Oil & Gas Science and Technology - Rev. IFP Energies nouvelles **73**, 38 (2018)

https://doi.org/10.2516/ogst/2018022

Available online at:
www.ogst.ifpenergiesnouvelles.fr

Numerical methods and HPC
A. Anciaux-Sedrakian and Q. H. Tran (Guest editors)

REGULAR ARTICLE OPEN ACCESS

Application of the mixed multiscale finite element method to parallel simulations of two-phase flows in porous media

Maria Adela Puscas[1], Guillaume Enchéry[2,*], and Sylvain Desroziers[2]

[1] CEA/Saclay, Bât. 451, 91191 Gif-sur-Yvette Cedex, France
[2] IFP Energies nouvelles, 1–4 avenue de Bois-Préau, 92852 Rueil-Malmaison Cedex, France

Received: 2 January 2018 / Accepted: 14 June 2018

Abstract. The Mixed Multiscale Finite Element method (MMsFE) is a promising alternative to traditional upscaling techniques in order to accelerate the simulation of flows in large heterogeneous porous media. Indeed, in this method, the calculation of the basis functions which encompass the fine-scale variations of the permeability field, can be performed in parallel and the size of the global linear system is reduced. However, we show in this work that a two-level MPI strategy should be used to adapt the calculation resources at these two steps of the algorithm and thus obtain a better scalability of the method. This strategy has been implemented for the resolution of the pressure equation which arises in two-phase flow models. Results of simulations performed on complex reservoir models show the benefits of this approach.

1 Introduction

Simulations of subsoil flows are based on geological models which provide a description of the geometry of the porous medium and its petrophysical properties such as porosity and permeability. The space variations of these properties play an important role on the flow. In industrial applications, like in the oil and gas industry for instance, the geological description of the studied area often leads to models where the permeability and porosity maps are defined on grids made of dozens or hundreds of millions of cells. This initial discretization will be referred to as fine-scale model in the following. Simulating multiphase flows, directly at this grid resolution, may lead to large computing times. This problem gets even worse when several simulations have to be launched in order to perform a sensitivity analysis, a calibration of the model parameters, or test different flow scenarios. To overcome this issue, various upscaling techniques and multiscale methods have been developed over the past years.

Upscaling techniques essentially consist in averaging the petrophysical properties defined in the fine-scale model on a coarser grid [1, 2]. But they usually do not enable to downscale solutions obtained with this grid onto the fine one. This requirement is particularly important when working with workflows including both the fine and coarse models. Local Grid Refinements (LGRs) can be used jointly with these techniques, as in [3], in order to obtain more accurate solutions at particular places within a reservoir. However LGRs cannot be used over large space areas. Therefore the prospect of computing efficiently solutions at both fine and coarse scales motivated the development of multiscale methods. In this class of methods, the petrophysical properties are used to compute basis functions which are associated to degrees of freedoms related to the coarse mesh and calculated by solving local cell problems defined on the fine mesh. This set of basis functions is then used to build a discrete linear system with a reduced number of unknowns on the coarse mesh.

This approach has several advantages. Compared to classical finite elements, the shape of the basis functions changes according to the local variations of the permeability field within their support. All cell problems are independent from each other and their resolution can be performed in parallel. Moreover, once the linear system has been solved at the coarse scale, according to the multiscale method, either the fluxes or the pressures can be downscaled on to the fine grid.

The first multiscale finite element method has been introduced in [4] for the study of porous materials. Unfortunately, this method does not turn out to be appropriate for flow simulations since the fluxes are not conservative on the coarse and fine meshes. A mixed formulation was therefore proposed in [5] which enables, in case of two-phase flows, to couple the resolution of the pressure equation with the resolution of the saturation equation. Note that

* Corresponding author: guillaume.enchery@ifpen.fr

other multiscale formulations like, for instance, multiscale finite-volume methods [6–8] have also been proposed and provide mass conservative fluxes too. The reader can refer to [9, 10] for an overview of these techniques. The aim of this work is here to provide strategies to improve the efficiency of these methods when they are used in a parallel environment and give quantitative speed-ups by using one of them, namely the Mixed Multiscale Finite Element Method (MMsFE) [5].

In this work, we consider a two-phase flow model where the pressure equation is decoupled from the saturation equation using IMPES or IMPIMS time discretizations. The pressure equation is solved by means of the MMsFE method whereas an upwind finite volume scheme is used to discretize the saturation equation at the fine-scale. This splitting is feasible since the MMsFE method provides conservative fluxes at the fine and coarse scales. The cell problems defined to compute the basis functions are discretized using a finite volume method with a Two Point Flux Approximation (TPFA) [11].

As mentioned before, the divide and conquer strategy which is inherent to multiscale methods makes them attractive for parallel simulations since all cell problems defined to compute the basis functions can be solved independently from each other. Shared memory architectures are usually used to efficiently deal with tasks in parallel. A parallel implementation of the MMsFE method for one single node has been proposed in [12] using the Matlab MRST toolbox [13]. For multi and manycores architectures, a parallel implementation has also been successfully developed in [14]. However, for large scale simulations, distributed memory computing is required. A recent work presents a hybrid parallel implementation within the DUNE framework [15]. Here, the hybrid parallel programming model consists in using a shared memory programming within SMP nodes and MPI-communications between nodes. In this approach, no communication is needed within one node. However, this requires to develop effective threadsafe components everywhere in the simulator.

