The DGIMPES Model in IPARS: Discontinuous Galerkin for Two-Phase Flow Integrated in a Reservoir Simulator Framework

Béatrice Rivière*

Abstract

This work is a documentation of the formulation and implementation of a Discontinuous Galerkin (DG) discretization of the two-phase flow problem in the Integrated Parallel Accurate Reservoir Simulator (IPARS) framework. An user's guide is also included.

*The Center for Subsurface Modeling (postdoctoral fellow 2000-2002), Texas Institute for Computational and Applied Mathematics, The University of Texas, Austin TX 78712 (riviere@ticam.utexas.edu).

Contents

1 Introduction

With the more affordable computing power, oil engineers are willing nowadays to increase the complexity of reservoir simulations. Some of the difficulties include complex geometry, faults, channels and deviated wells. It is therefore important to develop discretization methods that approximate accurately physical quantities such as pressure, flow rates and mass balances on highly unstructured and non-conforming grids. The current industrial reservoir simulators support finite differences, which are quite popular and efficient on regular structured grids. But, these methods can lose their stability on unstructured meshes and are not ideal for complex geometries. Discontinuous finite element methods appear to be very promising in producing a more accurate reservoir simulation on very general unstructured reservoir grids, while still maintaining the conservation properties of the finite difference methods. DG methods have several appealing properties: they are element-wise conservative, they support local approximations of high order, they are robust and non oscillatory in the presence of high gradients and they are implementable on unstructured and non-matching grids. Since DG methods are higher order methods, they are more costly than standard cell-centered finite differences. Oil engineers are willing to "pay the price for a more accurate simulation since mistakes can be costly..." [24].

There are a variety of methods using discontinuous discrete spaces such as the Bassi and Rebay method [7] and the Local Discontinuous Galerkin (LDG) [8, 3] method, the Oden, Babuška and Baumann method [15], the interior penalty Galerkin methods [30], and the NIPG methods [26, 25, 27, 11]. In Arnold, Brezzi, Cockburn and Marini [5] a general framework of these methods is presented. In this work, we present a formulation based on the NIPG method.

The two-phase flow problem is mathematically modeled by a system of non-linear partial differential equations. The phases considered here are a wetting phase (such as water) and a non-wetting phase (such as dense non-aqueous phase liquids). For each phase, the conservation of mass (or continuity equation) is required. The momentum equation for the two-phase flow yields a generalized Darcy's law. Under the assumption of incompressibility, a pressure-saturation formulation is derived, in which the non-linearity has been removed by time-lagging the coefficients. This sequential formulation is referred to as the IMPES formulation. There are other types of formulations. The reader can refer to [9, 17, 10] for a detailed description of multiphase

processes. We have chosen to discretize the IMPES formulation by the DG method, and the resulting model has been added to the Integrated Parallel Accurate Reservoir Simulator (IPARS-v2) framework under the model name DGIMPES.

The computing framework IPARS [1] has been developed by researchers at the Center for Subsurface Modeling over several years. The framework is written in a modular manner such that it addresses all of the important issues of a parallel simulator: variety of physical models, memory management, message passing, table lookup, solvers and I/O. More details can be found in the documentation available [2] on the framework. The description of the existing models and discretizations have been the object of several scientific papers. A non exhaustive list is [19, 31, 4, 14, 12, 17, 18, 20, 13, 16, 21]

The outline of this document is as follows: the model problem and some notation are introduced in Section 2. The implementation details are described in Section 4. Section 5 contains instructions on how to run IPARS with the DGIMPES model. Numerical examples are shown in Section 6. Finally, the last section presents some conclusions and possible future extensions of the actual code.

2 Model Problem

The reservoir (or porous medium) is denoted by Ω, a domain in $\mathbb{R}^d, d = 2, 3$. For the IMPES formulation, the primary variables are the pressure p_w and the saturation s_w of the wetting phase. It has been shown that this particular pressure-saturation formulation produces accurate results.

One of the advantage from a numerical point of view of the IMPES formulation lies in the fact that the difficulty arising from the nonlinearity is removed by time-lagging the coefficients. Let $0 = t_0 < t^1 < \cdots < t^N = T$ be a subdivision of the time interval $(0, T)$. For any function v that depends on time and space, we introduce the notation $v^i = v^i(\cdot) = v(t^i, \cdot)$ for $i = 0, \ldots, N$. Define the time step $\Delta t^i = t^{i+1} - t^i$.

The IMPES formulation is as follows: given (p_w^i, s_w^i), find (p_w^{i+1}, s_w^{i+1}) such

that

$$-\nabla \cdot (\boldsymbol{K}\lambda_t^i \nabla p_w^{i+1}) = -\nabla \cdot (\boldsymbol{K}\lambda_n^i |p_c'(s_w^i)| \nabla s_w^i) \\ -\nabla \cdot (\boldsymbol{K}(\lambda_n^i \rho_n + \lambda_w^i \rho_w) G \nabla D), \tag{1}$$

$$\frac{\phi}{\Delta t^{i+1}} s_w^{i+1} - \nabla \cdot (\boldsymbol{K} \frac{\lambda_n^i \lambda_w^i}{\lambda_t^i} |p_c'(s_w^i)| \nabla s_w^{i+1}) = \frac{\phi}{\Delta t^{i+1}} s_w^i \\ -\nabla \cdot (\frac{\lambda_w^i}{\lambda_t^i} \mathbf{u}_t^i) - \nabla \cdot (\boldsymbol{K} \frac{\lambda_o^i \lambda_w^i}{\lambda_t^i} (\rho_w - \rho_n) G \nabla D). \tag{2}$$

The derivation of the IMPES equations is recalled in the Appendix. The coefficients in (1) and (2) are defined below:

- $\boldsymbol{K}$ is the permeability tensor, that is symmetric positive definite and spatially dependent.

- $\lambda_t = \lambda_n + \lambda_w$ is the total mobility, that is the sum of the mobility of the non-wetting phase and the mobility of the wetting phase. Mobilities are functions that depend on the wetting phase saturation s_w in a non-linear fashion. $\lambda_i = k_i/\mu_i$, with k_i the relative permeability of phase i and μ_i the viscosity of phase i.

- $p_c = p_n - p_w$ is the capillary pressure, that is the difference of the pressures in the two phases. It is assumed to depend on the wetting phase saturation in a non-linear fashion. We have approximated the gradient of p_c at time t^i by $\nabla p_c(s_w^i) \approxeq p_c'(s_w^i) \nabla s_w^i$ and made the assumption that the capillary pressure is always decreasing.

- ρ_n, ρ_w are the phase densities. In the case of incompressible fluids, ρ_n and ρ_w are constant and equal to the reference densities $\rho_{n,ref}, \rho_{w.ref}$. In the slight compressible case, $\rho_\delta = \rho_{\delta,ref} e^{C_\delta P_w}$, for each phase $\delta = n, w$. The constants C_δ are the compressibility constants.

- G, D, ϕ are the gravity constant (9.81 SI), depth vector and the porosity constant.

- $\boldsymbol{u}_t = \boldsymbol{u}_n + \boldsymbol{u}_w$ is the total velocity, that is the sum of the two phases velocities. Each phase velocity is given as:

$$\boldsymbol{u}_\delta = -\boldsymbol{K}\lambda_\delta(\nabla p_\delta - \rho_\delta G \nabla D), \quad \delta = n, w.$$

Let us decompose the boundary of the porous medium $\partial\Omega$ first into a Neumann and a Dirichlet parts, then into an inflow and outflow parts:

$$\partial\Omega = \Gamma_N \cup \Gamma_D = \Gamma_{\text{in}} \cup \Gamma_{\text{out}}, \quad \Gamma_N \cap \Gamma_D = \Gamma_{\text{in}} \cap \Gamma_{\text{out}} = \emptyset,$$

Let $\boldsymbol{n}$ denotes the outward normal to $\partial\Omega$. The definition of the inflow boundary is $\Gamma_{\text{in}} = \{x \in \partial\Omega : \boldsymbol{u}_t \cdot \boldsymbol{n} < 0\}$. The boundary conditions associated to the IMPES formulation (1), (2) are

$$p_w^{i+1} = p_{\text{dir}} \quad \text{on} \quad \Gamma_D \tag{3}$$

$$\boldsymbol{u}_t^i \cdot \boldsymbol{n} = 0 \quad \text{on} \quad \Gamma_N, \tag{4}$$

$$(s_w^{i+1}\boldsymbol{u}_t^i - \boldsymbol{K}\frac{\lambda_n^i \lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla s_w^{i+1}) \cdot \boldsymbol{n} = s_{\text{in}}\boldsymbol{u}_t^i \cdot \boldsymbol{n} \quad \text{on} \quad \Gamma_{\text{in}}, \tag{5}$$

$$(-\boldsymbol{K}\frac{\lambda_n^i \lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla s_w^{i+1}) \cdot \boldsymbol{n} = 0 \quad \text{on} \quad \Gamma_{\text{out}}. \tag{6}$$

3 Scheme

In this section, we first establish some notation for the spatial discretization and we present our numerical scheme. Let $\mathcal{E}_h = \{E\}_E$ be a non degenerate subdivision of Ω, made of triangles or quadrilaterals in 2D and tetrahedra or prisms in 3D. Let h be the maximum diameter of the elements. Let Γ_h be the union of the open sets that coincide with interior edges (or interior faces) of elements of $\mathcal{E}_h$. Let e denote a segment of Γ_h shared by two triangles E^k and E^l of $\mathcal{E}_h$; we associate with e, once and for all, a unit normal vector $\boldsymbol{n}_e$ directed from E^k to E^l and we define formally the jump and average of a function ψ on e by:

$$[\psi] = (\psi|_{E^k})|_e - (\psi|_{E^l})|_e, \quad \{\psi\} = \frac{1}{2}(\psi|_{E^k})|_e + \frac{1}{2}(\psi|_{E^l})|_e.$$

If e is adjacent to $\partial\Omega$, then the jump and the average of ψ on e coincide with the trace of ψ on e and the normal vector $\boldsymbol{n}_e$ coincides with the outward normal $\boldsymbol{n}$.

For each integer r, we associate a finite element subspace of discontinuous piecewise polynomials:

$$\mathcal{D}_r(\mathcal{E}_h) = \{v : v|_E \in P_r(E) \quad \forall E \in \mathcal{E}_h\},$$

where $P_r(E)$ is a discrete space containing the set of polynomials of total degree less than or equal to r on E. We will approximate the wetting phase pressure and saturation by discontinuous polynomials of different orders. Let r_p and r_s be the degrees of polynomials for the pressure and saturation respectively.

We now introduce the following interior and boundary penalty term, for any integer r:

$$J^{\sigma,r}_{0,\Gamma_h}(\phi,\psi) = \sum_{e\in\Gamma_h} \frac{r\sigma_e}{|e|^\beta}\int_e [\phi][\psi], \quad J^{\sigma,r}_{0,\Gamma_D}(\phi,\psi) = \sum_{e\in\Gamma_D} \frac{r\sigma_e}{|e|^\beta}\int_e \phi\psi,$$

where σ is a discrete positive function that takes the constant value σ_e on the edge or face e, and is bounded below by $\sigma_0 \geq 0$, above by σ_m, $|e|$ denotes the measure of e and $\beta = (d-1)^{-1}$ is a real number.

We now derive the variational formulation for the IMPES problem, by considering the pressure equation (1) and the saturation equation (2) separately.

