocess the solution. To avoid limiting in regions where the solution is smooth, we check the magnitude of the jump $[u_h]$ across each element faces. If the jump at some point on ∂K is larger than some prescribed constant, we reduce the polynomial order of the local approximation in element K to a trilinear surface with the slopes being restricted by the minmod limiter. In other words, the TVB limiter $\Lambda\Pi_h$ is given by

$$\Lambda\Pi_h u_h^i = \begin{cases} \begin{aligned}\overline{u} \quad &+\overline{u}_x(x-x_i)\\ &+\overline{u}_y(y-y_j)+\overline{u}_z(z-z_j),\end{aligned} & \text{if } |[u_h]| > M \text{ on } \partial K\\ u_h^i, & \text{otherwise,}\end{cases} \tag{12}$$

where

$$\overline{u}_x = \frac{2}{\Delta x}\text{minmod}(u_x, \theta(\overline{u}_{i+1,j,k}-\overline{u}_{ijk}), \theta(\overline{u}_{ijk}-\overline{u}_{i-1,jk})), \text{ etc.}$$

Here $\overline{u}$, u_x, u_y, and u_z are the degrees-of-freedom corresponding to the basis functions 1, $\ell_1(\xi)$, $\ell_1(\eta)$, and $\ell_1(\zeta)$, respectively; that is, the constant and the slopes in the x-, y- and z-directions. The minmod function is defined as

$$\text{minmod}(a_1,\dots) = \begin{cases} \max_i a_i, & a_i < 0 \ \forall i,\\ \min_i a_i, & a_i > 0 \ \forall i,\\ 0, & \text{otherwise.}\end{cases}$$

For practical reasons, we only check the size of the jump in the approximation u_h at the midpoints on each element interface.

To solve the coupled system (1)–(2) we use a standard operator splitting and advance the total solution in time steps Δt. First, we compute the pressure p^k and mean velocity $\mathbf{v}^k$ at time t^k by solving (2) with the total mobility $\lambda_T(u^{k-1})$ fixed. Then we fix $\mathbf{v}^k$ (and p^k) and use an implicit discontinuous Galerkin scheme (5) to evolve the transport equation (1) a time step Δt. To obtain a reasonable accuracy, a small time step $\delta t < \Delta t$ is usually required. For schemes of second order or higher, we postprocess the solution by applying the limiter (12) to each component of the solution and each element in the domain to remove over- and undershoots in the solution. Optionally, we may also perform extra step(s) to add gravity and capillary forces. Then we solve (2) again, and the procedure is repeated until the desired final time T is reached.

The accuracy of the solution obtained with this scheme depends on the choice of time steps Δt and δt, as well as on the degree of coupling between (1) and (2). For a medium with relatively homogeneous porosity (e.g., Case 1 in the next section), we would typically choose a CFL number somewhere between one and ten to avoid too much smearing. For a medium with large variations in porosity and the magnitude of fluxes (e.g., Case 2 in the next section), one would typically want to choose a similar CFL target for the most 'typical' cells, thereby avoiding the time-step to be severely restricted by cells having a small volume and limited influence on the global solution. In commercial codes, it is common to impose a restriction on the maximum change in saturation allowed in single cell as an extra precaution. On the other hand, if one is merely interested in production curves and does not care too much about the spatial resolution elsewhere, one can use surprisingly high CFL numbers, as we will see in Case 2 in the next section.

For incompressible and weakly compressible flows, one can typically allow Δt to be much bigger than the time step δt used to solve (1) because the pressure and velocity fields are not very sensitive to changes in the total mobility λ_T. We refer the reader to [26] for a simplified analysis to support the choice of Δt.

4. Numerical Examples

In this section we assess the efficiency, accuracy, robustness, and flexibility of the dG schemes by applying them to three different models for incompressible multiphase flow in porous media: (i) two-phase flow, (ii) three-phase flow [20], and (iii) miscible two-phase, three-component flow [19]. Unless stated otherwise, we use a standard two-point finite volume scheme to discretize the pressure equation (2).

4.1. **Two-Phase Flow.**

We start by considering two-phase incompressible, immiscible flow of oil and water in the absence of gravity and capillary forces. As our primary unknown we pick the water saturation s. Then the conservation equation for water saturation is given by (1) with $\ell = 2$; that is,

$$\phi s_t + \mathbf{v}\cdot\nabla f(s) = 0. \tag{13}$$

Here the fractional flow or flux function reads $f(s) = \lambda_w(s)/(\lambda_w(s) + \lambda_o(s))$, where λ_w and λ_o are the mobilities of water and oil, respectively, given by

$$\lambda_w(s) = \mu_w s^2, \quad \lambda_o(s) = \mu_o(1-s)^2.$$

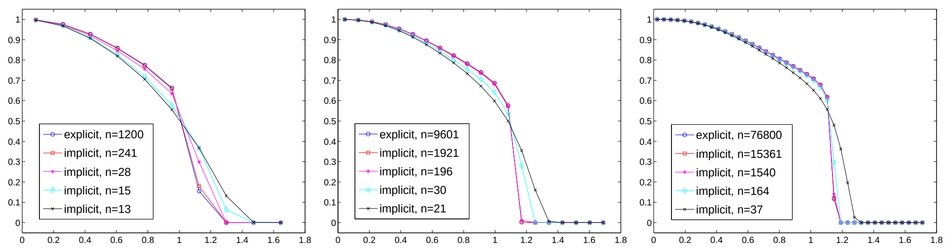

FIGURE 2. Saturation along the diagonal computed using n time steps on a $N \times N \times N$ grid for $N = 10, 20, 40$.

A backward Euler discretisation of (13) yields nonlinear boundary-value problems of the form,

$$\frac{\phi}{\Delta t} s^n + \mathbf{v} \cdot \nabla f(s^n) = \frac{\phi}{\Delta t} s^{n-1}. \tag{14}$$

In the following we will consider a few test cases in two and three spatial dimensions. In these testcases we have used $M = 0.0$ in (12). A few more test cases for incompressible and weakly compressible flow can be found in [23].

Case 1. The first model we consider is a simple quarter five-spot with unit permeability and porosity, where the flow is driven by two rate-controlled wells placed in diagonally opposite corners. In a water-injection scenario, solutions to (13) will typically consist of a shock followed by a rarefaction wave.