In practice, the efficiency of the MMsFE method for the two phase flow problem may be reduced for two reasons. Even if all cell problems, which are defined to compute the basis functions, are solved on a small space subdomain including a few coarse cells, these problems can be numerous. Therefore the MMsFE method leads to substantial speedups only if a small part of the basis functions is recomputed at each time step. A criterion based on the variations of the total mobility is usually used to trigger an update of the basis in the neighborhood of the saturation front. The performance of the multiscale method is thus optimal when this front is not too large and the calculation workload related to the resolution of all cell problems is dynamically balanced through the processors. The second issue comes from the small size of the coarse linear system. Since multiscale methods offer the possibility to agglomerate a significant number of fine cells into a coarse one, the resulting distributed linear system can be small compared to the number of MPI processes required by other parts of the simulation sequence. Therefore, whereas a promising scaling may be expected for other parts of the algorithm, this strong reduction of the size of the linear system can lead to scalability problems. This issue was already pointed out in [15].

The purpose of this work is to address this second issue. We investigated pure MPI and hybrid approaches to finally propose a two-level MPI alternative. In this work, we use a redistribution library for linear systems in addition to the MPI parallel programming model, which enables the use of a subset of MPI processes to solve the coarse pressure system and avoids scaling issues when a larger number of MPI processes is required elsewhere in the resolution process. The performance of this technique has been assessed on oil reservoir models.

This paper is organized as follows. In Section 2, we briefly present the two-phase flow model. We introduce some notations and recall the MMsFE method in Section 3. First numerical results are then presented in Section 4. The different parallel programming techniques mentioned before are detailed and discussed in Section 5. The performances of the MMsFE method used along with the techniques presented in Section 5 are evaluated in Section 6. At last, concluding remarks and perspectives of works are summarized in Section 7.

2 Mathematical model

In this section, we first describe a simplified model for two-phase fluid flows in porous media. Then, we briefly outline a finite-volume method based on a sequential splitting that is classically used to solve the corresponding system of equations.

2.1 Two-phase flow problem

We consider an immiscible and incompressible two-phase flow in a porous medium $\mathbf{\Omega}$. Here, $\mathbf{\Omega}$ is a subset of $\mathbb{R}^d$ with $d \geq 2$. Let us denote by subscripts "w" and "o" the water and oil phases respectively. The equations governing fluid flows in a heterogeneous porous medium are derived from the conservation of mass. For each phase α, we write the mass balance

$$\phi \frac{\partial}{\partial t} S_\alpha + \nabla \cdot \mathbf{v}_\alpha = q_\alpha, \tag{1}$$

where ϕ denotes the total porosity or saturated volume fraction, S_α the phase saturation, $\mathbf{v}_\alpha$ the phase velocity, and q_α a source (or sink) term. Since both fluids fill the porous volume, the saturations satisfy the closure equation

$$\sum_{\alpha \in \{w,o\}} S_\alpha = 1. \tag{2}$$

In this work, gravity and capillary pressure are not taken into account. Hence, according to the two-phase extension of Darcy's law [16], the velocity $\mathbf{v}_\alpha$ can be formulated for each phase α as follows:

$$\mathbf{v}_\alpha = -\frac{k_{r,\alpha}}{\mu_\alpha} \mathbf{K} \nabla P, \tag{3}$$

where $\mathbf{K}$ is the permeability tensor, μ_α the viscosity and $k_{r,\alpha}$ is the saturation dependent relative permeability of the phase α. We introduce the phase mobility

$$\lambda_\alpha = \frac{k_{r,\alpha}}{\mu_\alpha}$$

and the total mobility $\lambda(S_\mathrm{w}) = \lambda_\mathrm{w}(S_\mathrm{w}) + \lambda_\mathrm{o}(1 - S_\mathrm{w})$.

Combining (1), (2) and (3), the following pressure equation can be obtained:

$$\begin{aligned} \mathbf{v} &= -\lambda(S_\mathrm{w})\,\mathbf{K}\,\nabla P, \\ \nabla\cdot\mathbf{v} &= q, \end{aligned} \tag{4}$$

where $\mathbf{v} = \mathbf{v}_\mathrm{w} + \mathbf{v}_\mathrm{o}$ and $q = q_\mathrm{w} + q_\mathrm{o}$. Then, by introducing the water fractional flow

$$f_\mathrm{w}(S_\mathrm{w}) = \frac{\lambda_\mathrm{w}(S_\mathrm{w})}{\lambda(S_\mathrm{w})}$$

and writing the pressure gradient with respect to the total velocity $\mathbf{v}$, this leads to the saturation equation

$$\phi\frac{\partial S_\mathrm{w}}{\partial t} + \nabla\cdot(f_\mathrm{w}\,\mathbf{v}) = q_\mathrm{w}. \tag{5}$$

Let $\partial\Omega$ be the boundary of Ω whose outward unit normal vector is denoted by $\mathbf{n}$. $\partial\Omega$ is partioned so that $\partial\Omega = \Gamma_D \cup \Gamma_N$. Boundary conditions are imposed in the following way for the pressure equation (4):

$$\begin{cases} P = P_b(x) & \text{on} \quad \Gamma_D, \\ \mathbf{v}\cdot\mathbf{n} = 0 & \text{on} \quad \Gamma_N, \end{cases} \tag{6}$$

and for the saturation equation (5):

$$S_\mathrm{w} = S_\mathrm{b}(x) \quad \text{if} \quad \mathbf{v}\cdot\mathbf{n} < 0 \quad \text{on} \quad \Gamma_D. \tag{7}$$

From now on, the notation S instead of S_w will be used for simplicity's sake.