3.1 The pressure equation

Let us define, for $i = 0, \ldots, N$ the vector

$$\boldsymbol{\chi}^i = \boldsymbol{K}\lambda_n^i |p_c'(s_w^i)|\nabla s_w^i + \boldsymbol{K}(\lambda_n^i\rho_n + \lambda_w^i\rho_w)G\nabla D.$$

Then, (1) can be rewritten as:

$$-\nabla\cdot(\boldsymbol{K}\lambda_t^i\nabla p_w^{i+1}) = -\nabla\cdot\boldsymbol{\chi}^i. \tag{7}$$

Multiplying (7) by a test function $v \in \mathcal{D}_{r_p}$, and integrating by parts on one element E yields:

$$\int_E \boldsymbol{K}\lambda_t^i\nabla p_w^{i+1}\cdot\nabla v - \int_{\partial E}(\boldsymbol{K}\lambda_t^i\nabla p_w^{i+1}\cdot\boldsymbol{n}_E)v = \int_E \boldsymbol{\chi}^i\cdot\nabla v - \int_{\partial E}\boldsymbol{\chi}^i\cdot\boldsymbol{n}_E v,$$

where $\boldsymbol{n}_E$ is the outward normal to E. Summing over all the elements in $\mathcal{E}_h$ and using the fact that p_w and $\boldsymbol{\chi}$ are smooth enough, namely $[p_w^{i+1}] = 0$, $[\boldsymbol{K}\nabla p_w^{i+1}\cdot\boldsymbol{n}_e] = 0$ and $[\boldsymbol{\chi}^i\cdot\boldsymbol{n}_e] = 0$, we have

$$\sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{K}\lambda_t^i\nabla p_w^{i+1}\cdot\nabla v - \sum_{e\in\Gamma_h\cup\partial\Omega}\int_e\{\boldsymbol{K}\lambda_t^i\nabla p_w^{i+1}\cdot\boldsymbol{n}_e\}[v]$$
$$+\sum_{e\in\Gamma_h}\int_e\{\boldsymbol{K}\lambda_t^i\nabla v\cdot\boldsymbol{n}_e\}[p_w^{i+1}] = \sum_{E\in\mathcal{E}_h}\int_E\boldsymbol{\chi}^i\cdot\nabla v - \sum_{e\in\Gamma_h\cup\partial\Omega}\int_e\boldsymbol{\chi}^i\cdot\boldsymbol{n}_e[v]$$

Making use of the boundary conditions (3) and (4), we obtain

$$\sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{K}\lambda_t^n\nabla p_w^{i+1}\cdot\nabla v - \sum_{e\in\Gamma_h\cup\Gamma_D}\int_e\{\boldsymbol{K}\lambda_t^i\nabla p_w^{i+1}\cdot\boldsymbol{n}_e\}[v]$$
$$+\sum_{e\in\Gamma_h\cup\Gamma_D}\int_e\{\boldsymbol{K}\lambda_t^i\nabla v\cdot\boldsymbol{n}_e\}[p_w^{i+1}] = \sum_{E\in\mathcal{E}_h}\int_E\boldsymbol{\chi}^i\cdot\nabla v$$
$$-\sum_{e\in\Gamma_h\cup\Gamma_D}\int_e\boldsymbol{\chi}^i\cdot\boldsymbol{n}_e[v]+\sum_{e\in\Gamma_D}\int_e(\boldsymbol{K}\lambda_t^i\nabla v\cdot\boldsymbol{n})p_{\text{dir}} \tag{8}$$

3.2 The saturation equation

Let us define, for $i=0,\ldots,N$, the vector:

$$\boldsymbol{\zeta}^i = \frac{\lambda_w^i}{\lambda_t^i}\boldsymbol{u}_t^i + \boldsymbol{K}\frac{\lambda_n^i\lambda_w^i}{\lambda_t^i}(\rho_w-\rho_n)G\nabla D.$$

Then, (2) can be rewritten as:

$$\frac{\phi}{\Delta t^{i+1}}s_w^{i+1} - \nabla\cdot(\boldsymbol{K}\frac{\lambda_o^n\lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla s_w^{i+1}) = \frac{\phi}{\Delta t^{i+1}}s_w^i - \nabla\cdot\boldsymbol{\zeta}^i. \tag{9}$$

Similarly as for the pressure equation, we multiply by a test function $z\in\mathcal{D}_{r_s}$ over one element in $\mathcal{E}_h$, sum over all elements, and use the regularity of s_w^{i+1} and $\boldsymbol{\zeta}^i$. We finally obtain:

$$\int_\Omega\frac{\phi}{\Delta t^{i+1}}s_w^{i+1}z + \sum_{E\in\mathcal{E}_h}\int_E\boldsymbol{K}\frac{\lambda_n^i\lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla s_w^{i+1}\nabla z$$
$$-\sum_{e\in\Gamma_h\cup\partial\Omega}\int_e\{\boldsymbol{K}\frac{\lambda_n^i\lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla s_w^{i+1}\cdot\boldsymbol{n}_e\}[z]$$
$$+\sum_{e\in\Gamma_h}\int_e\{\boldsymbol{K}\frac{\lambda_n^i\lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla z\cdot\boldsymbol{n}_e\}[s_w^{i+1}]$$
$$=\int_\Omega\frac{\phi}{\Delta t^{i+1}}s_w^i z + \sum_{E\in\mathcal{E}_h}\int_E\boldsymbol{\zeta}^i\cdot\nabla z - \sum_{e\in\Gamma_h\cup\partial\Omega}\int_e\boldsymbol{\zeta}^i\cdot\boldsymbol{n}_e[z]$$

Making use of the boundary conditions (5), (6) and the continuity of pressure, we have:

$$\begin{aligned}\int_\Omega \frac{\phi}{\Delta t^{i+1}} s_w^{i+1} z + \sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{K}\frac{\lambda_n^i\lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla s_w^{i+1}\nabla z - \sum_{e\in\Gamma_{\rm in}}\int_e s_w^{i+1}\boldsymbol{u}_t^i\cdot\boldsymbol{n}_e z \\ - \sum_{e\in\Gamma_h}\int_e \{\boldsymbol{K}\frac{\lambda_n^i\lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla s_w^{i+1}\cdot\boldsymbol{n}_e\}[z] \\ + \sum_{e\in\Gamma_h}\int_e \{\boldsymbol{K}\frac{\lambda_n^i\lambda_w^i}{\lambda_t^i}|p_c'(s_w^i)|\nabla z\cdot\boldsymbol{n}_e\}[s_w^{i+1}] \\ = \int_\Omega \frac{\phi}{\Delta t^{i+1}} s_w^i z + \sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{\zeta}^i\cdot\nabla z - \sum_{e\in\Gamma_h\cup\partial\Omega}\int_e \boldsymbol{\zeta}^i\cdot\boldsymbol{n}_e[z] \\ - \sum_{e\in\Gamma_h\cup\Gamma_D}\int_e \frac{\lambda_w^i}{\lambda_t^i}\{\boldsymbol{K}\lambda_t^i\nabla z\cdot\boldsymbol{n}_e\}[P_w^i] - \sum_{e\in\Gamma_{\rm in}}\int_e s_{\rm in}\boldsymbol{u}_t^i\cdot\boldsymbol{n}_e z. \qquad (10)\end{aligned}$$

3.3 The discontinuous Galerkin approximations

Based on (8), (10) and the fact that the penalty terms vanish for the true solutions, we formulate the following discrete scheme:
given $(P_w^i, S_w^i)\in\mathcal{D}_{r_p}\times\mathcal{D}_{r_s}$, find $(P_w^{i+1}, S_w^{i+1})\in\mathcal{D}_{r_p}\times\mathcal{D}_{r_s}$ such that for all $(v,w)\in\mathcal{D}_{r_p}\times\mathcal{D}_{r_s}$:

$$\begin{aligned}\sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{K}\lambda_t(S_w^i)\nabla P_w^{i+1}\cdot\nabla v - \sum_{e\in\Gamma_h\cup\Gamma_D}\int_e \{\boldsymbol{K}\lambda_t(S_w^i)\nabla P_w^{i+1}\cdot\boldsymbol{n}_e\}[v] \\ + \sum_{e\in\Gamma_h\cup\Gamma_D}\int_e \{\boldsymbol{K}\lambda_t(S_w^i)\nabla v\cdot\boldsymbol{n}_e\}[P_w^{i+1}] + J_{0,\Gamma_h\cup\Gamma_D}^{\sigma,r_p}(P_w^{i+1}, v) = \sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{\chi}_h^i\cdot\nabla v \\ - \sum_{e\in\Gamma_h\cup\partial\Gamma_D}\int_e \tilde{\boldsymbol{\chi}}_h^i\cdot\boldsymbol{n}_e[v] + \sum_{e\in\Gamma_D}\int_e (\boldsymbol{K}\lambda_t(S_w^i)\nabla v\cdot\boldsymbol{n})p_{\rm dir} + J_{0,\Gamma_D}^{\sigma,r_p}(p_{\rm dir}, v) \qquad (11)\end{aligned}$$

$$\int_\Omega \frac{\phi}{\Delta t^{i+1}} S_w^{i+1} z + \sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{K}\frac{\lambda_n(S_w^i)\lambda_w(S_w^n)}{\lambda_t(S_w^n)}|p_c'(S_w^i)|\nabla S_w^{i+1}\nabla z$$

$$-\sum_{e\in\Gamma_{\rm in}}\int_e S_w^{i+1}\boldsymbol{U}_t^i\cdot\boldsymbol{n}_e - \sum_{e\in\Gamma_h}\int_e \{\boldsymbol{K}\frac{\lambda_n(S_w^i)\lambda_w(S_w^i)}{\lambda_t(S_w^i)}|p_c'(S_w^i)|\nabla S_w^{i+1}\cdot\boldsymbol{n}_e\}[z]$$

$$+\sum_{e\in\Gamma_h}\int_e \{\boldsymbol{K}\frac{\lambda_n(S_w^i)\lambda_w(S_w^i)}{\lambda_t(S_w^i)}|p_c'(S_w^i)|\nabla z\cdot\boldsymbol{n}_e\}[S_w^{i+1}] + J_{0,\Gamma_h}^{\sigma,r_s}(S_w^{i+1},z)$$

$$=\int_\Omega \frac{\phi}{\Delta t^{i+1}} S_w^i z + \sum_{E\in\mathcal{E}_h}\int_E \boldsymbol{\zeta}_h^i\cdot\nabla z - \sum_{e\in\Gamma_h\cup\Gamma_D}\int_e \tilde{\boldsymbol{\zeta}}_h^i\cdot\boldsymbol{n}_e[z]$$

$$-\sum_{e\in\Gamma_h\cup\Gamma_D}\int_e \frac{\lambda_w^i}{\lambda_t^i}\{\boldsymbol{K}\lambda_t^i\nabla z\cdot\boldsymbol{n}_e\}[P_w^i] - \sum_{e\in\Gamma_{\rm in}}\int_e s_{\rm in}\boldsymbol{U}_t^i\cdot\boldsymbol{n}_e, \qquad (12)$$

where $\boldsymbol{U}_t^i, \boldsymbol{\zeta}_h^i$ and $\boldsymbol{\chi}_h^i$ are the approximates of $\boldsymbol{u}_t^i, \boldsymbol{\zeta}^i$ and $\boldsymbol{\chi}^i$.

$$\begin{aligned}
\boldsymbol{U}_t^i &= -\boldsymbol{K}\lambda_w(S_w^i)(\nabla P_w^i - \rho_w G\nabla D)\\
&\quad -\boldsymbol{K}\lambda_n(S_w^i)(p_c'(S_w^i)\nabla S_w^i + \nabla P_w^i - \rho_n G\nabla D)\\
\boldsymbol{\chi}_h^i &= \boldsymbol{K}\lambda_n(S_w^i)|p_c'(S_w^i)|\nabla S_w^i + \boldsymbol{K}(\lambda_n(S_w^i)\rho_n + \lambda_w(S_w^i)\rho_w)G\nabla D\\
\boldsymbol{\zeta}_h^i &= \frac{\lambda_w(S_w^i)}{\lambda_t(S_w^i)}\{\boldsymbol{U}_t^i\} + \boldsymbol{K}\frac{\lambda_n(S_w^i)\lambda_w(S_w^i)}{\lambda_t(S_w^i)}(\rho_w-\rho_n)G\nabla D.
\end{aligned}$$

The quantities $\tilde{\boldsymbol{\chi}}_h$ and $\tilde{\boldsymbol{\zeta}}_h$ are upwind values of $\boldsymbol{\chi}_h$ and $\boldsymbol{\zeta}_h$ with respect to the normal component of the total velocity $\{\boldsymbol{U}_t\}$.

$$\forall e = \partial E^k \cap \partial E^l, \quad \forall\psi, \quad \tilde{\psi} = \begin{cases} \psi|_{E^k} & \text{if} \quad \{\boldsymbol{U}_t\}\cdot\boldsymbol{n}_e \geq 0,\\ \psi|_{E^l} & \text{if} \quad \{\boldsymbol{U}_t\}\cdot\boldsymbol{n}_e < 0.\end{cases}$$

From the derivations in Sections 3.1 and 3.2, we obtain the consistency of the discontinuous Galerkin scheme, that is if (p_w, s_w) is a solution of (1), (2), then (p_w, s_w) is also a solution of (11), (12).

3.4 Mass balance

Let us fix an element E and a test function $v \in \mathcal{D}_{r_p}$ that vanishes outside of E. For simplicity, we assume that E is an interior element in Ω. The

pressure equation (11) becomes:

$$\int_E \boldsymbol{K}\lambda_t(S_w^i)\nabla P_w^{i+1}\cdot\nabla v - \int_{\partial E}\{\boldsymbol{K}\lambda_t(S_w^i)\nabla P_w^{i+1}\cdot\boldsymbol{n}_E\}v$$
$$+\int_{\partial E}\frac{1}{2}\boldsymbol{K}\lambda_t(S_w^i)\nabla v\cdot\boldsymbol{n}_E[P_w^{i+1}] = \int_E \boldsymbol{\chi}_h^i\cdot\nabla v - \int_{\partial E}\tilde{\boldsymbol{\chi}}_h^i\cdot\boldsymbol{n}_E v.$$

Now, if $v = 1$ on E, then the equation becomes:

$$\int_{\partial E}(\{-\boldsymbol{K}\lambda_t(S_w^i)\nabla P_w^{i+1}\} + \tilde{\boldsymbol{\chi}}_h^i)\cdot\boldsymbol{n}_E = 0,$$

which is the mass conservation of the DG scheme.