We start by a qualitative comparison of implicit versus explicit temporal discretization; the latter has previously been the method of choice with dG discretizations. To this end, we consider the unit cube represented by a uniform $N \times N \times N$ grid. We place an injector in cell $(1, 1, 1)$ and a producer in cell (N, N, N). To keep matters simple, we only consider dG(0) with first-order Euler temporal discretization and without any reordering, and we compute the velocity field only once, initially. Figure 2 shows the saturation along the diagonal computed with the explicit and the implict scheme. For the explict scheme we use the maximal time step allowed by the CFL condition, giving n_e equally spaced time steps. For the implicit scheme we use a very simple adaptive time-stepping: first we try to use $\lfloor n_e/M \rfloor$ (or one if zero) equally spaced time steps for $M = 5$, 50, 500, and 5000. If the Newton–Raphson method fails to converge, we reduce the time step by one half until the time step converges. Similarly, if we are using a reduced time step and this has been stable for two consecutive time steps, we double the next time step. As we can see from the figure, one can safely reduce the number of implicit time step two or three orders of magnitude compared with the explicit scheme without significantly changing the saturation profile by excessive numerical dissipation. Henceforth, we therefore tacitly assume that our implicit dG schemes can use relatively large implicit time steps.

TABLE 1. Runtimes and average number of nonlinear iterations per cell per time-step versus the time-step Δt for Case 1. The time used to compute the permutation P was 8.0×10^{-4} seconds.

Δt	NR–UMFPACK		NR–PFS		NPFS	
days	time (sec)	iterations	time (sec)	iterations	time (sec)	iterations
2	1.34e-01	13.88	2.45e-02	13.88	2.48e-03	1.98
4	1.43e-01	14.69	2.54e-02	14.69	2.71e-03	2.27
8	1.48e-01	15.12	2.59e-02	15.12	3.06e-03	2.65
16	1.47e-01	15.00	2.53e-02	15.00	3.41e-03	3.17
32	1.48e-01	15.00	2.62e-02	15.00	3.97e-03	3.84

Second, we will assess the efficiency of our reordering strategy for the simple dG(0) discretization of (14). To this end, we consider a $100 \times 100 \times 1$ grid and compare three different solvers for the nonlinear system arising from (14) in each time step. The first nonlinear solver (NR–UMFPACK) is a plain implementation of Newton–Raphson's method with the direct sparse solver from the UMFPACK library [10]. To get a robust scheme we use 0.5 as initial guess in all iterations, use a relaxation factor of 0.9, and iterate until the nonlinear residual is less than 10^{-6}. In the second solver (NR–PFS), we replace the direct sparse linear solver by a permuted forward substitution (PFS) solver, which is the same as applying reordering to the discrete system arising from the Newton–Raphson linearization. The third solver (NPFS) is the reordering method introduced above, where we use Ridder's method [27] for the scalar nonlinear equations and the NR–UMFPACK method described above for strongly connected components.

Table 1 reports the elapsed CPU time per time-step for different choices of the time-step. We first observe that NR–PFS is almost an order of magnitude faster than NR–UMFPACK. The better performance of NR–PFS is directly attributed to the efficiency of the PFS linear solver. Secondly, we observe that NPFS is an order of magnitude faster than NR–PFS. This difference can to a large extent be attributed to the iteration count. Because the NPFS solver computes the solution cell by cell, it has the ability to vary the number of iterations used from on cell to the next. In fact, it will even use no iterations in cells where the nonlinear residual is zero, for instance, in the unswept zone, where saturation values are identically zero. The Newton–Raphson method, on the other hand, is bound to use the same number of iterations in all cells and therefore has one order of magnitude higher iteration count.

To further assess the efficiency of the reordering method, we consider a homogeneous quarter five-spot consisting of N^3 cells to see how the runtime grows with increasing N. Figure 3 shows the average computational time per element per time step for $N = 10, 15, \ldots, 50$ measured at times corresponding to approximately 0.4, 0.8 and 1.2 pore volumes injected for dG(0).

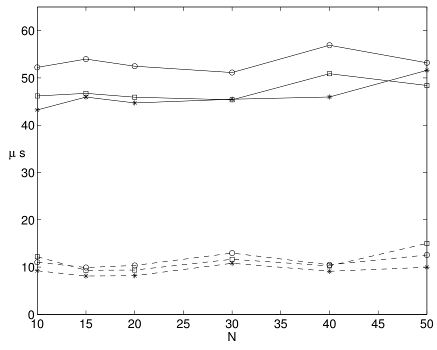

FIGURE 3. Average computational time per element per time step (in μs) for a homogeneous quarter-five spot simulations with N^3 elements computed with dG(0) (dashed lines) and dG(1) (solid lines) at times 0.4, 0.8 and 1.2 PVI. The implementation is not optimised.

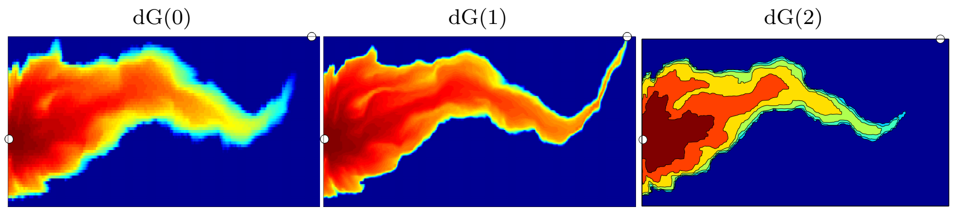

FIGURE 4. Saturation at time $t = 0.25$ PVI computed by dG(n), $n = 0, 1, 2$ for Layer 6 in SPE 10. The wells are indicated by white circles.

Because there are no cycles in the velocity fields, the runtime per element is close to linear in the number of elements for both schemes. Figure 3 also shows timings for a unoptimized dG(1) solver, which also exhibits the same complexity as our optimised dG(0) code.

Case 2. Next, we compute water injection in a square 2D reservoir with permeability distribution sampled from Layer 6 of Model 2 from the 10th SPE Comparative Solution Project [7] and unit porosity. This layer is characterised by a smoothly varying (lognormal) permeability field that spans six orders of magnitude. The rectangular domain is partitioned in 220×60 cells of size 6.096×3.048 and an injection and a production well are placed in elements $(1, 24)$ and $(217, 60)$, respectively.