2.2 Numerical resolution

The system of equations (4) and (5) could be solved by different ways [17]. Here, the splitting of the system into an elliptic equation and a hyperbolic equation that was introduced in the previous section is used to compute sequentially both pressure and saturation according to the scheme proposed in [18]. In a first step, the pressure equation (4) is solved at the current time step using saturation values from the previous time step. Then the saturation equation (5) is solved using the new pressure values. Thus, at each time iteration $n+1$, the system to be solved reads

$$\nabla\cdot\left(\mathbf{v}\left(P^{n+1}, S^n\right)\right) = 0, \tag{8}$$

$$\phi\frac{S^{n+1} - S^n}{\Delta t} + \nabla\cdot\left(f_w\left(S^\star\right)\mathbf{v}\left(P^{n+1}, S^n\right)\right) = 0, \tag{9}$$

with the boundary conditions introduced in (6) and (7) and $\Delta t = t^{n+1} - t^n$. To evaluate the water fractional flow in (9), a first possible choice consists in using an explicit Euler scheme where $S^\star = S^n$ leading to the IMPES formulation (IMplicit in Pressure, Explicit in Saturation). To ensure the stability of the saturation calculations, a CFL condition on the time steps is required [19]. An implicit Euler scheme can also be used. In that case, $S^\star = S^{n+1}$. This second formulation is known as IMPIMS scheme (IMplicit in Pressure, IMplicit in Saturation). Appendix A.2 gives the complete discretization of this equation. At last, the computation of the saturations S^{n+1} requires the resolution of a nonlinear system due to the presence of the water fractional flow. This system is solved by means of a Newton-Raphson method [20]. The solution obtained with this method is unconditionally stable whatever the time step Δt is.

Hereafter, equation (8) is called the pressure equation and equation (9) the saturation equation. The saturation equation is discretized on the fine mesh using an upwind cell-centered finite volume method [21] where the fluxes are computed either by means of the two-point flux approximation (TPFA) [11] or by means of the MMsFE method. The TPFA approximation is well-known to be consistent and monotonous for Cartesian grids. Since, in this work, our numerical examples only use this type of grid, the TPFA discretization will be considered for our comparisons as the fine reference solution. Appendix A recalls that discretization. Note that, when dealing with unstructured meshes, multi-point flux approximations [22] or mimetic finite difference methods [23] can be employed to ensure the consistency of the fluxes. The next section introduces the MMsFE method.

3 Mixed multiscale finite element method

Multiscale finite element methods provide an efficient way to compute the pressure P and the velocity $\mathbf{v}$ by solving a linear system defined on a coarse mesh while taking the details of a geological model into account at a fine-scale. Unlike upscaling techniques which average the values of the rock properties, multiscale methods actually compute basis functions whose shape depends on the space variability of these properties. The construction of such basis functions is achieved through the resolution of local subproblems called cell problems. Then a coarse linear system is assembled using a variational formulation based on these basis functions.

3.1 Multiscale mesh construction

Corner-point geometry grids [24] which are composed of hexaedral cells with a Cartesian topology are usually used for real test cases. Here, as a first trial, only Cartesian grids were considered for this study since the coarsening of such grids is easy for parallel simulations. In the following of this section, we first introduce some notations and the construction of the coarse mesh afterwards.

3.1.1 General notations

Let $\mathscr{K}_h$ be a conforming polyhedral mesh of the domain Ω such that $h = \max_{k\in\mathscr{K}_h} h_k$ where h_k is the diameter of the cell

$k \in \mathcal{K}_h$. Let $\mathcal{F}_h$ denote the set of faces related to the mesh $\mathcal{K}_h$. Within this set, $\mathcal{F}_h^i$ stands for the group of *inner* faces:

$$\mathcal{F}_h^i = \{\sigma = \partial k_1 \cap \partial k_2 | k_1, k_2 \in \mathcal{K}_h\}$$

and $\mathcal{F}_h^b$ for the group of *boundary* faces:

$$\mathcal{F}_h^b = \{\sigma \subset \partial k \cap \partial\Omega | k \in \mathcal{K}_h\}.$$

Let us denote by $\mathcal{F}_h^D$ the set of boundary faces included in Γ_D:

$$\mathcal{F}_h^D = \{\sigma \in \mathcal{F}_h^b | \sigma \subset \Gamma_D\}.$$

For all cell $k \in \mathcal{K}_h$, we also define

$$\mathcal{F}_{h,k} = \{\sigma \in \mathcal{F}_h | \sigma \subset \partial k\},$$

$$\mathcal{F}_{h,k}^i = \mathcal{F}_{h,k} \cap \mathcal{F}_h^i,$$

$$\mathcal{F}_{h,k}^b = \mathcal{F}_{h,k} \cap \mathcal{F}_h^b,$$

$$\text{and } \mathcal{F}_{h,k}^D = \{\sigma \in \mathcal{F}_{h,k}^b | \sigma \subset \Gamma_D\}.$$

Conversely, let us define for all $\sigma \in \mathcal{F}_h$

$$\mathcal{K}_{h,\sigma} = \{k \in \mathcal{K}_h | \sigma \subset \partial k\}.$$

For all inner face $\sigma \in \mathcal{F}_h^i$, an orientation is arbitrarily chosen and a unit normal vector $\mathbf{n}_\sigma$ is defined based on this orientation. For all $k \in \mathcal{K}_{h,\sigma}$, we denote by $\mathbf{n}_{\sigma,k}$ the unit normal vector of σ oriented outside of cell k and we define $\varepsilon_{\sigma,k} = \mathbf{n}_{\sigma,k} \cdot \mathbf{n}_\sigma$.