The standard mass balance equation for each phase δ on each element E with outward normal $\boldsymbol{n}_E$ is:

$$\int_E \frac{\phi N_\delta^{i+1} - \phi N_\delta^i}{\Delta t} + \int_{\partial E}\rho_\delta \boldsymbol{u}_\delta\cdot\boldsymbol{n}_E = 0.$$

Here, $N_\delta = \rho_\delta S_\delta$. Note that for the IMPES formulation, the mass balances for each phase are not satisfied. One way to have the correct mass balances is to post-process the saturations and obtain a new N_δ^{i+1} [28]. One can solve the following local problem on each element E:

$$\int_E \phi N_\delta^{i+1} z = \int_E \phi N_{\delta^i} z + \int_E \rho_\delta^{i+1}\boldsymbol{u}_\delta^{i+1}\cdot\nabla z - \int_{\partial E}\rho_\delta^{i+1}\boldsymbol{u}_\delta^{i+1}\cdot\boldsymbol{n}_E z, \quad \forall z\in\mathcal{D}_r(\mathcal{E}_h)_{r_S}(\mathcal{E}_h).$$

3.5 Initialization

We consider two cases: first the gravity term is not taken into account, second the gravity effects are included.

3.5.1 No gravity effects:

We assume that both pressure and saturation for each phase are constant in the reservoir. The oil pressure and water saturation are given by the parameters `DGPOINIT`, `DGSWINIT` in the input file. The initial water pressure is then computed using the capillary pressure:

$$P_w(t=0) = DGPOINIT - P_c(DGSWINIT).$$

Thus, the primary variables P_w, S_w are initialized with constant values in the whole reservoir.

3.5.2 With gravity effects:

The reservoir variables are initialized by assuming hydrostatic equilibrium conditions for water and oil pressure. The water and oil saturations are then obtained through the capillary pressure. The user specifies an initial oil pressure and water saturation at the bottom of the reservoir by the parameters `DGPOINIT`, `DGSWINIT`. The initial water pressure at the bottom of the reservoir is obtained via the capillary pressure. The phase pressures are then equal to the hydrostatic pressures:

$$P_w(z=0,t=0) = DGPOINIT - P_c(DGSWINIT),$$
$$P_w(z,t=0) = P_w(z=0,t=0) - \rho_w g z,$$
$$P_o(z,t=0) = DGPOINIT - \rho_o g z.$$

From the linear initial phase pressures, the water saturation is obtained by solving the nonlinear equation on each element E:

$$\int_E P_c(S_w) - \int_E (P_o - P_w) = 0.$$

Newton's method is used to solve this nonlinear equation, and the initial guess for the Newton iterates is given by the initial water saturation at the bottom of the reservoir. An example of initial water pressure and saturation is given in Fig. 1. In that example, the reservoir has three layers of elements in the vertical direction. The initial saturation is constant on each element but varies in the vertical direction.

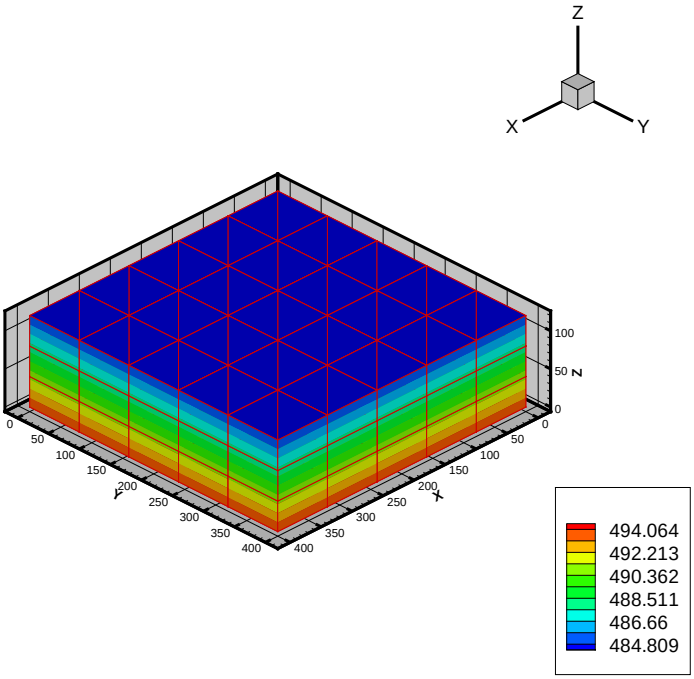
3.6 Well model and Dirichlet boundary conditions

3.6.1 No gravity effects:

In the case where the gravity is not included in the model, Dirichlet boundary conditions are simply set to constant pressures. The flow is then driven by the difference in the Dirichlet pressures. These pressures are set in the input file (see Section 5.1) by the parameters `PRESS_VAL1, PRESS_VAL2,...` No well models are allowed in that case.

3.6.2 With gravity effects:

In our fully three dimensional reservoir, gravity effects cannot be neglected. The flow is then driven by injection and production wells. In our well models,



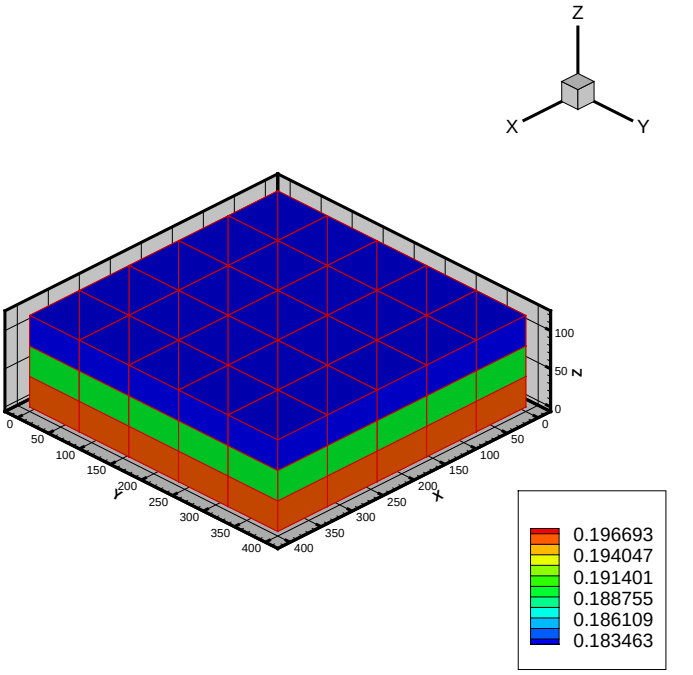


Figure 1: Water pressure (top) and saturation (bottom) after initialization of the reservoir.

bottom-hole pressures are specified via the input parameters `PRESS_VAL1`, `PRESS_VAL2,...` We then compute each phase pressure along the completion interval. At the water injection well, the pressure is specified at the well bore like a Dirichlet boundary condition:

$$P_w(x, y, z, t) = PRESS - VAL1 - \rho_w g z.$$

At the production well, the water pressure is defined as follows:

$$P_w(x, y, z, t) = PRESS - VAL2 - \rho^* g z,$$

where ρ^* is the fluid mixture density obtained by computing the phase flow rates through the well bore. We note that at the well, the capillary pressure is zero and we have:

$$\rho^* = \frac{q_w + q_o}{\frac{q_w}{\rho_w} + \frac{q_o}{\rho_o}},$$

$$q_w = \int_{\text{well surface}} \rho_w \lambda_w(S_w) \boldsymbol{K} \nabla P_w \cdot \boldsymbol{n},$$

$$q_o = \int_{\text{well surface}} \rho_o \lambda_o(S_w) \boldsymbol{K} \nabla P_w \cdot \boldsymbol{n}.$$

4 Integration in IPARS

In this section, we describe the technical implementation of the DG method by explaining the structure of the code. This section is essential for further developments of the model. If the reader is only interested in running the code, then the reader should directly go to Section 5.

4.1 Structure of code

There are several models contained in the IPARS framework. The internal number of the DGIMPES model is 14. The DG discretization of two-phase flow has been integrated into the main framework of IPARS, via the following interface files:

- `dgfluidsc.h`: definition of the fluid parameters (F77)
- `dgintf.c`: interface routines between the framework and the DG core routines (C)

- `dgisdat.df`: initialize the parameters from input file, call from framework file `input/idata.df` (F77)
- `dgivdat.df`: initialize physical variables such as viscosities, porosities, initial pressure and saturation, densities, permeabilities, call from framework file `drive/ipars.df` (F77)
- `dglookup.df`: set up the lookup tables, call to framework (F77)
- `dgquit.df`: free all memory that has been allocated for the DG data structure, call from framework file `drive/ipars.df` (F77)
- `dgstep.df`: solve one step of the DG discretization, the primary variables are updated with their new values, call from framework file `drive/ipars.df` (F77)
- `dgvisual.df`: write out tecplot output files, call from framework (F77)

The DG core routines are hidden in the subdirectory `dgimpes/dg3d/src`. In particular, since the DGIMPES model allows for fully unstructured meshes, the data structure and mesh initialization are included in the DG core routines.

4.2 The DG core routines

This directory is divided into two folders: `lib` and `src`. The latter contains all the DG core routines, organized in a modular fashion. The philosophy between the classification of the files is that one can change the data structure without having to rewrite the entire code, but only the directory (or the file) that deals directly with the data structure. All the core routines are written in C.

- `dgf_fem`: all the files related to finite element computations and data structures

  `dgff_data_structure.c`: routines that are directly aware of the data structure. This is the file to be modified if the data structure is changed.

  `dgff_fem_util.c`: functions that perform finite element calculations, such as computing the basis functions, the determinant...

`dgff_lin_solver.c`: interface to the dense LAPACK direct solver. This solver is not efficient at all, and should only be used to debug small problems.

- `dgm_mesh_mod`: all the files related to mesh refinement and derefinement: `dgmf_mesh_mod.c, dgmf_mmod_prism.c, dgmf_mmod_util.c, dgm_mesh_mod.h`. These files have not been tested thoroughly. These files are to be changed if one wants to add other types of elements (tetrahedra...).

- `dgu_utilities`: Utilities files that can be used anywhere. The Gauss integration points are listed in the file `ut_gauss.c`, and the other general routines are listed in `ut_util.c`.

- `dgb_solver_iter`: Directory containing the iterative solver files. The file `dgbf_iter_intf.c` assembles the local matrices. The files `it_block.c`, `it_block.h`, `it_gmres.c` are local to that directory.

- `dgi_io`: routines related to input and output.

  `dgif_data.c`: functions that allow for restart capabilities, this has not been tested.

  `dgif_data_jk.c`: functions that initialize the data structure from mesh input files.

  `dgif_tecplot.c`: write output files in a Tecplot format, that is compatible with the framework way of writing out data.

- `dgp_conv_diff`: main routines containing the DG discretization algorithm.

  `dgpf_stiff.c`: compute local element and face stiffness matrices.

  `dgpf_coeff.c`: compute the coefficients of the PDE.

  `dgpf_adapt.c`: this is where the computation of local error indicators goes. This has not been tested.

  `dgpf_post.c`: routines applying the slope limiters and computing the mass balance errors.

  `dgv_intf.h`: header file for global variables.

- `include`: header files for control parameters, data structures, definition of functions.