Figure 4 shows saturation profiles after 0.25 pore volumes of water have been injected computed with the standard single-point upwind method (i.e., dG(0)), dG(1), and dG(2). In the simulation the pressure has been updated three times and the time steps were chosen to meet a target CFL number

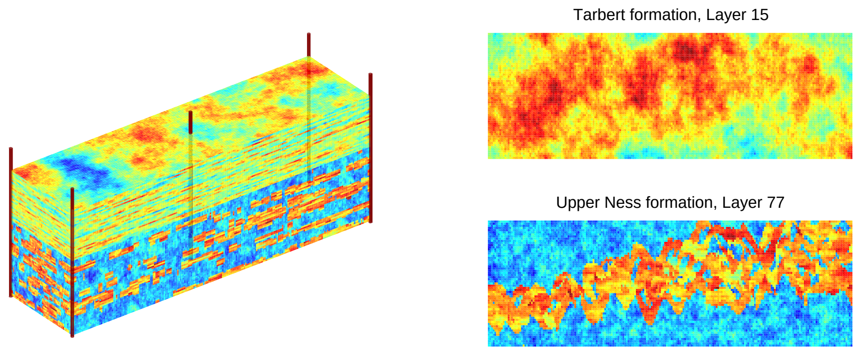

FIGURE 5. Model 2 from the 10th SPE Comparative Solution Project [7].

of 10. It is clear that the resolution increases significantly from the first-order single-point upwind method to the second-order dG(1), in particular for the thin water finger near the production well. However, increasing the accuracy from second to third order does not produce a significantly better resolution of the front. The reason for this is that the limiter (12) employed in the two higher-order methods effectively reduces the order of the schemes near jumps in the solution. The effect of higher order might be noticeable in the smooth part of the solution, but not near shocks.

Case 3. The full test case from the SPE 10 comparative solution project is a model with $60 \times 220 \times 85$ grid cells, see Figure 5. The 35 first layers of the domain consist a smooth Tarbert formation laid on top of 50 layers of a fluvial Upper Ness formation. This test case was originally designed as a benchmark for upscaling methodologies and as such includes many difficult details that are seen in real oil reservoirs.

Our production scenario consists of a five-spot configuration, where we inject water in a vertical well in the middle of the domain and produce oil and water from vertical wells placed in each of the four corners. The simulation time is 2000 days. For more details see [7]. We compute this two-phase problem with one initial pressure update using a standard two-point flux-approximation scheme. Figure 6 shows the water-cut curves for each of the four wells computed by an optimised implementation of the single-point upwind scheme (i.e., dG(0)) using uniform time steps of twenty days. The accuracy of the production curves is similar to what is obtained by a commercial streamline simulator. The total computational time of the transport steps computed by dG(0) is two minutes, which is approximately the same as for the transport step in a *highly optimised* streamline solver [30]. This makes the standard implicit first-order upwind scheme with reordering a prime candidate for applications where one usually has to resort to streamlines, like direct simulation of high-resolution grid models with multimillion cells, history matching, and screening of multiple geological models.

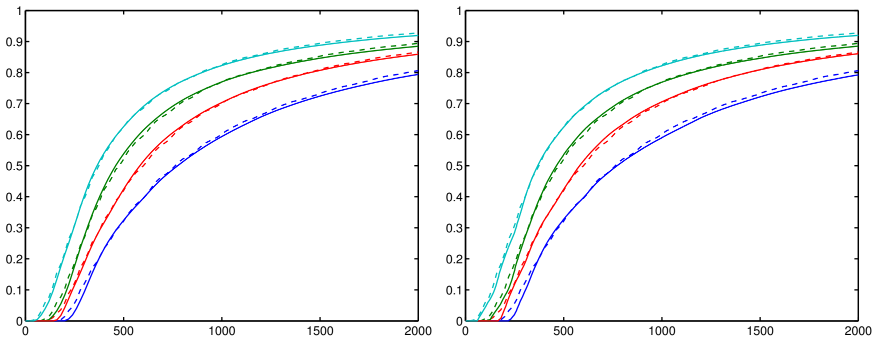

FIGURE 6. Water-cut curves for the full SPE 10 model computed with dG(0) using a time step of 20 days (left) and dG(1) using a time step of 1 day (right). Both computations are compared to a commercial streamline simulator (dashed line).

Figure 6 also shows production curves computed by dG(1) with uniform time steps of one day. Here the improved spatial accuracy of the higher-order method is masked by the numerical diffusion introduced by relatively long time steps used in the implicit discretisation. However, the point of the example is to demonstrate that it is possible to run a higher-order method on a standard desktop PC for a model with realistic and very complex heterogeneity and 1,1 million cells without the use of any kind of parallelisation.

Case 4. The corner-point grid format used in industry allows very general geometries to model real geological formations having as faults, throws, and pinched layers. A corner-point grid consists of a set of hexahedral cells that are aligned in a logical Cartesian manner. Figure 7 shows one such, artificially constructed, faulted reservoir having 80 220 grid cells and 254 869 cell faces. The grid is specified in terms of a set of vertical pillars defined over an areal Cartesian 2D mesh in the lateral direction. Each volumetric cell is restricted by four pillars and is defined by specifying the eight corner-points of the cell, two on each pillar. Our heterogeneity model is a layered permeability field with lognormal variation inside each layer spanning three orders of magnitude. We have placed one water injector and two producers in the reservoir, all modelled as sources uniformly distributed in all layers. The flow is described using the two-phase model introduced above with an oil-water viscosity ratio of ten.

For such complex models, a two-point scheme for the total pressure may not produce accurate solutions. We therefore solve simultaneously for the pressure and velocity using a mimetic finite-difference scheme [5, 6], which is very accurate and applicable for general polyhedral grids. However, the velocity field produced by this scheme will not in general be loop-free.