3.1.2 Upgridding

We denote by $\mathcal{K}_h$ the fine mesh and by $\mathcal{K}_H$ its corresponding coarse mesh with $h = H$. In our case, the coarse mesh $\mathcal{K}_H$ is generated by agglomerating the fine cells $k \in \mathcal{K}_h$ by means of a uniform partitionning of the structured grid $\mathcal{K}_h$. By that way, the coarse mesh faces are still aligned with the underlying fine grid. An example of such a mesh is given in Figure 1. Note that other works [25] proposed flow-based coarsening methods that were also applied along with multiscale methods.

Finally, we define the following sets mapping elements of $\mathcal{K}_h$ and $\mathcal{K}_H$:

$$\mathcal{M}_K^{H,h} = \{k \in \mathcal{K}_h | k \subset K\},$$

$$\mathcal{M}_k^{h,H} = \{K \in \mathcal{K}_H | k \subset K\}.$$

3.2 Definition

The MMsFE method introduces the multiscale finite element space

$$\mathcal{V}_H = \{\mathbf{q}_H \in H(\text{div};\Omega) : \mathbf{q}_{H|K} \in MS(K), \forall K \in \mathcal{K}_H\}$$

to approximate the coarse fluxes $\int_\Sigma \mathbf{v} \cdot \mathbf{n}_{\Sigma,K}\, \mathrm{d}s$ for all $\beta \in \{i, b\}, K \in \mathcal{K}_H, \Sigma \in \mathcal{F}_{H,K}^\beta$. Here $MS(K)$ denotes a

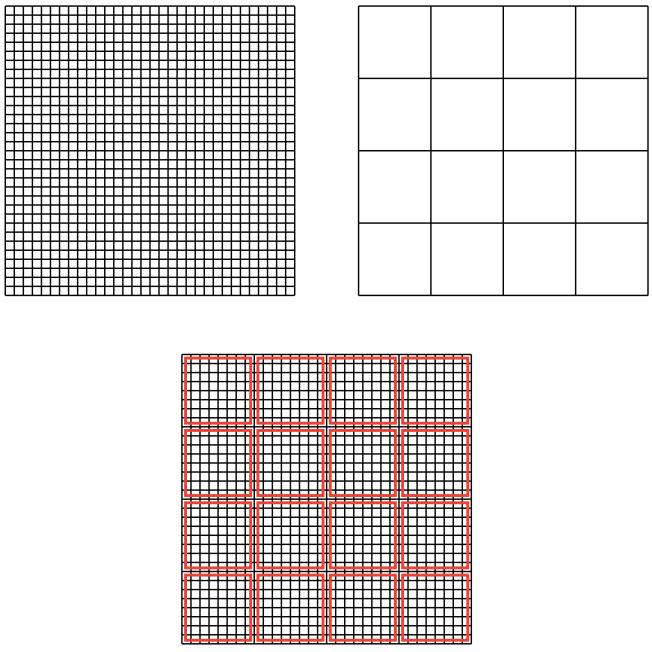

Fig. 1. Construction of the coarse cartesian grid on the right from the fine cartesian grid on the left. In the bottom, the agglomeration is shown.

finite element space spanned by basis functions $\boldsymbol{\psi}_\Sigma = -\lambda K \nabla \phi_\Sigma, \Sigma \in \mathcal{F}_{H,K}^\beta$. These basis functions are computed by solving cell problems on the fine underlying cells $k \in \mathcal{M}_K^{H,h}$. The degrees of freedom of $\mathbf{q}_{H|K} \in MS(K)$ are the fluxes $\int_\Sigma \mathbf{q}_{H|K} \cdot \mathbf{n}_{\Sigma,K}\, \mathrm{d}s$. In its primary formulation [5], the boundary conditions of the cell problems were chosen independently from the local heterogeneities. An oversampling technique consisting in extending the local domain of the cell problem was proposed to reduce the impact of these boundary conditions and improve the accuracy of the method. However, the use of oversampling may induce a loss of volume balance. Static and dynamic strategies were proposed in [26] to define the boundary conditions in conservative ways. These two approaches are the ones we tested in this work and we now detail. In the static approach, the cell problem of a face $\Sigma \in \mathcal{F}_H^i$ is defined on a domain $K_{1,2} = K_1 \cup K_2$ with $K_1, K_2 \in \mathcal{K}_{H,\Sigma}$:

$$\begin{cases} \boldsymbol{\psi}_\Sigma &= -\lambda \mathbf{K} \nabla \phi_\Sigma & \text{in } K_{1,2}, \\ \text{div}(\boldsymbol{\psi}_\Sigma) &= \varepsilon_{\Sigma,K_1} w_1 & \text{in } K_1, \\ \text{div}(\boldsymbol{\psi}_\Sigma) &= \varepsilon_{\Sigma,K_2} w_2 & \text{in } K_2, \\ \boldsymbol{\psi}_\Sigma \cdot \mathbf{n} &= 0 & \text{on } \partial K_{1,2}, \end{cases} \tag{10}$$

where $\mathbf{n}$ is the unit normal vector oriented outside of the domain $K_{1,2}$. Functions w are weight functions verifying