4.3 The DG data structure

The data structure that has been used in the code is defined in a modular manner. For each of the pressure and saturation equations, there is a pointer to a structure that consists of pointers to different structures, each of which characterizes definite properties of the problems. The data structure was first defined in the DG3D stand alone code developed by Banas (further details can be found in [6]) and based on the original 2D code by Rivière [23]. Modifications have been made for handling the two-phase flow applications.

```
typedef struct {
    dgft_ctrls  ctrl;   /* control parameters */
    dgft_linss  lins;   /* linear solver parameters */
    dgft_adpts  adpt;   /* adaptation parameters */
    dgft_meshs  *mesh;  /* pointer to mesh data structure */
    dgft_solut  *dofs;  /* pointer to dofs structure */
} dgft_discr;
```

The `dgft_meshs` structure is based on the concept of parent-child finite element data structure. Since the DG formulation involves integration on faces, we define specific face and edge structures. This type of data structure was first implemented in a 2D DG code solving the coupled flow and transport problem [23].

```
typedef struct {
    dgft_meshp  parm;   /* mesh parameters */
    dgft_nodes  *node;  /* pointer to array of nodes */
    dgft_edges  *edge;  /* pointer to array of edges */
    dgft_faces  *face;  /* pointer to array of faces */
    dgft_elems  *elem;  /* pointer to array of elements */
} dgft_meshs;

/* mesh parameters structure */
typedef struct {
    int    nrno;        /* number of active nodes */
    int    nmno;        /* maximal index of initiated node */
    int    mxno;        /* maximal available index of node */
    int    pfno;        /* pointer to first free node space*/
    int    nred;        /* number of active edges */
```

```
    int    nmed;        /* maximal index of initiated edge */
    int    mxed;        /* maximal available index of edge */
    int    pfed;        /* pointer to first free edge space*/
    int    nrfa;        /* number of active faces */
    int    nmfa;        /* maximal index of initiated face */
    int    mxfa;        /* maximal available index of face */
    int    pffa;        /* pointer to first free face space */
    int    nrel;        /* number of active elements */
    int    nmel;        /* maximal index of initiated element */
    int    mxel;        /* maximal available index of element */
    int    pfel;        /* pointer to first free element space */
} dgft_meshp;

/* node structure */
typedef struct {
    double x;   /* x-coor of the node  */
        /* (or -1e11 for free spaces ) */
    double y;   /* y-coor of the node */
                /* (or next free space number) */
    double z;   /* z-coor of the node */
} dgft_nodes;

/* edge structure */
typedef struct {
    int type;   /* 1  - active edge */
                /* <0 - inactive edge, number of attempted divisions */
                /* 0  - free space */
    int node[2];/* for active edges: node's numbers */
                /* for free spaces: node[0] - the next free space */
                /* for inactive edges: son's numbers */
} dgft_edges;

/* face structure */
typedef struct {
    int    type;        /* type and status of the face: */
                        /*  DGFP_TRIA - triangle, */
                        /*  DGFP_QUAD - quadrilateral */
                        /*  0 - free space */
```

```
                        /* >0 - active, not refined */
                        /* <0 - inactive, refined */
    int    bc;          /* bc flag for the face */
                        /*  Dirichlet (DGFP_BC_DIRI) boundary */
                        /*      0 < bc < 10 */
                        /*  Neumann (DGFP_BC_NEUM) boundary */
                        /*     10 < bc < 20 */
                        /*  Robin (DGFP_BC_MIXED) boundary */
                        /*     20 < bc < 30 */
                        /*  dirichlet pres. and outflow sat. */
                        /*     (DGFP_BC_OUTFLOW) boundary faces:  */
                        /*     30 < bc < 40 */
                        /* for INTERIOR (DGFP_INTERIOR) faces: */
                        /*  bc<=0 - shift between positions of nodes  */
                        /*      for the first and the second neighbor */
                        /* for free spaces: next free space's number */
    int edge[4];        /* edges' numbers - in proper order */
                        /*  0 -edge with the same orientation as face */
                        /*  <0 - edge with opposite orientation */
    int neig[2];        /* neighboring elements' global numbers */
                        /* (first neighbor has the same orientation */
                        /*  as the face and ordering of its nodes */
                        /*  defines ordering of nodes on the face) */
                        /*  >0 - equal size neighbor */
                        /*  -1 - indicates big neighbor */
                        /*  0 - boundary (always second neighbor) */
} dgft_faces;

/* element structure */
typedef struct {
    int    type;        /* DGFP_PRISM - prism */
                        /* 0 - free space (face and sons not allocated) */
                        /* >0 - active, not refined */
                        /* <0 - inactive, refined */
    int    mate;        /* material number */
                        /* for free space: number of the next free structure */
    int    fath;        /* parent's global number */
                        /* 0: if the element has no parent */
```

```
    int    refi;        /* type of the last refinement */
                        /* 0 - sons not allocated */
                        /* DGFP_REF_ISO - isotropic refinement */
    int    *face;       /* faces' global numbers (sign=orientation)*/
    int    *sons;       /* children */
} dgft_elems;
```

The solution vectors corresponding to the water pressure and saturation are stored in the solution structure for each element in the mesh. The current values correspond to `sol_2` and the new values are stored in `sol_1`.

```
typedef struct {
    int    pdeg;     /* PRISM - pdeg = 100*pdegz + pdegxy */
                     /* (pdeg = -1 - sol_x vectors not allocated) */
    double *sol_1;   /* pointer to array of DG solution dofs */
    double *sol_2;   /* pointer to array of DG solution dofs */
} dgft_solut;

/* structure with control parameters */
typedef struct {
  int     nr_sol;       /* number of solution vectors for each element */
  int     nr_mat;       /* number of materials (0 - no materials data)*/
  int     base;         /* type of shape functions:  */
                        /*     DGFP_TENSOR - tensor product */
                        /*     DGFP_COMPLETE - complete polynomials */
                        /* the order of the first four dofs  */
                        /* is: 1,x,y,z  */
  int     slope;        /* slope limiter: */
                        /*      1 - active */
                        /*      0 - not active */
  double penal;         /* penal - penalty parameter  */
  double coeff3;        /* coeff3 - slope limiting parameter  */
  double **bc_val;      /* array of pointers to values of boundary c. */
  double **ic_val;      /* array of pointers to values of initial c. */
} dgft_ctrls;

/* structure with linear solver control parameters */
typedef struct {
  int    type;          /* method identifier */
```

```
                        /*       0 - direct solver (?) */
                        /*       1 - preconditioned GMRES */
                        /*       2 - multigrid preconditioned  GMRES */
                        /*      10 - standard iterations */
                        /*      20 - V-cycle multigrid */
  int    max_iter;      /* maximal iteration number */
  int    conv_type;     /* convergence criterion number */
                        /*      0 - relative to initial residual */
                        /*      1 - absolute residual */
                        /*      2 - relative to rhs */
  double conv_meas;     /* convergence measure */
  int    monitor;       /* monitoring level: */
                        /* (for the meaning see include/dgff_intf.h) */
                        /*   DGFP_SILENT      0  */
                        /*   DGFP_ERRORS      1  */
                        /*   DGFP_INFO        2  */
                        /*   DGFP_ALLINFO     3  */
} dgft_linss;

/* structure with adaptation parameters */
typedef struct {
  int    type;          /* strategy number for adaptation */
  int    interval;      /* number of time steps between adaptations */
  int    maxgen;        /* maximum generation level for elements */
  int    maxgendiff;    /* maximum difference of generation levels */
                        /* for neighboring elements */
  double eps;           /* coefficient for choosing elements to adapt */
  double ratio;         /* ratio of errors for derefinements */
  int    monitor;       /* monitoring level: */
                        /* (for the meaning see include/dgff_intf.h) */
                        /*   DGFP_SILENT      0  */
                        /*   DGFP_ERRORS      1  */
                        /*   DGFP_INFO        2  */
                        /*   DGFP_ALLINFO     3  */
} dgft_adpts;
```

Full tensor capabilities: We assume that the permeability field is a symmetric tensor defined by six values. The global variable `dgpv_perm` is

initialized with the following input parameters: `dgpv_perm[0] = DGPERMXX`, `dgpv_perm[1] = DGPERMYY`, `dgpv_perm[2] = DGPERMZZ`, `dgpv_perm[3] = DGPERMXY`, `dgpv_perm[4] = DGPERMYZ`, `dgpv_perm[5] = DGPERMXZ`.

4.4 Time stepping

The user can set the time intervals and the final time for the simulation in the input file. The IPARS framework will compute the time steps accordingly. We allow the pressure time step to be a multiple of the saturation time step. This means that at each pressure time step, the pressure is computed once, and the saturation is then computed at different times. The user initializes the number of saturation steps per pressure step by the variable `DGNSSTEP`.

4.5 Slope Limiting

Discontinuous approximations yield overshoot and undershoot in the neighborhood of the front of the injected phase. Slope limiters are the appropriate tools for decreasing the local oscillations [8].

4.6 Solvers

For higher order methods, the choice of efficient solvers is important. In its current stage, direct and iterative solvers are implemented. The direct solver consists of a call to LAPACK procedures for dense Gaussian elimination. The iterative solver consists of a GMRES with multi-level preconditioning. Further details can be found in [6].

4.7 Units

The code assumes the following units for the physical quantities.

- Pressure in psi.
- Time in day.
- domain dimensions in feet.
- permeability in milliDarcys.
- viscosity in centipoise

- density in lb per bbl

Conversion constants are included in the code.

5 User's Guide

Note: *This section focuses on how to set up the right input files for the DGIMPES model. If one wants to know how to obtain and make the code IPARS, one should read the detailed documentation [2] that is available. The author assumes that the reader is familiar with the IPARS framework. If not, then the documentation refered above should be read first.*

Following the procedures described in [2], the user has created a a work directory. In this directory, the following files are needed:

- `ipars.siz` : this file is copied from the directory `/size/modular/`, and the user should modify it, in order to specify the model to be run.
- `Makefile`: the user should modify this file in order to specify which modular makefile is to be used. The original file is in `make/modular/unix.mak/Makefile`
- `dginput`: this is the main input file that will define the problem to be solved. The user sets the name of this file. In the section below, we will describe the important keywords for the dgimpes model.
- `IPARS.IN`: this allows a batch run of IPARS. This file contains the name of the input file, here `dginput`, and the name of the output file, also chosen by the user.
- `mesh_jk1.dat`: this file contains the mesh structure for the pressure. See explanation below.
- `mesh_jk2.dat`: this file contains the mesh structure for the saturation.

Note that the meshes for pressure and saturation may be different, However this capability has not been tested yet, and it is clear that some of the routines will have to be modified.

5.1 The mesh input files

Meshes consist of prismatic elements, that are generated from a 2D unstructured triangular mesh. An example of a mesh input file is the following

```
300000 1300000 1600000 600000
 25 0.0 20.0 1 11 11
 0.  0.  20.  0.  40.  0.  60.  0. 80.  0.
 0.  20.  20.  20.  40.  20.  60.  20. 80.  20.
 0.  40.  20.  40.  40.  40.  60.  40. 80.  40.
 0.  60.  20.  60.  40.  60.  60.  60. 80.  60.
 0.  80.  20.  80.  40.  80.  60.  80. 80.  80.
 32
 1 6 1 7
-x  2 9
 2 2 7 1
3 1 -x
 3 7 2 8
2 4 11
 4 3 8 2
5 3 -x
 5 8 3 9
4 6 13
 6 4 9 3
7 5 -x
 7 9 4 10
6 8 15
 8 5 10 4
-x 7 -x
 9 6 7 11
1 10 -x
 10 12 11 7
18 9 11
 11 7 8 12
3 12 10
 12 13 12 8
20 11 13
 13 8 9 13
5 14 12
```

```
 14 14 13 9
22 13 15
 15 9 10 14
7 16 14
 16 15 14 10
24 15 -x
 17 16 11 17
25 -x 18
 18 12 17 11
19 17 10
 19 17 12 18
18 20 27
 20 13 18 12
21 19 12
 21 18 13 19
20 22 29
 22 14 19 13
23 21 14
 23 19 14 20
22 24 31
 24 15 20 14
-x 23 16
 25 16 17 21
17 26 -x
 26 22 21 17
-x 25 27
 27 17 18 22
19 28 26
 28 23 22 18
-x 27 29
 29 18 19 23
21 30 28
 30 24 23 19
-x 29 31
 31 19 20 24
23 32 30
 32 25 24 20
-x 21 -x
```

The first line indicates the amount of memory allocated for some data structure.

The second line gives respectively the number of nodes in the 2D triangular mesh (that lies in the xy plane), the z coordinate of the bottom layer, the z coordinate of the top layer, the number of horizontal layers (here, we only have one layer), the type of boundary condition of the bottom layer and the type of boundary condition of the top layer.

From line 3 to line 7, the xy coordinates of the nodes are listed.

Line 8 contains the number of triangular elements.

The rest of the file lists the connectivity of the triangular elements and some information related to the edges of each triangle. For example, line 9 contains the number of the element, the global number of the three nodes N1, N2, N3, of that element. The following line gives for first for the edge between N1 and N2, the global number of the neighboring element, if that number is positive, or -(type of boundary condition) if that number is negative, i.e., the edge belongs to the boundary of the domain.

The boundary conditions allowed are the following. First, for the pressure mesh, if the face is of type Dirichlet, then $|x|$ should be set between 1 and 9; if the face is of type Neumann, then $|x|$ should be set between 11 and 19. Second, for the saturation mesh, if the face is of inflow type, then $|x|$ takes the values between 21 and 29, if the face is of no-flow (or Neumann) type then $|x|$ is set between 11 and 19. The outflow faces that correspond to a Dirichlet face for the pressure mesh, correspond to $|x|$ between 31 and 39.

The input file `dginput` contains the values associated to the types of boundary conditions.