As shown in Case 3, accurate production curves can be obtained using fairly large time steps. We have computed the water flooding using two

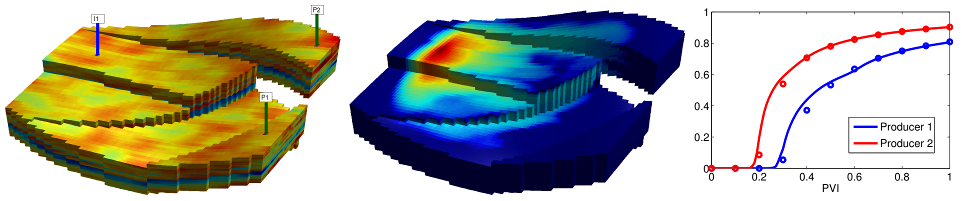

FIGURE 7. The left plot shows the fractured layered reservoir with lognormal permeability field. The middle plot shows water saturation after 0.18 pore volumes injection. The right plot shows water-cut curves for with 100 pressure steps (solid line) and 10 pressure steps (dots).

different time steps for the pressure solver. In the first computation, we used 100 pressure steps with 10 sub-steps in the saturation solver and in the second computation we used only 10 pressure steps and 10 saturation sub-steps. In both runs, the mimetic pressure solver produced velocity fields with approximately 200 loops involving about 1 100 cells altogether. Furthermore, the majority of the loops were small, coupling less than 15 cells. The largest loop that occurred in the simulations involved only 72 grid cells. This amount of loops does not significantly alter the speed, accuracy or robustness of the saturation solver. However, the efficiency of the solver would have decayed somewhat, of course, if a large percentage of the cells had been part of cycles.

Figure 7 shows a snapshot of the water saturation after 0.18 pore volumes of water has been injected in the first run. To show the accuracy obtained in the two runs, we have also plotted the water-cut curves in the two producers.

Case 5. Our next model is a real-life geological model of a North Sea sandstone formation, with one large fault and several smaller ones. The permeability and porosity fields are quite smooth with jumps between the layers, giving a mild heterogeneity where the ratio of largest to smallest permeability is 8.8×10^4. The grid, however, is quite rough with 27 437 cells, many of which are twisted and pinched. The ratio of largest to smallest grid cells is roughly 4×10^3 and the ratio of largest to smallest face areas is 10^6.

The model was originally used to study the feasibility of CO_2 deposition, but here we pretend that it is a petroleum reservoir. The reservoir covers roughly 50×50 kilometers and 1 kilometer in the vertical direction and the volumes and time scales involved are, of course, totally unrealistic, but the heterogeneity and geometrical complexity are representative also for a petroleum reservoir. We therefore scale the model by a factor 0.1 in each spatial dimension and place one producer in the center near the large fault and three injectors in different sectors, see Figure 8. All wells are vertical, pressure-controlled wells with bottom-hole pressures of 300 bar in the producer and 700 bar in the injectors.

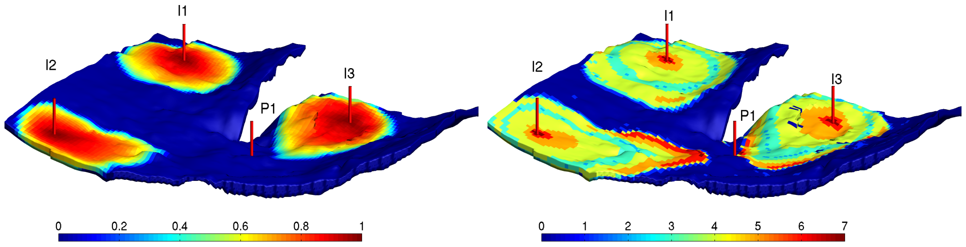

Figure 8. The Johansen formation from the North Sea. The left plot shows the water saturation after 1 000 days of water injection and the right plot shows the number of iterations per cell for a typical time step.

Table 2. Runtimes and average number of nonlinear iterations per cell per time-step versus the time-step Δt for waterflood simulations of the Johansen formation from the North Sea.

Δt	NR–UMFPACK		NR–PFS		NPFS	
days	time (sec)	iterations	time (sec)	iterations	time (sec)	iterations
125	2.26e+00	12.69	3.28e-01	12.69	4.44e-02	0.93
250	2.35e+00	12.62	3.32e-01	12.62	4.73e-02	1.10
500	2.38e+00	13.25	3.46e-01	13.25	4.16e-02	1.41
1000	2.50e+00	13.50	3.49e-01	13.50	4.21e-02	1.99

The left plot in Figure 8 shows the water saturation after 1 000 days of water injection. In the computation, the number of strongly connected components in the flux fields was on average 77.4, the number of cells in strongly connected components was around 780, and the largest component involved contained 380 cells. The time used to compute the permutation P was 3.6×10^{-3} seconds.

Table 2 reports a similar comparison of nonlinear solvers as given in Case 1. Also here we see that that NPFS is about one order of magnitude faster than NR–PFS, which again is about one order of magnitude faster than NR–UMFPACK. We also note the low iteration count for NPFS. To support the explanation given in Case 1, the right plot in Figure 8 shows the number of iterations in each cell for a typical time-step after the injection fronts have extended into the reservoir. The figure shows that no iterations are performed in the unswept zone and that the maximum number of iterations appears in the well blocks and in regions behind the injection fronts. In [23] we also show results for simulations with a compressible oil phase.

Case 6. Finally, we show that the schemes presented here can also be implemented for general unstructured grids. In Figure 9, we have included a direct simulation of flow in a synthetic fractured medium using dG(0)

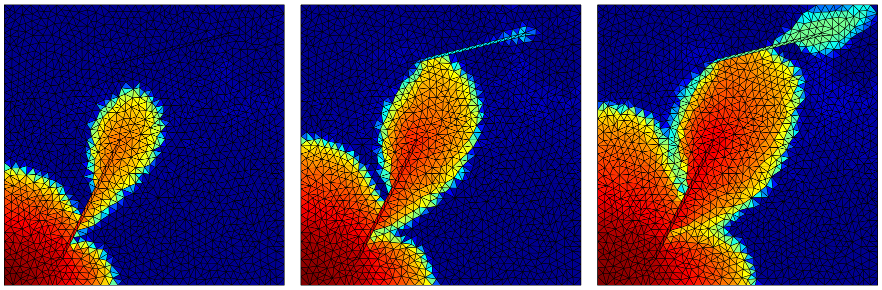

Figure 9. Water saturation in fractured medium at 0.15, 0.24 and 0.36 PVI computed with the single-point upwind method on a triangular grid.

on triangular elements. Each time step is computed using exactly the same sequential solution procedure that was applied for the Cartesian grids above.