$$w_i(x) = \frac{\theta(x)}{\int_{K_i} \theta(y)\, \mathrm{d}y}.$$

Different choices are possible for θ, the most simple being $\theta = 1$. In this work, we choose

$$\theta = (\lambda \mathbf{K}\mathbf{n}_\Sigma)\cdot \mathbf{n}_\Sigma.$$

Note that the sink and source terms along with the no-flux boundary conditions on $\partial K_{1,2}$ defined for problem (10) imply that the flux of $\boldsymbol{\psi}_\Sigma$ through Σ is equal to one. Functions ϕ_Σ are defined up to one constant. In practice, these basis functions are chosen so that the w-weighted average is equal to zero. For a face $\Sigma \in \mathscr{F}_H^D$ and $K \in \mathscr{K}_{H,\Sigma}$, the related basis function is defined in the cell K and is solution to

$$\begin{cases} \boldsymbol{\psi}_\Sigma &= -\lambda \mathbf{K}\nabla\phi_\Sigma & \text{in } K, \\ \operatorname{div}(\boldsymbol{\psi}_\Sigma) &= w & \text{in } K, \\ \boldsymbol{\psi}_\Sigma\cdot\mathbf{n} &= 0 & \text{on } \partial K\ \Sigma, \\ \boldsymbol{\psi}_\Sigma\cdot\mathbf{n} &= w_\Sigma & \text{on } \Sigma, \end{cases} \tag{11}$$

where function w_Σ is given by

$$w_\Sigma(x) = \frac{\theta(x)}{\int_\Sigma \theta(y)\,\mathrm{d}y}.$$

The static approach sometimes fails to reach a good accuracy in the computation of the fluxes since it only takes local properties into account. Hereafter, when the static approach is used to compute the basis functions, these functions will be referred to as local basis functions. The idea of the dynamic approach consists in computing a one-phase flow at the fine scale whenever new boundary conditions are defined during the simulation. The obtained velocity field v^{mono} is then used to define the boundary conditions of the cell problem. Thus, for each face $\Sigma \in \mathscr{F}_H^{iD}$ and each cell $K \in \mathscr{K}_{H,\Sigma}$, the basis functions $\boldsymbol{\psi}_{K,\Sigma}$ and $\phi_{K,\Sigma}$ are defined in the following way:

$$\begin{cases} \boldsymbol{\psi}_{K,\Sigma} &= -\lambda \mathbf{K}\nabla\phi_{K,\Sigma} & \text{in } K, \\ \nabla\cdot(\boldsymbol{\psi}_{K,\Sigma}) &= \varepsilon_{\Sigma,K} w & \text{in } K, \\ \boldsymbol{\psi}_{K,\Sigma}\cdot\mathbf{n} &= 0 & \text{on } \partial K\ \Sigma, \\ \boldsymbol{\psi}_{K,\Sigma}\cdot\mathbf{n} &= \frac{v^{\text{mono}}(x)\cdot\mathbf{n}}{\int_\Sigma v^{\text{mono}}\cdot\mathbf{n}} & \text{on } \Sigma. \end{cases} \tag{12}$$

Hereafter, the basis functions obtained with the dynamic approach are referred to as global basis functions. Let us remark that a slightly different strategy can be applied. It consists in using, simultaneously, several global pressure fields obtained with different prescribed boundary conditions. In that case the boundary conditions of the cell problems remain fixed during the simulation but the number of degrees of freedom per coarse face is increased. This strategy, as suggested in [27], helps to improve the accuracy of the method.

In this work, the cell problems (11) and (12) are discretized using a TPFA scheme.

Figures 2 and 3 show multiscale basis functions computed on a one-dimensional mesh with 1000 cells and a coarse mesh made of two coarse cells. The three multiscale basis functions obtained for a constant permeability field equal to one are represented in Figure 2. They correspond, in that case, to the hat functions used to approximate the velocity in the $\mathbb{RT}_0$ mixed finite element space. Both graphs

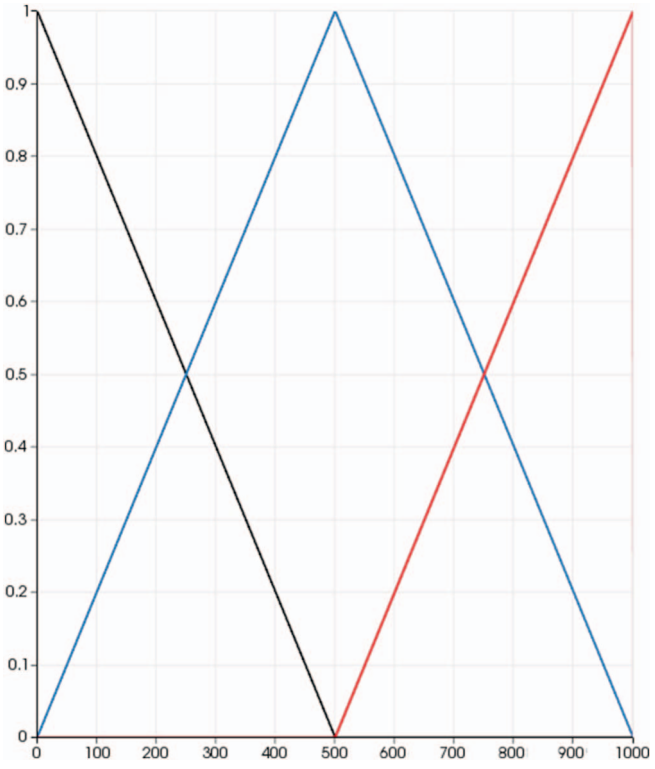

Fig. 2. 1-d basis functions computed with $\mathbf{K} = \mathbf{Id}$.