5.2 The framework input file for DGIMPES

Below, an example of an input file `dginput` is given.

```
TITLE(2)="DG_3D IMPES TEST "

DESCRIPTION()=

"BLOCK   LENGTH (FT)   WIDTH (FT)   HEIGHT (FT)    SIZE     CORNER"
"DATE : 7/31/02"

DG_IMPES
```

```
TIMEEND = 250.0

$$$$$$$$$$$$$

$ I/O OPTIONS

OUTLEVEL = 2

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ FAULT BLOCK AND MESH DATA

BLOCKNAME(1) = "BLOCK1"
DOWN(1 TO 3,1) = 1 0 0
NX(1) = 1          NY(1) = 1           NZ(1) = 1
DX(,1) = 1.  DY(,1) = 1.  DZ(,1) = 1.
XYZ111(,1) = 0. 0. 0.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ IPARS data
$ INITIAL CONDITIONS

DGPOINIT = 500. DGSWINIT = 0.200000

PRESS_MESHTYP = 2
PRESS_DEG = 2
PRESS_BASE = 2
PRESS_PENAL = 0.0
PRESS_T_MONIT = 2
PRESS_A_TYPE = -1
PRESS_L_TYPE = 0
PRESS_L_MAX = 1000
PRESS_L_CONVTYP = 2
PRESS_L_COME = 1.E-12
PRESS_L_MONIT = 0

$BOUNDARY CONDITIONS AND WELLS
PRESS_BC = 3
PRESS_TYPE1 = 1
```

```
PRESS_TYPE2 = 2
PRESS_TYPE3 = 11
PRESS_VAL1 = 550.0
PRESS_VAL2 = 300.0
PRESS_VAL3 = 0.0

PRESS_OUTPUT = 1

SAT_MESHTYP = 2
SAT_DEG = 1
SAT_BASE = 2
SAT_SLOPE = 1
SAT_PENAL = 0.0
SAT_COEFF3 = 0.7
SAT_A_TYPE = -1
SAT_L_TYPE = 1
SAT_L_MAX = 100
SAT_L_CONVTYP = 2
SAT_L_COME = 1.E-14
SAT_L_MONIT = 0

$BOUNDARY CONDITIONS
SAT_BC = 3
SAT_TYPE1 = 21
SAT_TYPE2 = 11
SAT_TYPE3 = 31
SAT_VAL1 = 0.2
SAT_VAL2 = 0.0
SAT_VAL3 = 0.0

SAT_OUTPUT = 1

$ VISCOSITY

DGOILVIS = 2.0[cp]
DGWATVIS = 0.5[cp]

$ POROSITY
```

```
DGPOROSITY = 0.2

$DENSITY
DGOILDEN = 56.0
DGWATDEN = 62.34

$COMPRESSIBILITY
DGOILCOMP = 1.E-10
DGWATCOMP = 1.E-12

$NUMBER OF SATURATION STEPS PER PRESSURE STEP
DGNSSTEP = 4

$ Note that the line below is not used in DG code but is aksed in the
$ framework by GETROCK
POROSITY1() = .2

$ PERMEABILITIES

DGPERMXX = 436.0
DGPERMYY = 110.0
DGPERMZZ = 100.0
DGPERMXY = 147.0
DGPERMYZ = 0.0
DGPERMXZ = 147.0

$ Note that the line below is not used in DG code but is asked in the
XPERM1() = 0  YPERM1() = 0

$ the following tables will define ko, kw
KOSW(1) Block        $ OIL RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Linear
  Extrapolation Constant
  Constraint  0 At .8
  Derivative  0 At .8
  Constraint  1 At  0
  Nodes  .2  .58
```

```
  Data  0.  1. , .1  .67 , .2  .46 , .4  .2 , .6  .055 , .7  .015 , .8  0
EndBlock

KWSW(1) Block        $ WATER RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Linear
  Extrapolation Constant
  Constraint  0 At .15
  Derivative  0 At .15
  Constraint  1 At   1
  Nodes  .55  .7  .75
  Data  .15  0 , .3  .035 , .4  .085 , .6  .28 , .8  .776 , .9 .9, 1  1
EndBlock

$ WATER-OIL CAPILLARY PRESSURE - ROCK TYPE 1
PCOW(1) Block        $ WATER-OIL CAPILLARY PRESSURE - ROCK TYPE 1
  Interpolation Spline3
  Extrapolation Same
  Nodes  .25  .4  .7  .9
  Pole   .12
  Data
  .16   9.00 , .2   6.12 , .225 4.86 , .25  4.22 ,
  .275 3.78 , .325 3.20 ,
  .375 2.74 , .45  2.28 , .55  1.94 ,
  .65  1.74 , .75  1.61 , .85  1.54
 .925 1.44 , .95  1.37 , .975 1.14 , 1.0  .70
EndBlock

EndInitial

$tables output
$VIS_TABOUTTYPE = 1

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ TRANSIENT DATA INPUT BLOCKS

BeginTime    0.
DELTIM = 1.0  DTIMMUL = 1.0  DTIMMAX = 1.0  DTIMOUT = 1.0 DTIMMIN = 1.0
TIMOUT = 1.0
```

```
VISOUT = 0.0
DVISOUT = 1.0
VISFLAG = 3
VIS_FNAME = "RUN"
EndTime
```

Some parameters are defined by the framework, such as the relative permeabilities arrays, the capillary pressure array and the transient variables (DELTIM, TIMEND, ...) TIMEEND which characterizes the final time of the simulation. Here, we will only explain the parameters that are specific to DG-IMPES. Since the data structure for unstructured meshes is different from the data structure for hexaedral elements, we redefine physical properties, usually used in the framework, by adding the prefix DG to their names. For example, DGPOINIT, DGSWINIT, DGOILVIS, DGWATVIS, DGPOROSITY, DGOILDEN, DGWATDEN, DGOILCOMP, DGWATCOMP are physical scalar quantities, that are assumed to be constant throughout the simulation. The symmetric permeability field K, in this example, is assumed to be a full tensor, with uniform values throughout the domain: K_{xx} =DGPERMXX, K_{yy} =DGPERMYY, K_{zz} =DGPERMZZ, K_{xy} =DGPERMXY, K_{yz} =DGPERMYZ and K_{xz} =DGPERMXZ.

For solving both the pressure and the saturation equations, several keywords are necessary. Since we allow for different meshes, we need to define different parameters for both pressure and saturation meshes. We will below explain those parameters for the pressure mesh. Then, we describe the few parameters that only exist for the saturation case.

- PRESS_MESHTYP indicates the type of mesh. If the mesh input file contains a 2D triangular mesh, as the one explained above, then the mesh type is 2.

- PRESS_DEG indicates the degree of polynomial approximation. At least quadratic polynomials (PRESS_DEG=2) need to be used for the pressure. For the saturation, one can use either constant or linears. Higher order can also be used for the saturation, as long as the slope limiters are appropriately defined.

- PRESS_BASE is the type of basis functions used. If the tensor product functions are chosen for the prismatic elements: $(1, x, y, x^2, y^2, xy) \otimes (1, z, z^2)$, then the parameter is set to 1. If the complete set of basis functions is chosen $(1, x, y, z, xy, yz, xz, x^2, y^2, z^2)$, then the parameter is set to 2.

- PRESS_PENAL is the penalty parameter. We assume here that it is uniformly constant over the domain.

- PRESS_T_MONIT = 2 is a monitor flag, that prints out some information about the simulation time integration.

- PRESS_A_TYPE = -1 is the adaptivity parameter. If it is negative, then the mesh is uniformly refined once. If it is positive, then the mesh is not refined.

- The following parameters define the type of solver to be used. If PRESS_L_TYPE=0, then the direct LAPACK solver is used. If it is 1, then the iterative single level GMRES solver is used. If it is 2, then the multi-level GMRES solver is used. PRESS_L_MAX is the maximum number of iterations allowed. PRESS_L_CONVTYP is the type of convergence criterion to be used to stop the iteration. The value 0 corresponds to relative to initial residual, the value 1 corresponds to absolute residual and the value 2 corresponds to a stopping criterion relative to the right-hand side. PRESS_L_COME is the convergence tolerance. PRESS_L_MONIT is a monitoring flag. The value 0 corresponds to a silent run, the value 1 produces warning messages, the value 2 adds information on restart data, and the value 4 also produces information on iteration data.

- The boundary conditions are then prescribed using the following parameters. PRESS_BC indicates the number of different boundary condition on the boundary of the domain. The type of each boundary condition is given by PRESS_TYPEi, where $i = 1, \ldots,$PRESS_BC. The corresponding values are given by PRESS_VALi. In the example given above, there are two Dirichlet boundary conditions with the values 550 and 300 and also a zero Neumann boundary condition. Currently, if gravity is turned on, the type 1 corresponds to a water injection well, and the type 2 corresponds to a production well.

- PRESS_OUTPUT indicates the output step for writing out Tecplot files.

There are three parameters specific to the saturation problem:

- SAT_SLOPE is a flag that specifies if a slope limiter is used (1) or not (0),

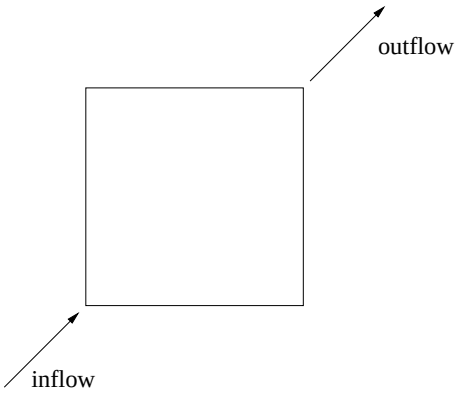


Figure 2: Quarter-five spot configuration.

- `SAT_LIMIT` is a parameter for the amount of slope limiting that is wanted. (see section on slope limiters).
- `DGNSSTEP` is the number of saturation steps per pressure step. The default value is 1.

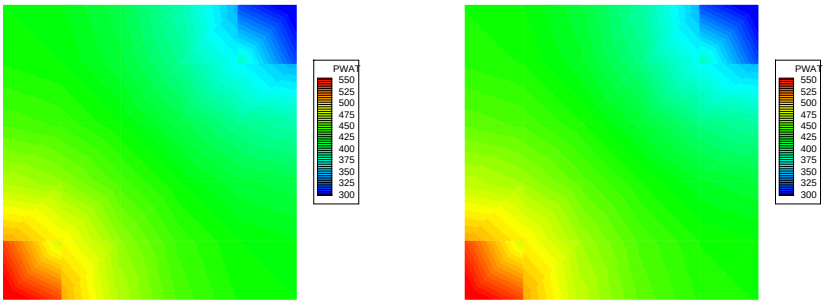
5.3 The output files

The Tecplot output files are written in the same way as in the framework. There are of the type `RUN.0.0.3.1` if the parameter `VIS_FNAME=RUN`. The variables that are printed are the wetting pressure and saturation. The script `mteccos` is then used to create a single file from the data at each time step, this script is in `visual/scripts/mteccos`.

6 Numerical examples

6.1 Two-dimensional simulations

Two dimensional results are obtained on three-dimensional domains by neglecting the gravity effects and using input data that is independent of the third dimension.



(a) (b)

Figure 3: CASE 1: Quadratic pressure at (a) 200 and (b) 500 days.

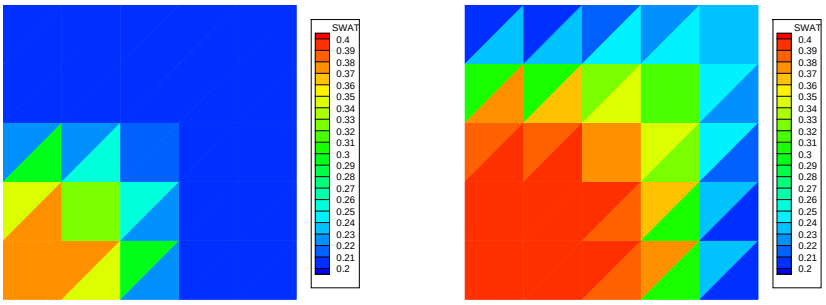
6.1.1 Quarter-five spot

In this example, the two-dimensional domain considered is the square $(0, 400) \times (0, 400)$ subdivided into triangular elements. We assume two types of permeability tensors: diagonal tensor and full tensor. The wetting phase is injected at the left bottom corner of the . The non-wetting phase is pushed out of the domain at the right top corner. This configuration corresponds to the quarter-five spot problem (see Fig. 2).

The following CASE 1, ... CASE 5 correspond to reservoirs with *diagonal permeability tensor*. The permeability to equal to $k\mathbf{I}$ with $k = 100mD$ (mDarcies). The injected saturation is 0.4. Boundary conditions for pressure are 550.0 psi at the bottom left corner and 300.0 psi at the top right corner. We note that the mesh is quite coarse. We consider several cases that show the capabilities of the method and the code. Final time of simulation is 500 days.

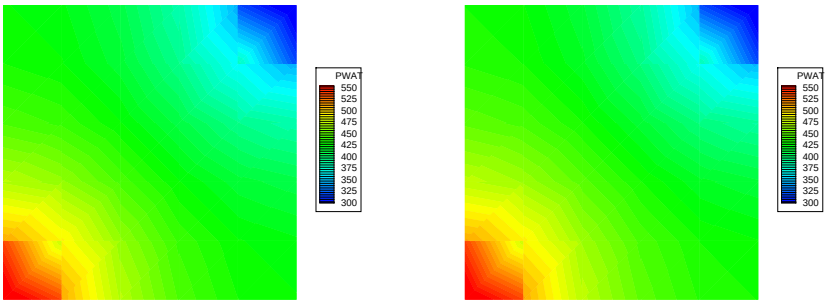
CASE 1: The DG saturation is piecewise constant and the DG pressure piecewise quadratic. Four saturation steps are computed per pressure step, which yields a total of 25 pressure steps and 100 saturation steps. The wetting phase pressure and saturation contours are shown at 200 days and 500 days in Fig. 3 and Fig. 4. The input file is given in the Appendix.