In a forthcoming paper [15], we study stationary single-phase transport equations on the form

$$\mathbf{v} \cdot \nabla T = \phi(\mathbf{x}) \tag{15}$$

for naturally fractured reservoirs modelled using unstructured triangular grids with large aspect and cell-size ratios and demonstrate the feasibility of reordering in combination with dG(n) for $n \leq 5$. From a computational point of view, solving the semi-discrete equations of multiphase transport (3) is simpler than solving the time-of-flight equation (15). Whereas (3) has solutions in $[0, 1]$ and models a time-increment of a problem with finite speed of propagation, the solutions of (15) are integral solutions along streamlines, show large local variations that span several orders of magnitude and are therefore much harder to approximate using piecewise polynomial representations. Therefore, given the rather successful application of the dG/reordering method in [15], extending it to to the transport equations considered herein on fully unstructured tetrahedral grids is a question of a rather straightforward implementation. Similarly, it is in principle possible to extend the higher-order dG methods to unstructured hexahedral grids. However, here the implementation is a bit more involved, because the corner-point format allows various geometrical and topological degeneracies that need to be accounted for, such as non-matching interfaces arising near faults.

4.2. **Three-Phase Flow.** To demonstrate that the methodology also can be applied to more complex physical processes we next consider a water-alternating-gas (WAG) injection modelled by the hyperbolic three-phase system introduced by Juanes and Patzek [20]. For simplicity, we assume (somewhat nonphysically in the presence of gas) that the system is incompressible. Then the flow model can be written as (13), with s denoting a vector with first component representing the water saturation s_w and the second component the gas saturation s_g. The mobilities are modelled by

$$\begin{aligned}\lambda_w(s_w) &= (a_w s_w + (1-a_w)s_w^2)/\mu_w,\\ \lambda_g(s_g) &= (a_g s_g + (1-a_g)s_g^2)/\mu_g,\\ \lambda_o(s_w,s_g) &= (1-s_w-s_g)(1-s_w)(1-s_g)/\mu_o,\\ \lambda_T &= \lambda_w+\lambda_g+\lambda_o,\end{aligned}$$

where $a_w = 0$, $a_g = 0.1$, $\mu_w = 0.35$, $\mu_g = 0.012$ and $\mu_o = 0.8$. The two components of the flux function are λ_w/λ_T and λ_g/λ_T. This system is strictly hyperbolic except for the single umbilical point of 100% gas saturation, i.e., $s = (0,1)^T$, where the eigenvalues coincide [20]. Thus, all characteristics are positive and we may apply the upwind flux as before. In the remaining test cases we have used $M = 0.05$ in (12).

Case 7. We consider a three-phase WAG injection into the two-dimensional reservoir from Case 2 with permeability data taken from the sixth layer of the second SPE 10 model. The WAG injection sequence consists of 0.1 pore volumes of water, followed by 0.1 PVI gas, and then again 0.1 PVI water, etc. (To avoid the umbilical point, we inject a wet gas consisting of a mixture of 95% gas and 5% oil rather than a pure gas). In the operator splitting, we update the pressure at the end of each step in the injection cycle. As in Case 2, we use a CFL number of 10 in the saturation solver.

Figure 10 shows saturation profiles at the end of the first three steps in the cycle, i.e., at times $t = 0.1$, 0.2, and 0.3 PVI. Because gas has a much higher mobility than water, the gas injected in the second step will flow fast through the system and break through in the producer earlier than the water. In the third step, all gas will be displaced by water inside the water finger. As a result, a large amount of gas is produced before water breaks through. Again, the higher accuracy of the second-order scheme yields solutions with a much sharper resolution of the thin finger near the production well. From the plot, it may look like the gas saturation has oscillations seen as a number of small 'islands' in the profiles. A careful examination showed that are partially plotting artifacts created by the contouring algorithm (at $t = 0.2$) and partially small pockets of gas that have been displaced into low-permeable regions and encapsulated by the waterfront.

4.3. **Two-Phase, Three-Component Flow.**

Since the early 1970's, it has been recognised that many discrete methods for computing miscible fluid flow give qualitatively different results when one changes the orientation of the spatial grid with respect to the geometry of the physical flow.

In the last example we repeat a numerical experiment from [19] describing miscible injection of solvent in a quarter five-spot with unit porosity and permeability. In this model, the solvent is assumed to dissolve readily in the oil phase, while the water and oil phases are immiscible. The model can be