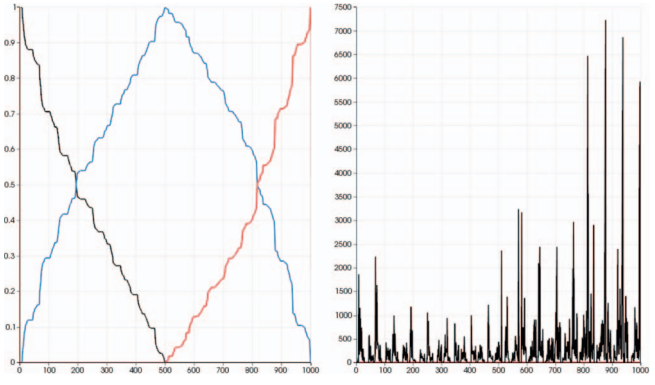

Fig. 3. On the left, 1-d basis functions computed with $\mathbf{K} = \mathrm{K}\mathbf{Id}$, K being represented on the right (in mD).

on Figure 3 depict the three multiscale basis functions and the permeability values used to compute them.

3.3 Update of the basis functions

In the IMPES or IMPIMS schemes, the total mobility λ is updated after each resolution of (15), which requires an update of the basis functions. However, the variations of λ are usually small from one time step to one another in regions far from the water saturation front. Theferefore, before a new resolution of the pressure equation, only a few basis functions need to be recomputed. To trigger these updates in cells where significant mobility variations occur, we chose the criterion proposed in [28]. Considering the quantity

$$\left(D_\lambda^n\right)_k = \frac{\lambda\left(S_k^n\right)}{\lambda\left(S_k^{\text{last}}\right)},$$

where S_k^{last} is the saturation computed during the last update of the basis functions, a threshold ε_{tol} is fixed and for each coarse cell $K \in \mathscr{K}_H$ where

$$\max_{k\in K}\left(D_\lambda^n\right)_k > 1+\varepsilon_{\text{tol}} \text{ or } \min_{k\in K}\left(D_\lambda^n\right)_k < \frac{1}{1+\varepsilon_{\text{tol}}}$$

the basis functions $\boldsymbol{\psi}_\Sigma$ and ϕ_Σ for each face $\Sigma \in \mathscr{F}_{H,K}^{i\vee D}$ are recomputed.

4 Application of the method

The MMsFE method described above was implemented using the HPC simulation framework developed at IFPEN and based on the C++ Arcane platform [29]. Mesh data structures, parallelism, I/Os and scalable linear solvers are, for instance, some core functionalites provided by this platform. In our experiments, the coarse and fine linear systems are solved using the BiCGStab solver from PETSc [30] preconditioned with an AMG preconditioner from Hypre [31]. All cell problems are solved using UMFPACK linear solvers [32]. The threshold $\varepsilon_{\text{tol}} = 0.8$ is used to trigger locally an update of the basis functions. In our implementation, at each time step, once all fluxes have been computed by one of the two methods (TPFA or MMsFE), the new water saturations are computed implicitly using a Newton-Raphson non-linear solver. The value of the time step is adjusted during the simulation according to the number of the Newton-Raphson iterations: it is increased if the number of iterations is less than a number given by the user, it remains the same if this number is exceeded while obtaining the convergence and is decreased if the convergence is not reached. This classical strategy enables the use of big time steps during the simulation but leads to update a larger number of basis functions at each time iteration. It may not be the optimal strategy for the MMsFE method since smaller time steps require fewer updates. But we adopted it for the runs carried out with the TPFA scheme and used the same sequence of time steps for the MMsFE method in order to have the solution at the same times.

The differences in the fluxes given by the TPFA method and the MMsFE method are measured by computing the relative L^2 error of the water saturations:

$$\frac{\sqrt{\sum_{k\in\mathscr{K}_h}|k|\phi_k\left(S_k^f - S_k^{ms}\right)^2}}{\sqrt{\sum_{k\in\mathscr{K}_h}|k|\phi_k\left(S_k^f\right)^2}}. \tag{13}$$

In the following section, we introduce the tenth SPE test case which has become a classical benchmark for upscaling and multiscale methods over the past years [33]. We compare, for this case, the solutions obtained with a local and global calculation of the basis functions.