CASE 2: Similar situation than in CASE 1. Here, at each time step, we compute both pressure and saturation. There are 100 pressure and saturation steps. The wetting phase pressure and saturation contours are shown at 200 days and 500 days in Fig. 5 and Fig. 6.



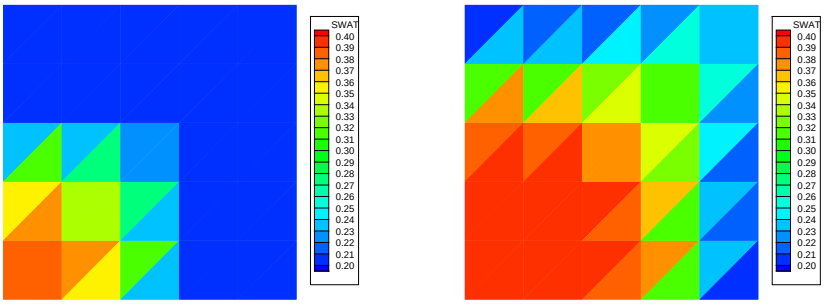
(a) (b)

Figure 4: CASE 1: Constant saturation at (a) 200 and (b) 500 days.



(a) (b)

Figure 5: CASE 2: Quadratic pressure at (a) 200 and (b) 500 days.



(a) (b)

Figure 6: CASE 2: Constant saturation at (a) 200 and (b) 500 days.

CASE 3: The DG saturation is piecewise linear and the DG pressure piecewise cubic. Four saturation steps are computed per pressure step, which yields a total of 25 pressure steps and 100 saturation steps. The wetting phase pressure and saturation contours are shown at 500 days in Fig. 7.

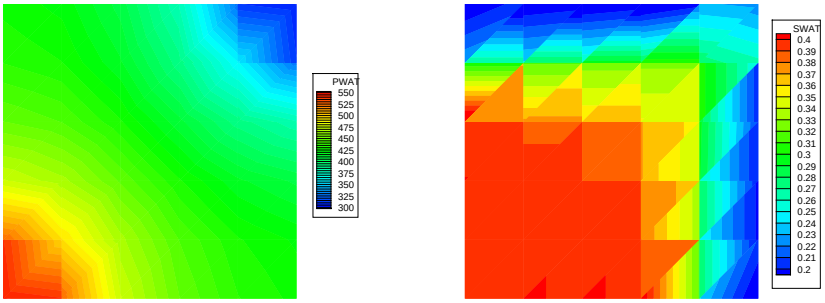
CASE 4: Same as in CASE 3 except that the pressure is piecewise quadratic. Here, at each time step, we compute both pressure and saturation. There are 100 pressure and saturation steps. The wetting phase pressure and saturation contours are shown at 500 days in Fig. 8.

CASE 5: Same situation as in CASE 3, except for the fact that penalty terms are used for solving the saturation equation. The penalty parameter is chosen to be $\sigma_e = 1.0$. The wetting phase pressure and saturation contours are shown at 500 days in Fig. 9.

The following CASE 6 and CASE 7 correspond to a reservoir with *full permeability tensor*. CASE 6 is on the coarse mesh (50 elements) and CASE 7 on the refined mesh (200 elements). The permeability field is defined as:

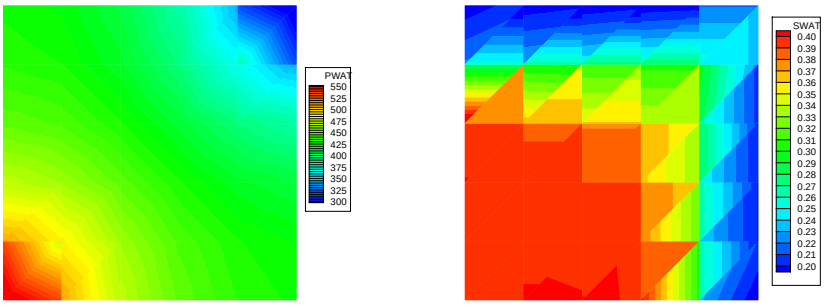
$$\mathbf{K} = \begin{pmatrix} 200 & 50 \\ 50 & 100 \end{pmatrix}$$

CASE 6: The DG saturation is piecewise linear and the DG pressure piecewise quadratic. Twenty saturation steps are computed per pressure step, which yields a total of 25 pressure steps and 500 saturation steps. The mesh used is the coarsest mesh (25 elements) as in the cases above. The wetting phase pressure and saturation contours are shown at 200 days and 500 days in Fig. 10 and Fig. 11.



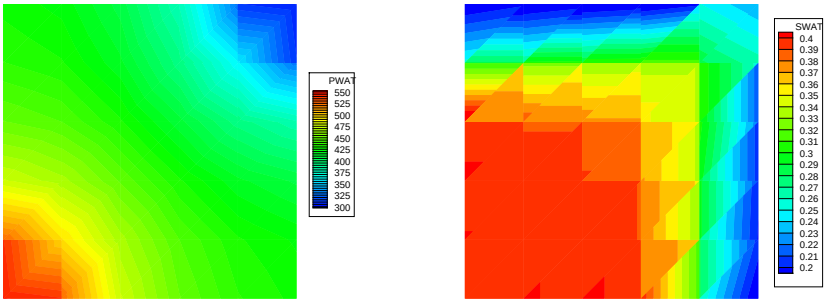
(a) (b)

Figure 7: CASE 3: (a) Cubic pressure and (b) linear saturation at 500 days.



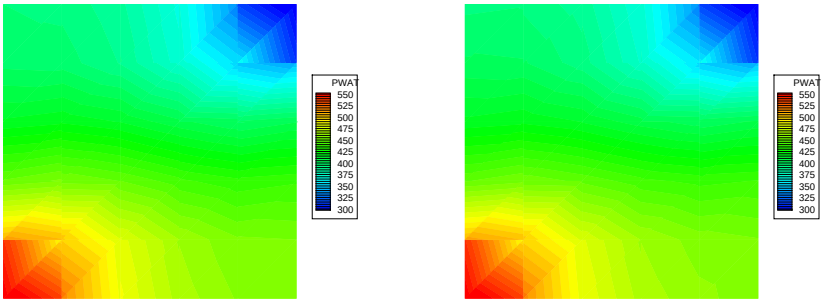
(a) (b)

Figure 8: CASE 4: (a) Quadratic pressure and (b) linear saturation at 500 days.



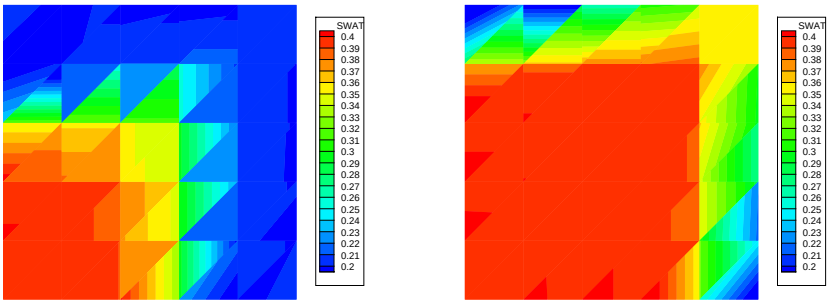
(a) (b)

Figure 9: CASE 5: (a) Cubic pressure and (b) linear saturation at 500 days. Penalty parameters for saturation are activated.



(a) (b)

Figure 10: CASE 6: Quadratic pressure at (a) 200 and (b) 500 days.



(a) (b)

Figure 11: CASE 6: Linear saturation at (a) 200 and (b) 500 days.

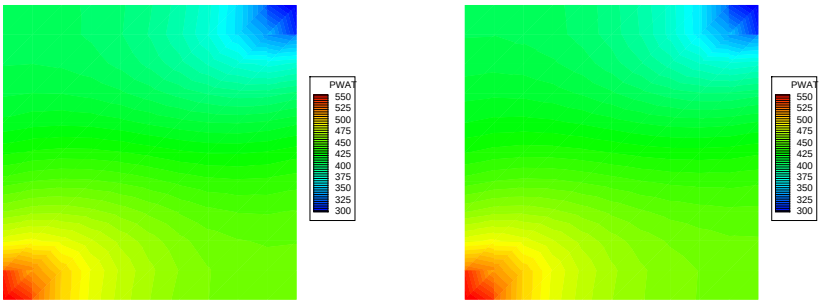
CASE 7: Same as in CASE 6, except for the fact that the mesh is refined with 200 elements instead of 25. The wetting phase pressure and saturation contours are shown at 200 days and 500 days in Fig. 12 and Fig. 13.

6.1.2 Fractured reservoir

In this example, the reservoir is characterized by slanted faults and an heterogeneous permeability field: $\mathbf{K} = k\boldsymbol{I}$, with $k \in \{100, 1000, 3000, 10000\}$ in milliDarcies (see Fig. 14). The domain is rectangular with dimensions $1500 \times 1000 ft^2$. Dirichlet conditions are imposed on the left (550psi) and right (300psi) boundaries and zero Neumann conditions on top and bottom boundaries for the pressure equation. The inflow boundary is the left boundary, where a saturation of 0.4 is injected in the medium.

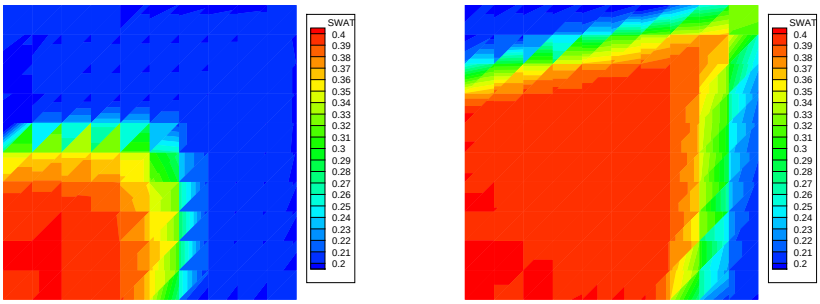
CASE 8: The DG pressure is piecewise quadratic and the DG saturation is piecewise constant. At each time step, both pressure and saturation are computed. The coarse mesh is used. The final simulation time is 300 days and the time step is taken to be 1 day. The wetting phase pressure and saturation contours are shown at 100 days and 300 days in Fig. 15 and Fig. 16.

CASE 9: The DG pressure is piecewise quadratic and the DG saturation is piecewise linear. At each time step, both pressure and saturation are computed. The coarse mesh is used. The final simulation time is 300 days and the time step is taken to be 1 day. The wetting phase pressure and saturation contours are shown at 100 days and 300 days in Fig. 17 and Fig. 18.



(a) (b)

Figure 12: CASE 7: Quadratic pressure at (a) 200 and (b) 500 days.



(a) (b)

Figure 13: CASE 7: Linear saturation at (a) 200 and (b) 500 days.

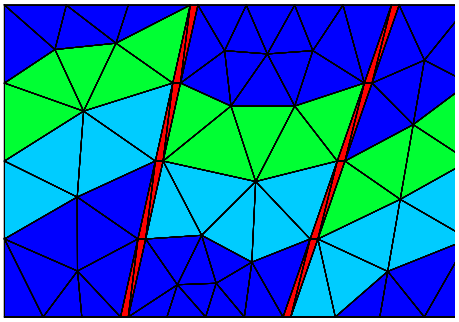
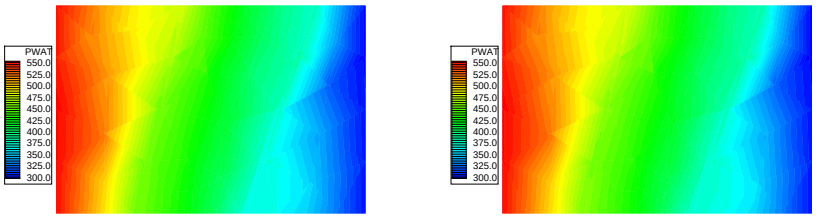
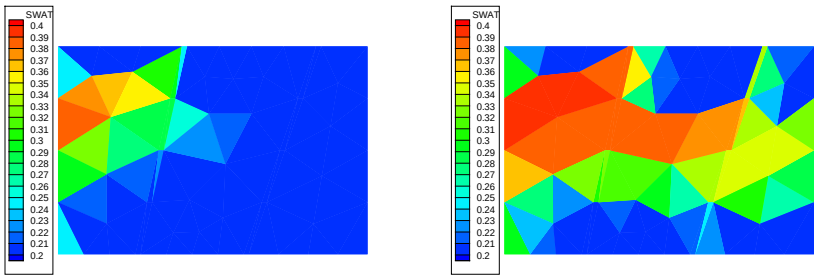


Figure 14: Permeability field for the fractured reservoir.