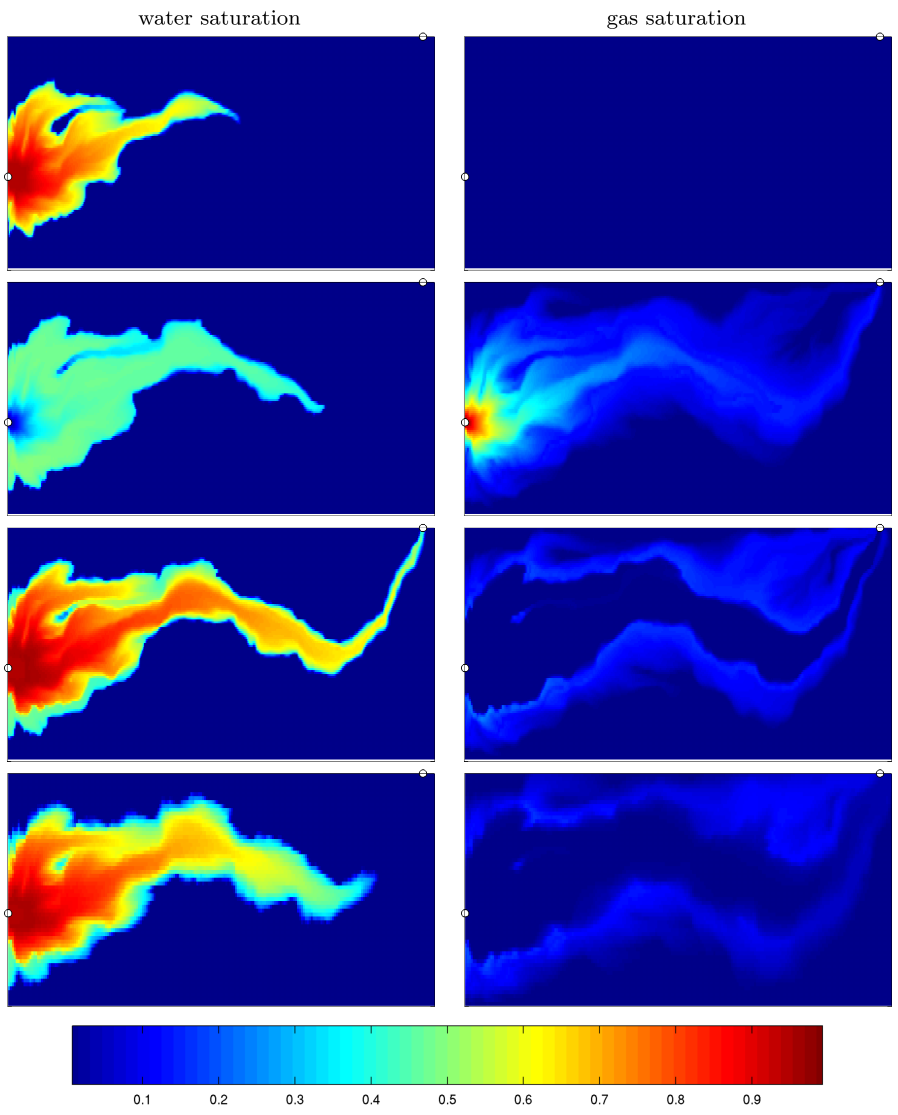

FIGURE 10. Water and gas saturation for the three-phase WAG injection.

written as

$$s_t + \mathbf{v} \cdot \nabla f(s) = 0,$$
$$c_t + \mathbf{v} \cdot \nabla \left(\frac{1 - f(s)}{1 - s} c \right) = 0,$$

where s and c are the water and solvent saturations, respectively. See [19] for more details. The flux function f is given by the ratio λ_w/λ_T and the

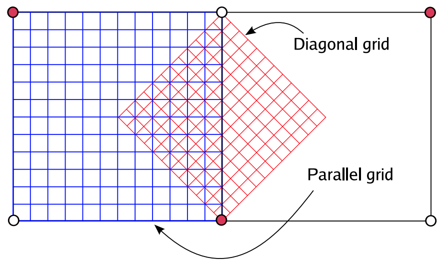

FIGURE 11. The parallel and diagonal grids used in Case 8. Producers are marked with circles and injectors with filled circles.

water and hydrocarbon mobilities are modelled by

$$\lambda_w(S) = \begin{cases} 0, & S < S_{wc} \\ \frac{1}{\mu_w}\left(\frac{S-S_{wc}}{1-S_{wc}}\right)^2, & \text{otherwise,} \end{cases}$$

$$\lambda_h(S) = \begin{cases} 0, & 1-S < S_{hc} \\ \frac{1}{\mu_h}\left(a_h \frac{1-S-S_{hc}}{1-S_{hc}} + (1-a_h)\big(\frac{1-S-S_{hc}}{1-S_{hc}}\big)^2\right), & \text{otherwise,} \end{cases}$$

with $S_{wc} = S_{hc} = 0.2$, $a_o = 0.1$, and $\mu_w = 1.0$. The hydrocarbon viscosity is modelled by a power law

$$\mu_h(\chi) = \left[\frac{1-\chi}{\mu_o^{1/4}} + \frac{\chi}{\mu_g^{1/4}}\right]^{-4}, \qquad \chi = \frac{C}{1-S},$$

with $\mu_o = 4.0$ and $\mu_g = 0.4$. Initially, the domain is filled with 30% water and 70% oil, and the injected fluid is pure solvent. Two fronts are present in the solution. The fastest front is a stable shock, while the slowest front is a contact discontinuity. This problem has an adverse mobility ratio, i.e., the mobility of the injected fluid is greater than the mobility of the resident fluid at the contact discontinuity. Problems with adverse mobility ratios are infamous for so-called grid-orientation effects, which means that the numerical solution depends strongly on the orientation and size of the grid. Without any diffusive terms, this problem is physically unstable. Grid-orientation effects are expected to vanish only if the physical diffusion dominates the numerical diffusion.

Case 8. To examine the grid-orientation effects of the discontinuous Galerkin schemes, we consider a five-spot problem (with zero physical diffusion) on two different grids that are commonly referred to as the diagonal and the parallel grid, see Figure 11. It is well-known that single-point upwind schemes exhibit strong grid-orientation effects. However, we expect that higher-order spatial discretisations are less sensitive to this effect. Therefore, the dG(0) method is likely to be inferior to any higher-order dG method for this problem.

Figure 12 shows the gas concentration (in which the front corresponds to a contact discontinuity) at time $t = 0.25$ PVI computed on the parallel and