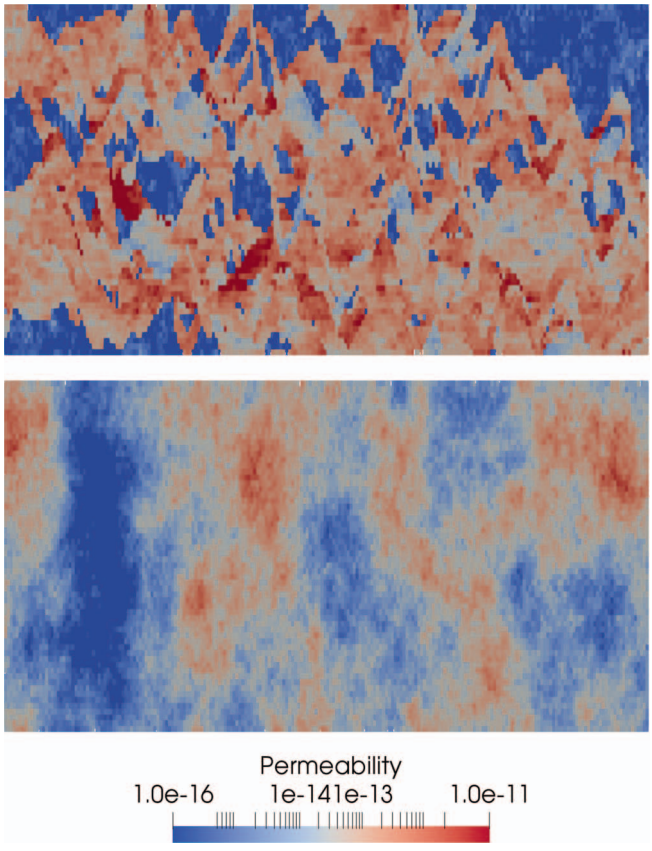

Fig. 4. Permeability in m^2 of the 1st layer (bottom) in the Tarbert formation and the 85th layer (top) in the Upper Ness formation.

4.1 SPE10 test case

The reservoir defined in the tenth SPE test case is a parallepiped made of 60 cells of size 20 ft in x-direction, 220 cells of size 10 ft in y-direction and 85 cells of size 2 ft in z-direction. Unlike the original data set, the permeability values are here taken equal to the horizontal values and truncated to 0.1 mD if they are below this threshold. This reduces the permeability contrasts but make all cells active in the resolution process. Figure 4 shows the permeability field in two layers of the Tarbert and Upper Ness formations.

Since our main objective in this paper is to quantify the influence of the permeability heterogeneities on the two different pressure solvers, we used a constant porosity value throughout the domain. Note that the contrasts of porosities here only have an impact on the conditioning of the linearized problem related to the saturation equation (9). Relative permeabilities are chosen such that

$$kr_{\text{w}} = \left(\frac{S - S_{\text{wi}}}{1 - S_{\text{wi}} - S_{or}}\right)^2 \text{ and } kr_{\text{o}} = \left(\frac{1 - S - S_{\text{or}}}{1 - S_{\text{wi}} - S_{or}}\right)^2,$$

with $S_{\text{wi}} = S_{\text{or}} = 0.2$. Oil and water viscosities are equal to 1 cp and 0.3 cp respectively. On the reservoir boundaries, the following conditions are imposed:

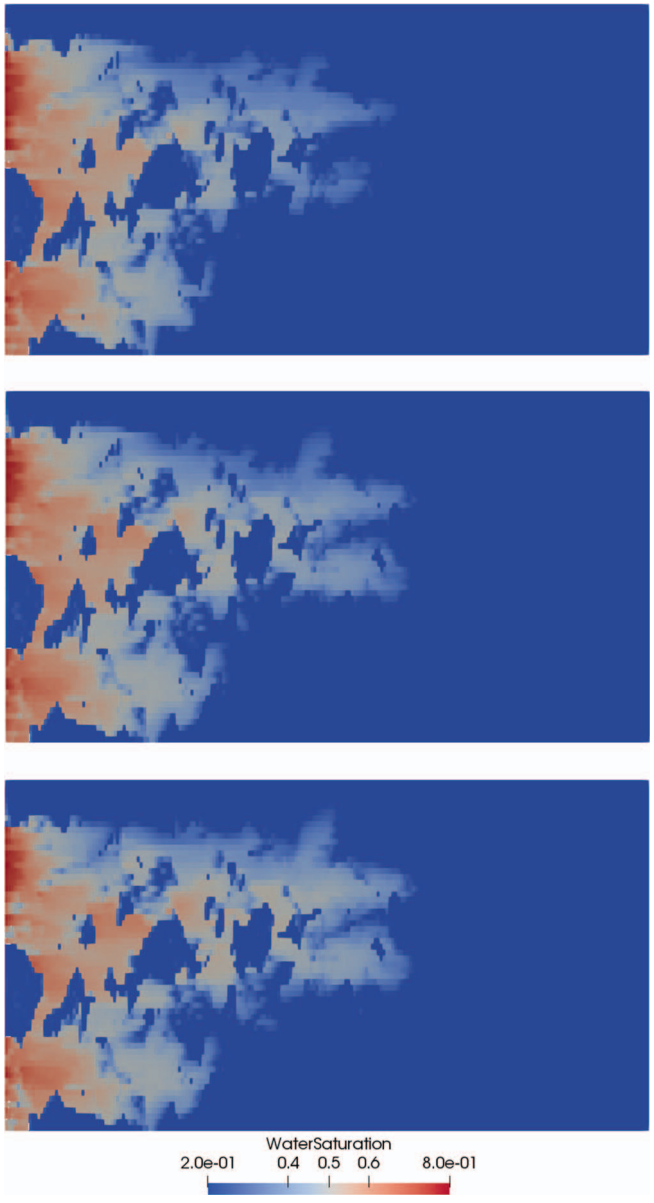

Fig. 5. Saturation solutions obtained with the TPFA method (bottom) and the MMsFE methods with global basis (middle) and local basis (top) functions.