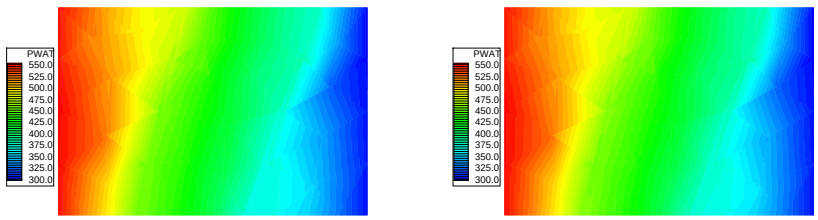
(a) (b)

Figure 15: CASE 8: Quadratic pressure at (a) 100 and (b) 300 days.



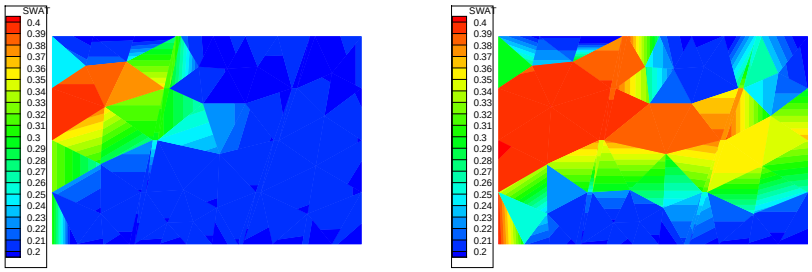
(a) (b)

Figure 16: CASE 8: Constant saturation at (a) 100 and (b) 300 days.



(a) (b)

Figure 17: CASE 9: Quadratic pressure at (a) 100 and (b) 300 days.



(a) (b)

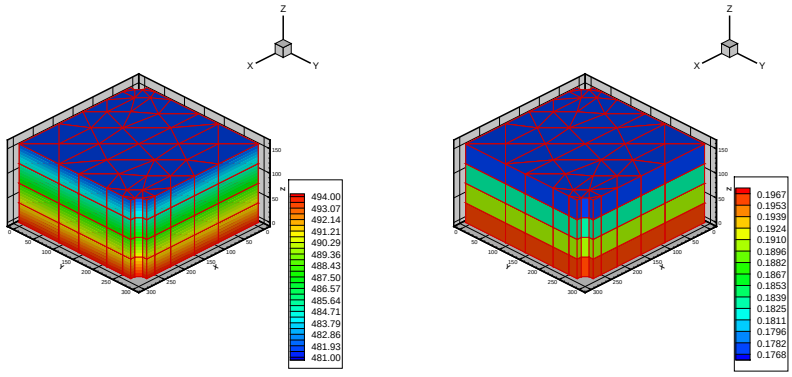
Figure 18: CASE 9: Linear saturation at (a) 100 and (b) 300 days.

6.2 Three dimensional simulations

Here, we present simulations of two-phase flow with injection and production wells. Gravity effects are included. The domain is the box $(0, 300) \times (0, 300) \times (0, 160)$ subdivided into prismatic elements. The permeability tensor is diagonal: $k\mathbf{I}$ with $k = 100$ mDarcies. Water is injected at the injection well and oil is produced at the production well. For the well models, the bottom-hole water pressure is specified: $510psi$ for the injection well, and $350psi$ for the production well. The injected saturation is 0.4.

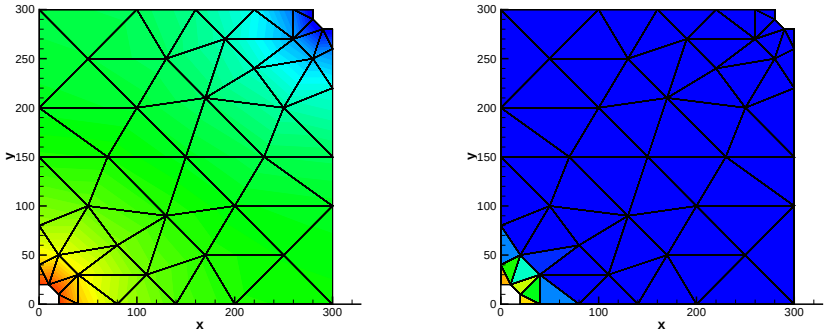
CASE 10: The DG pressure is piecewise quadratic and the DG saturation is piecewise constant. At each time step, both pressure and saturation are computed. The coarse mesh is used. The final simulation time is 300 days and the time step is taken to be 1 day. The initial pressure and saturation are shown in Fig. 19. The wetting phase pressure and saturation contours are shown at 100 days and 300 days in Fig. 21 and Fig. 22.

CASE 11: Same as for CASE 10, except that the mesh has been refined once and the DG saturation is piecewise linear. The wetting phase pressure and saturation contours are shown at 150 days and 300 days in Fig. 23 and Fig. 24.



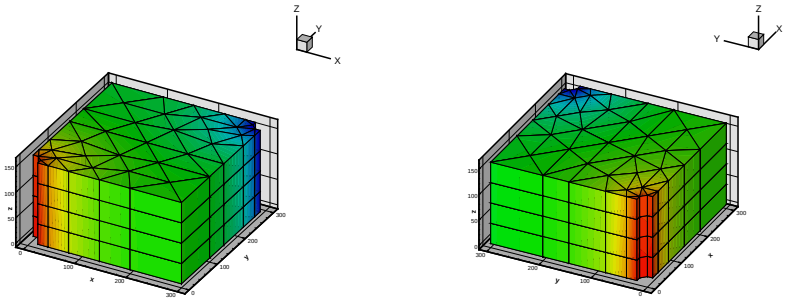
(a) (b)

Figure 19: CASE 10: Initialization of pressure (a) and saturation (b).



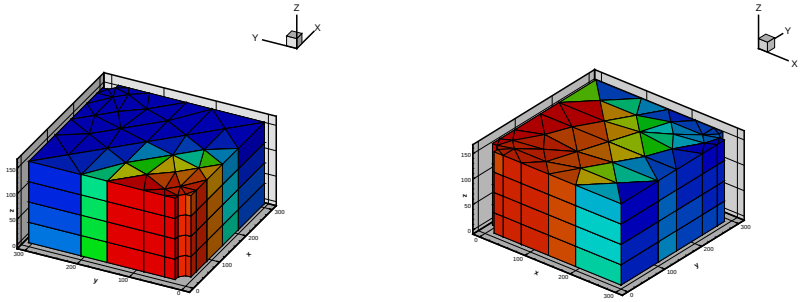
(a) (b)

Figure 20: CASE 10: Quadratic pressure (a) and constant saturation (b) at 10 days.



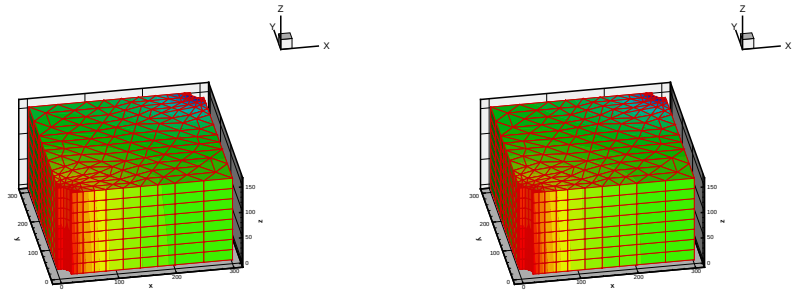
(a) (b)

Figure 21: CASE 10: Quadratic pressure at (a) 100 and (b) 300 days.



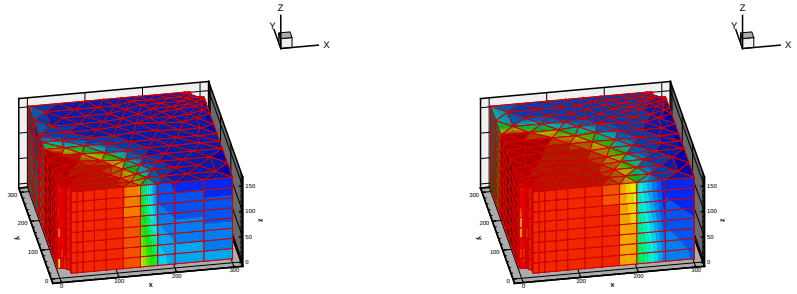
(a) (b)

Figure 22: CASE 10: Constant saturation at (a) 100 and (b) 300 days.



(a) (b)

Figure 23: CASE 11: Quadratic pressure at (a) 150 and (b) 300 days.



(a) (b)

Figure 24: CASE 11: Linear saturation at (a) 150 and (b) 300 days.

7 Conclusions and future work

We have formulated and implemented a discretization of the IMPES model for two-phase flow problem, that uses discontinuous polynomial approximations. The scheme can easily handle unstructured meshes and full tensors. We have implemented a well model using specified bottom-hole pressures. The numerical simulations show the advantages of the methods for taking into account the heterogeneities of the porous medium such as faults.

We summarize below the capabilities of the current model:

- Full permeability tensors.
- Penalty terms.
- Several saturation steps may be computed per pressure step.
- Prismatic elements.
- Bottom-hole pressure specified injection and production wells.

The full capabilities of the scheme have not been exploited yet. In particular, h and p adaptivity techniques can help in reducing the cost of the method. Penalty terms are included, but the effects of penalties have not been fully investigated. Solvers and slope limiters are two other important issues that should be further studied.

8 Acknowledgments

The author would like to thank Prof. Mary Wheeler for her help and support, and for her contribution in developing the model, Dr. Malgozarta Peszynska for her help in integrating DGIMPES into IPARS and Dr. John Wheeler for his help with the well model and the gravity term. We also acknowledge Dr. Krzysztof Banas for providing the 3D integration routines via quadrature rules.

9 Appendix

9.1 Derivation of the IMPES model for incompressible fluids

The conservation of mass for each phase are written as follows:

$$\frac{\partial(\phi N_\delta)}{\partial t} + \nabla \cdot \rho_\delta \boldsymbol{u}_\delta = 0, \quad \delta = n, w,$$

where $N_\delta = \rho_\delta S_\delta$ and where the phase velocities are given by:

$$\boldsymbol{u}_\delta = -\lambda_\delta \boldsymbol{K}(\nabla p_\delta - \rho_\delta G \nabla D).$$

Assuming that the fluids are incompressible, we can divide each of the above equations by the phase densities:

$$\frac{\partial(\phi S_\delta)}{\partial t} + \nabla \cdot \boldsymbol{u}_\delta = 0, \quad \delta = n, w.$$

We then add the two equations and use the fact that $S_w + S_n = 1$:

$$\nabla \cdot (\boldsymbol{u}_n + \boldsymbol{u}_w) = 0$$

We can rewrite the total velocity $\boldsymbol{u}_t = \boldsymbol{u}_n + \boldsymbol{u}_w$ by using the capillary pressure $p_c = p_n - p_w$:

$$\boldsymbol{u}_t = -(\lambda_w + \lambda_n)\boldsymbol{K}\nabla p_w - \lambda_n \boldsymbol{K}\nabla p_c + (\lambda_n \rho_n + \lambda_w \rho_w)\boldsymbol{K}G\nabla D.$$

Thus, if $\lambda_t = \lambda_w + \lambda_n$:

$$-\nabla \cdot \lambda_t \boldsymbol{K}\nabla p_w = \nabla \cdot \lambda_n \boldsymbol{K}\nabla p_c - \nabla \cdot (\lambda_n \rho_n + \lambda_w \rho_w)\boldsymbol{K}G\nabla D$$

This is (1) if we have $\nabla p_c = |p_c'|\nabla s_w$.

We also have

$$\frac{\lambda_w}{\lambda_t}\boldsymbol{u}_t = -\lambda_w \boldsymbol{K}\nabla p_w - \frac{\lambda_n \lambda_w}{\lambda_t}\boldsymbol{K}\nabla p_c + \frac{\lambda_w}{\lambda_t}(\lambda_n \rho_n + \lambda_w \rho_w)\boldsymbol{K}G\nabla D.$$

or

$$\frac{\lambda_w}{\lambda_t}\boldsymbol{u}_t = \boldsymbol{u}_w - \frac{\lambda_n \lambda_w}{\lambda_t}\boldsymbol{K}\nabla p_c + (\frac{\lambda_w}{\lambda_t}(\lambda_n \rho_n + \lambda_w \rho_w) - \lambda_w \rho_w)\boldsymbol{K}G\nabla D.$$

Therefore the wetting phase saturation equation becomes:

$$\frac{\partial(\phi s_w)}{\partial t}+\nabla\cdot\frac{\lambda_w}{\lambda_t}\boldsymbol{u}_t+\nabla\cdot\frac{\lambda_n\lambda_w}{\lambda_t}\boldsymbol{K}\nabla p_c+(-\frac{\lambda_w}{\lambda_t}(\lambda_n\rho_n+\lambda_w\rho_w)+\lambda_w\rho_w)\boldsymbol{K}G\nabla D=0$$

After algebraic manipulation of the right-hand side, we obtain

$$\frac{\partial(\phi s_w)}{\partial t}-\nabla\cdot\frac{\lambda_n\lambda_w}{\lambda_t}\boldsymbol{K}|p_c'|\nabla s_w=-\nabla\cdot\frac{\lambda_w}{\lambda_t}\boldsymbol{u}_t-\frac{\lambda_w\lambda_n}{\lambda_t}(\rho_w-\rho_n)\boldsymbol{K}G\nabla D.$$

This is exactly the equation (2).