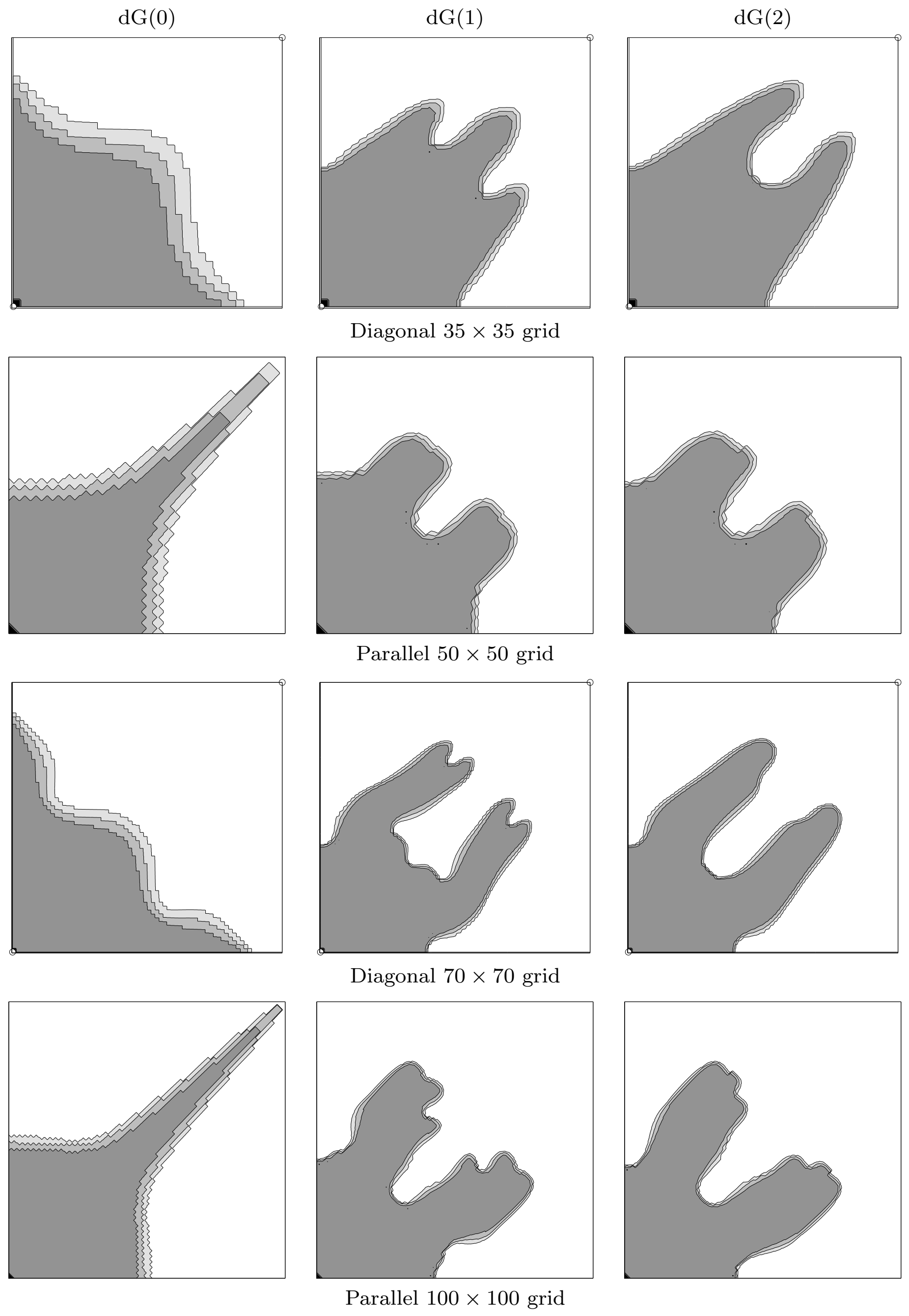

Figure 12. Gas concentration on the parallel and diagonal grids.

diagonal grids. In all plots we have used CFL = 10 in the saturation solver and 400 pressure updates. To illustrate the problem of grid-orientation effects, we have included results by the first-order single-point upwind method (i.e., dG(0)), which is clearly very sensitive to the grid orientation. Moreover, the effects of grid orientation are increased, rather than reduced, when the number of grid cells increases. The second and third-order dG methods produce solutions that are qualitatively similar on the parallel and diagonal grids. However, notice that the number of "wiggles" increases on the finer grids, due to a decreased amount of numerical dissipation. Altogether, the figure demonstrates the advantages of using a higher-order method for simulating miscible flow.

5. Final Remarks

In this paper we have introduced two new ideas for efficient computation of advection-dominated flow. First, we have introduced an implicit discontinuous Galerkin discretisation to derive schemes that allow high-order accuracy and compact stencils in combination with an ability to take large time steps. This yields very robust schemes. Since all degrees-of-freedom used in the higher-order stencils are located inside a single element (or grid cell), the method can easily be extended to include p-adaptivity (i.e., local adaptivity in the order of the method). For simple flows, like the standard Buckley–Leverett two-phase model, higher-order dG elements should only be used if one is able to take reasonably small time steps. If not, the increased spatial resolution of higher-order schemes will be completely overshadowed by the excessive numerical dissipation introduced when taking large implicit time steps. For more complex flows, e.g., miscible flow with adverse mobility ratios, the use of higher-order elements may stabilise the computations and diminish grid-orientation effects.

The second idea of the current paper is a reordering method that may be used to speed up the nonlinear solution procedure by decomposing the discrete global nonlinear system into a sequence of local nonlinear systems. By using this method, one may not only speed up the nonlinear iterations, but also avoid the assembly of global discretisation matrices and thereby increase the size of problems that one can solve. Here we have applied the reordering method to discontinuous Galerkin schemes on uniform Cartesian grids, on a industry-standard corner-point grid (logically Cartesian), and on an unstructured triangular grid. However, the method can (in principle) be applied to any discretisation method that in a discrete sense preserves the unidirectional dependency of the underlying differential equations. In particular, the reordering idea easily applies to general unstructured grids, as long as one is able to map cells and interface fluxes to vertices and edges in a directed graph. In fact, by viewing the (unstructured) grid as a graph and discretising using the single-point upwind method, all one needs to know is the edge connections and fluxes and the cell volumes and porosities in order to make a *very efficient* first-order implicit method. For a higher-order discontinuous Galerkin method one also needs quadrature rules and appropriate interpolation schemes for the velocities inside each cell.

Although the reordering method has great advantages with respect to computational efficiency, it also has an important limitation: the assumption of unidirectional flow. This assumption is violated by more general transport equations that include for instance gravitational or capillary forces. The natural way of extending the methodology presented here to more general transport equations is by operator splitting, as is also done for streamline methods, see [4, 17]. In this sense, the method is somewhat related to, and may have the same areas of applicability as streamline methods [21] in for instance direct simulation of models with multimillion cells and complex connectivities, history-matching, model screening and ranking, etc. Compared with streamlines, a reordered finite-volume scheme has a few advantages. Most notably, the discontinuous Galerkin schemes are conservative, and mass-balance errors are therefore not a problem. Furthermore, whereas a streamline simulator needs a method to distribute streamlines in the domain and methods for projecting saturations back and forth between streamlines and grid cells in physical space, we only need to find a reordering. This can be accomplished using well-known and efficient graph algorithms. Last but not least, this scheme can be built into conventional reservoir simulators that use a sequential splitting, and be used when appropriate.

6. Acknowledgement

We would like to thank Vegard Kippe for the streamline simulations in Case 3, and Ola I. Røe for the simulation of fractured media using triangular elements. This work was financed by the Research Council of Norway through grants 139144/431 and 158908/I30.