9.2 Input file for CASE 1

```
TITLE(2)="CASE1 DGIMPES TEST "

DESCRIPTION()=

"BLOCK   LENGTH (FT)   WIDTH (FT)   HEIGHT (FT)    SIZE     CORNER"
"DATE : 8/02/02"

DG_IMPES

TIMEEND = 500.0

$$$$$$$$$$$$$

$ I/O OPTIONS

OUTLEVEL = 2

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ FAULT BLOCK AND MESH DATA

BLOCKNAME(1) = "BLOCK1"
DOWN(1 TO 3,1) = 1 0 0
NX(1) = 1          NY(1) = 1           NZ(1) = 1
DX(,1) = 1.  DY(,1) = 1.  DZ(,1) = 1.
XYZ111(,1) = 0. 0. 0.
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ IPARS data
$ INITIAL CONDITIONS

DGPOINIT = 500. DGSWINIT = 0.200000

PRESS_MESHTYP = 2
PRESS_DEG = 2
PRESS_BASE = 2
PRESS_PENAL = 0.0
PRESS_T_MONIT = 2
PRESS_A_TYPE = 0
PRESS_L_TYPE = 0
PRESS_L_MAX = 1000
PRESS_L_CONVTYP = 2
PRESS_L_COME = 1.E-12
PRESS_L_MONIT = 0

$BOUNDARY CONDITIONS
PRESS_BC = 3
PRESS_TYPE1 = 1
PRESS_TYPE2 = 2
PRESS_TYPE3 = 11
PRESS_VAL1 = 550.0
PRESS_VAL2 = 300.0
PRESS_VAL3 = 0.0

PRESS_OUTPUT = 1

SAT_MESHTYP = 2
SAT_DEG = 0
SAT_BASE = 2
SAT_SLOPE = 0
SAT_PENAL = 0.0
SAT_LIMIT = 0.7
SAT_A_TYPE = 0
SAT_L_TYPE = 1
SAT_L_MAX = 100
```

```
SAT_L_CONVTYP = 2
SAT_L_COME = 1.E-14
SAT_L_MONIT = 0

$BOUNDARY CONDITIONS
SAT_BC = 3
SAT_TYPE1 = 21
SAT_TYPE2 = 11
SAT_TYPE3 = 31
SAT_VAL1 = 0.4
SAT_VAL2 = 0.0
SAT_VAL3 = 0.0

SAT_OUTPUT = 1

$ VISCOSITY

DGOILVIS = 2.0[cp]
DGWATVIS = 0.5[cp]

$ POROSITY

DGPOROSITY = 0.2

$DENSITY
DGOILDEN = 56.0
DGWATDEN = 62.34

$COMPRESSIBILITY
DGOILCOMP = 1.E-10
DGWATCOMP = 1.E-12

$NUMBER OF SATURATION STEPS PER PRESSURE STEP
DGNSSTEP = 4

$ TURN ON GRAVITY = WELL MODEL
DG_GRAVITY = -1
```

```
$ Note that the line below is not used in DG code but is aksed in the
$ framework by GETROCK
POROSITY1() = .2

$ PERMEABILITIES

DGPERMXX = 100.0
DGPERMYY = 100.0
DGPERMZZ = 100.0
DGPERMXY = 0.0
DGPERMYZ = 0.0
DGPERMXZ = 0.0

$ Note that the line below is not used in DG code but is aksed in the
XPERM1() = 0  YPERM1() = 0

$ the following tables will define ko, kw
KOSW(1) Block        $ OIL RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Linear
  Extrapolation Constant
  Constraint  0 At .8
  Derivative  0 At .8
  Constraint  1 At  0
  Nodes  .2  .58
  Data  0.  1. , .1  .67 , .2  .46 , .4  .2 , .6  .055 , .7  .015 , .8  0
EndBlock

KWSW(1) Block        $ WATER RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Linear
  Extrapolation Constant
  Constraint  0 At .15
  Derivative  0 At .15
  Constraint  1 At   1
  Nodes  .55  .7  .75
  Data  .15  0 , .3  .035 , .4  .085 , .6  .28 , .8  .776 , .9 .9, 1  1
EndBlock

$ WATER-OIL CAPILLARY PRESSURE - ROCK TYPE 1
```

```
PCOW(1) Block        $ WATER-OIL CAPILLARY PRESSURE - ROCK TYPE 1
  Interpolation Spline3
  Extrapolation Same
  Nodes  .25  .4  .7  .9
  Pole   .12
  Data
  .16   9.00 , .2   6.12 , .225 4.86 , .25  4.22 ,
  .275 3.78 , .325 3.20 ,
  .375 2.74 , .45  2.28 , .55  1.94 ,
  .65  1.74 , .75  1.61 , .85  1.54
 .925 1.44 , .95  1.37 , .975 1.14 , 1.0  .70
EndBlock

EndInitial

$tables output
$VIS_TABOUTTYPE = 1

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ TRANSIENT DATA INPUT BLOCKS

BeginTime    0.
DELTIM = 20.0  DTIMMUL = 1.0  DTIMMAX = 1000.0  DTIMOUT = 20.0 DTIMMIN = 20.0
TIMOUT = 1.0
VISOUT = 0.0
DVISOUT = 20.0
VISFLAG = 3
VIS_FNAME = "BEA"
EndTime
```

References

[1] Integrated Parallel Accurate Reservoir Simulator, developed by the researchers J. Wheeler, M. Wheeler, K. Banas, S. Bryant, S. Chippada, J. Eaton, C. Edwards, E. Jenkins, S. Lacroix, W. Lee, Q. Lu, B. Monken, M. Noh, M. Parashar, V. Parr, M. Peszynska, B. Riviere, P. Quandalle,

K. Sephernoori, Y. Vassilevski, P. Wang, G. Xiuli, I. Yotov at the Center for Subsurface Modeling.

[2] IPARS v2, Users' Manual, The Center for Subsurface Modeling, 1998.

[3] V. Aizinger, C.N. Dawson, B. Cockburn and P. Castillo. The local discontinuous Galerkin method for contaminant transport. Advances in Water Resources 24 (2000), pp. 73–87.

[4] T. Arbogast, S. Bryant, J. Eaton, Q. Lu, M. Peszynska, M.F. Wheeler and I. Yotov. A parallel multiblock/multidomain approach for reservoir simulation. SPE Journal, in revision.

[5] D. Arnold, F. Brezzi, C. Cockburn, and D. Marini. Discontinuous Galerkin methods for elliptic problems. In G.E. Karniadakis B. Cockburn and eds. C.-W. Shu, editors, *First International Symposium on Discontinuous Galerkin Methods*, volume 11 of *Lecture Notes in Computational Science and Engineering*, pages 89–101. Springer-Verlag, 2000.

[6] K. Banas DG3D - A discontinuous Galerkin, hp-adaptive finite element code for 3D simulations. preprint, 2001.

[7] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier-stokes equations. *J. Comput. Phys.*, 131:267–279, 1997.

[8] B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.*, 35(6):2440–2463 (electronic), 1998.

[9] R. Helmig. Multiphase Flow and Transport Processes in the Subsurface. Springer (1997).

[10] R. Huber and R. Helmig. Multiphase Flow in Heterogeneous porous media: a classical finite element method versus an implicit pressure-explicit saturation-based mixed finite element-finite volume approach. International Journal for Numerical Methods in Fluids, 29 (1999), pp. 899–920.

[11] P. Houston, E. Suli and C. Schwab. Discontinuous hp-finite element methods for advection-diffusion problems. to appear in Math. Comp.

[12] W. Lee, M. Noh, and M. F. Wheeler. Air-water flow simulation in unsaturated porous media. In L. R. Bentley, J. F. Sykes, C. A. Brebbia, W. G. Gray, and G. F. Pinder, editors, *Computational Methods in Water Resources*, pages 93–100. A. A. Balkema, 2000.

[13] Q. Lu, M. Peszynska, M.F. Wheeler. A Parallel Multi-Block Black-Oil Model in Multi-Model Implementation. SPE Journal, to appear.

[14] S. Minkoff, C.M. Stone, J.G. Arguello, S. Bryant, J. Eaton, M. Peszynska and M.F. Wheeler. Staggered in time coupling of reservoir flow simulation and geomechanical deformation: Step 1 - one-way coupling. 1999 SPE Symposium on Reservoir Simulation, Houston, Texas, 1999, SPE 51920.

[15] J. T. Oden, I. Babuška, and C. E. Baumann. A discontinuous hp finite element method for diffusion problems. *J. Comput. Phys.*, 146:491–516, 1998.

[16] M. Peszynska, E.W. Jenkins, M.F. Wheeler. Boundary Conditions for Fully Implicit Two-Phase Flow Models. *Barrett Lecture proceedings, Contemporary Mathematics Series*, to appear. also TICAM Report 01-38.

[17] M. Peszynska, Q. Lu, and M. F. Wheeler. Coupling different numerical algorithms for two phase fluid flow. In J. R. Whiteman, editor, *MAFELAP Proceedings of Mathematics of Finite Elements and Applications*, pages 205–214, Uxbridge, U.K., 1999. Brunel University.

[18] M. Peszynska, Q. Lu, and M. F. Wheeler. Multiphysics coupling of codes. In L. R. Bentley, J. F. Sykes, C. A. Brebbia, W. G. Gray, and G. F. Pinder, editors, *Computational Methods in Water Resources*, pages 175–182. A. A. Balkema, 2000.

[19] M. Parashar, J. Wheeler, G. Pope, K. Wang and P. Wang. A new generation EOS compositional reservoir simulator, part II: Framework and multiprocessing. Fourteenth SPE Symposium on Reservoir Simulation, Dallas, Texas, SPE, June 1997, pp. 31–38.

[20] M. Peszynska, M. F. Wheeler, and I. Yotov. Mortar upscaling for multiphase flow in porous media. Computational Geosciences 6 (2002), to appear.

[21] M. Peszynska. Multiphysics Coupling of Three-Phase and Two-Phase Models of Flow in Porous Media. submitted,also TICAM Report 02-20.

[22] M. Peszynska, Advanced Techniques and Algorithms for Reservoir Simulation, III: Multiphysics coupling for two phase flow in degenerate conditions. *IMA Volumes in Mathematics and its Applications*, Volume 131: Resource Recovery, Confinement, and Remediation of Environmental Hazards Editors: J. Chadam, A. Cunningham, R.E. Ewing, P. Ortoleva, and M.F. Wheeler, pp 21-40, Springer-Verlag, 2002

[23] B. Rivière. Discontinuous Galerkin finite element methods for solving the miscible displacement problem in porous media. PhD Thesis, The University of Texas at Austin, May 2000.

[24] B. Rivière and E. Jenkins In Pursuit of Better Models and Simulations, Oil Industry Looks to the Math Sciences. *SIAM News*, 35 (1), January-February 2002.

[25] B. Rivière and M.F. Wheeler. A discontinuous Galerkin Method applied to nonlinear parabolic equations. First International Symposium on Discontinuous Galerkin Methods (B. Cockburn, G.E. Karniadakis and C.-W. Shu, eds.), Lecture Notes in Computational Science and Engineering, Springer-Verlag, 11 (2000), pp. 231–244.

[26] B. Rivière, M.F. Wheeler and V. Girault, Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. Part I. Computational Geosciences 3 (1999), pp. 337–360.

[27] B. Rivière, M.F. Wheeler and V. Girault. A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems. SIAM J. Numer. Anal. 39 (3) (2001), pp. 902–931.

[28] B. Rivière and M.F. Wheeler, High order numerical schemes for two-phase flow modeling. In preparation.

[29] K. Kundert. Sparse matrix techniques, in circuit analysis, simulation and design. A. Ruehli ed., North Holland 1986.

[30] M.F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM J. Numer. Anal.*, 15(1):152–161, 1978.

[31] P. Wang, I. Yotov, M.F. Wheeler, T. Arbogast, C. Dawson, M. Parashar and K. Sepehrnoori. A new generation EOS compositional reservoir simulator, part I: Formulation and discretization. Fourteenth SPE Symposium on Reservoir Simulation, Dallas, Texas, SPE, June 1997, pp. 55–64.