References

[1] I. Aavatsmark. An introduction to multipoint flux approximations for quadrilateral grids. *Comput. Geosci.*, 6(3-4):405–432, 2002.

[2] J. B. Bell, P. Colella, and J. A. Trangenstein. Higher order Godunov methods for general systems of hyperbolic conservation laws. *J. Comput. Phys.*, 82(2):362–397, 1989. ISSN 0021-9991.

[3] M. Blunt and B. Rubin. Implicit flux limiting schemes for petroleum reservoir simulation. *J. Comput. Phys.*, 102(1):194–210, 1992. ISSN 0021-9991.

[4] F. Bratvedt, T. Gimse, and C. Tegnander. Streamline computations for porous media flow including gravity. *Transp. Porous Media*, 25(1): 63–78, oct 1996. doi: 10.1007/BF00141262.

[5] F. Brezzi, K. Lipnikov, and V. Simoncini. A family of mimetic finite difference methods on polygonial and polyhedral meshes. *Math. Models Methods Appl. Sci.*, 15:1533–1553, 2005.

[6] F. Brezzi, K. Lipnikov, M. Shashkov, and V. Simoncini. A new discretization methodology for diffusion problems on generalized polyhedral meshes. *Comput. Methods Appl. Mech. Engrg.*, 196(37-40):3682–3692, 2007.

[7] M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reserv. Eval. Eng.*, 4(4):308–317, 2001. `http://www.spe.org/csp/`.

[8] B. Cockburn and C.-W. Shu. The Runge-Kutta local projection P^1-discontinuous-Galerkin finite element method for scalar conservation laws. *RAIRO Modél. Math. Anal. Numér.*, 25(3):337–361, 1991.

[9] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Math. Comp.*, 52(186):411–435, 1989.

[10] T. A. Davis and I. S. Duff. UMFPACK: An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM J. Matrix Anal. Appl.*, 18(1):140–158, 1997.

[11] J. E. Dennis, Jr., J. M. Martínez, and X. Zhang. Triangular decomposition methods for solving reducible nonlinear systems of equations. *SIAM J. Optim.*, 4(2):358–382, 1994.

[12] I. S. Duff and J. K. Reid. An implementation of tarjans algorithm for block triangularization of a matrix. *ACM Trans. Math. Software*, 4(2):137–147, 1978.

[13] L. J. Durlofsky. A triangle based mixed finite element – finite volume technique for modeling two phase flow through porous media. *J. Comput. Phys*, 105(2):252–266, 1993.

[14] M. G. Edwards. A higher-order Godunov scheme coupled with dynamic local grid refinement for flow in a porous medium. *Comput. Methods Appl. Mech. Engrg.*, 131(3-4):287–308, 1996. ISSN 0045-7825.

[15] B. Eikemo, K.-A. Lie, H. K. Dahle, and G. T. Eigestad. A discontinuous Galerkin method for transport in naturally fractured media. 2008.

[16] Y. Epshteyn and B. Rivière. Fully implicit discontinuous finite element methods for two-phase flow. *Appl. Numer. Math.*, 57(4):383–401, 2007. ISSN 0168-9274. doi: http://dx.doi.org/10.1016/j.apnum.2006.04.004.

[17] R. H. J. Gmelig Meyling. Numerical methods for solving the nonlinear hyperbolic equations of porous media flow. In *Third International Conference on Hyperbolic Problems, Vol. I, II (Uppsala, 1990)*, pages 503–517. Studentlitteratur, Lund, 1991.

[18] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Rev.*, 43(1):89–112 (electronic), 2001.

[19] R. Juanes and K.-A. Lie. Numerical modeling of multiphase first-contact miscible flows. Part 2. analytical Riemann solver. *Transp. Porous Media*, 72(1):97–120, 2008. doi: 10.1007/s11242-007-9139-y.

[20] R. Juanes and T. W. Patzek. Analytical solution to the riemann problem of three-phase flow in porous media. *Tranp. Porous Media*, 55(1):

47–70, 2004.

[21] M. J. King and A. Datta-Gupta. Streamline simulation: A current perspective. *In Situ*, 22(1):91–140, 1998.

[22] P. Lasaint and P.-A. Raviart. On a finite element method for solving the neutron transport equation. In *Mathematical aspects of finite elements in partial differential equations (Proc. Sympos., Math. Res. Center, Univ. Wisconsin, Madison, Wis., 1974)*, pages 89–123. Publication No. 33. Math. Res. Center, Univ. of Wisconsin-Madison, Academic Press, New York, 1974.

[23] J. R. Natvig and K.-A. Lie. On efficient implicit upwind schemes. In *11th European Conference on the Mathematics of Oil Recovery (ECMOR XI), Bergen, Norway.* EAGE, 8–11 September 2008.

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

[25] J. M. Nordbotten, I. Aavatsmark, and G. T. Eigestad. Monotonicity of control volume methods. *Numer. Math.*, 106(2):255–288, 2007. ISSN 0029-599X.

[26] I. Osako, A. Akhil Datta-Gupta, and M. J. King. Timestep selection during streamline simulation through transverse flux correction. *SPE J.*, 9(4):450–464, 2004.

[27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C.* Cambridge University Press, Cambridge, second edition, 1992. ISBN 0-521-43108-5. The art of scientific computing.

[28] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific. Laboratory, 1973.

[29] R. Sedgewick. *Algorithms in C.* Addison Wesley Professional, 1990.

[30] V. R. Stenerud, V. Kippe, K.-A. Lie, and A. Datta-Gupta. Multiscale-streamline simulation and dynamic data integration for high-resolution subsurface models. *submitted*, 2007.

[31] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[32] F. Wang and J. Xu. A crosswind block iterative method for convection-dominated problems. *SIAM J. Sci. Comput.*, 21(2):620–645 (electronic), 1999.

(Jostein Roald Natvig)
SINTEF ICT, Department of Applied Mathematics, P.O. Box 124, Blindern, N-0314 Oslo, Norway.
E-mail address: Jostein.R.Natvig@sintef.no

(Knut–Andreas Lie)
SINTEF ICT, Department of Applied Mathematics, P.O. Box 124, Blindern, N-0314 Oslo, Norway.
Also: Centre of Mathematics for Applications, University of Oslo, Norway.
E-mail address: Knut-Andreas.Lie@sintef.no