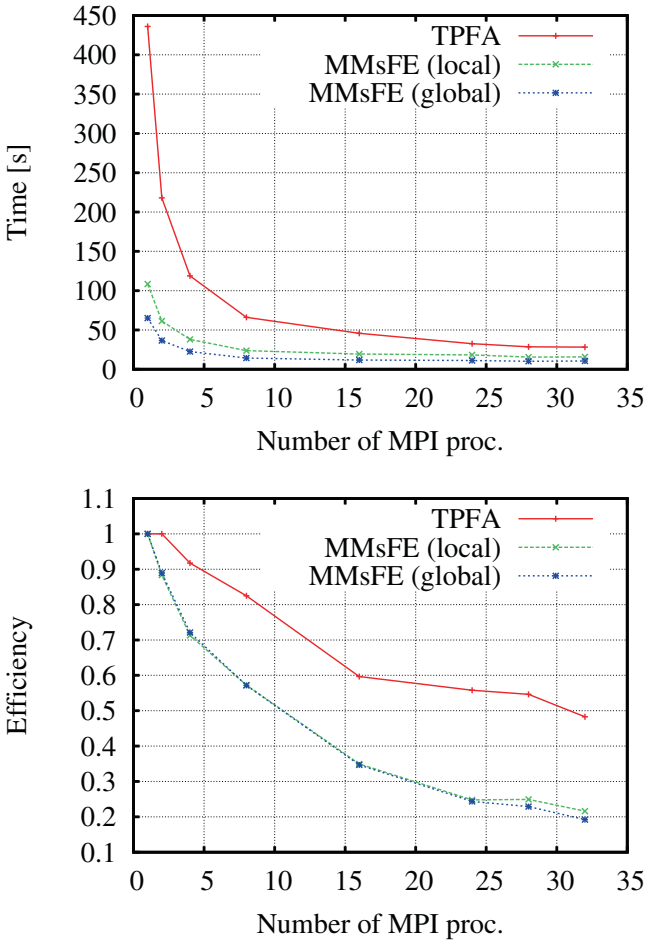

Fig. 6. CPU times and efficiency of the TPFA and MMsFE solvers.

– a fixed pressure of 1000 psi along with a water saturation equal to 1 on the border $y = 0$,
– a fixed pressure of 500 psi on the border $y = y_{\max}$,
– no flux on the other boundaries.

The computation is performed on a coarse grid of size $12 \times 20 \times 17$. The simulations were run until the water volume injected into the reservoir reached 5% of its porous volume.

Figure 5 shows the water saturations obtained with the TPFA method on the fine grid and with the MMsFE method using local basis functions and global basis functions. The use of dynamic information for the resolution of the cell problems enables the global approach to better reproduce the fine reference solution compared to the local one. The relative L^2 error defined in (13) is indeed equal to 6.4% for the global method and to 11.6% for the local one.

5 Parallel computation

This section focuses on the parallel implementation of the MMsFE method and highlights some key points to ensure performance and scalability on clusters. The performances of the MMsFE method are analyzed through the tenth SPE test case introduced in the previous section. The number of MPI processes used for this test case is relatively small but scalability issues can already be found out on that example.

The results presented hereafter were obtained thanks to simulations performed on the IFPEN cluster. On this machine, each SMP node is composed of 2 Intel Xeon octo-cores processors E7-2670 (2.6 GHz). All 378 nodes are interconnected with QDR infiniband links (40 Gbit/s) and offer a peak of 110 Teraflops.

5.1 Pure MPI approach

A classical SPMD domain decomposition is used to distribute the calculation work into multiple tasks. Each parallel task works on a subdomain of the reservoir and the communications between the subdomains are done using the standard MPI interface.

The upper plot in Figure 6 shows the CPU times obtained with the classical TPFA pressure solver and with the local and global MMsFE solvers as a function of the numbler of MPI processes. Here the acceleration rises up to a factor 9 but falls to 4 when more than 16 MPI processes are used. The lower plot in Figure 6 shows that a better efficiency is achieved with the TPFA solver than with the MMsFE solvers.

5.1.1 Balancing the calculations of the basis functions

The performances of multiscale methods highly depend on the number of basis functions that should be recomputed at each time step. As mentioned in Section 3.3, we used a criterion based on the variations of the total mobility to trigger an update of the basis functions defined in the regions where these changes occur. Thus, the parallel performance of the MMsFE method depends on the way the saturation front is distributed among the processors. In the tenth SPE test case introduced in the previous paragraph, the water flows from the boundary $Y_{\min}$ to the boundary $Y_{\max}$ and this direction should be taken into account when creating the domain decomposition. In the case depicted in Figure 7 only a small part of the processors will be used to compute the basis functions whereas a better scalability can be achieved in the case represented in the right plots of Figure 7. With unstructured mesh, it is still possible to use weights in the connectivity graph to guide the partition of the domain in some directions. But, in oil reservoir simulations, the flow can be localized for some time around a few wells and, in that case, one has to resort to a dynamic load balancing of the mesh. Figure 8 shows the CPU time used for the computation of the local and global basis functions as a function of the MPI processes in the tenth SPE test case.

5.1.2 Balancing the resolution of the coarse linear system

A second issue comes from the number of MPI processes used to solve the coarse linear system. Since multiscale methods enable to agglomerate a significant number of fine cells into a coarse one, the resulting linear system can be small in comparison to the number of MPI processes which are available, leading to scalability problems as illustrated in Figure 9 for the tenth SPE case. Here, the CPU time increases when more than 16 processes are used. On the other hand, the saturation solver which only works on the fine grid, keeps a good scaling when increasing the number of MPI processes as shown in Figure 10.

A first solution to improve the speed-ups related to this step of the method could consist in using a hybrid parallel programming technique which uses both MPI and threads processes, in order to reduce the number of MPI processes

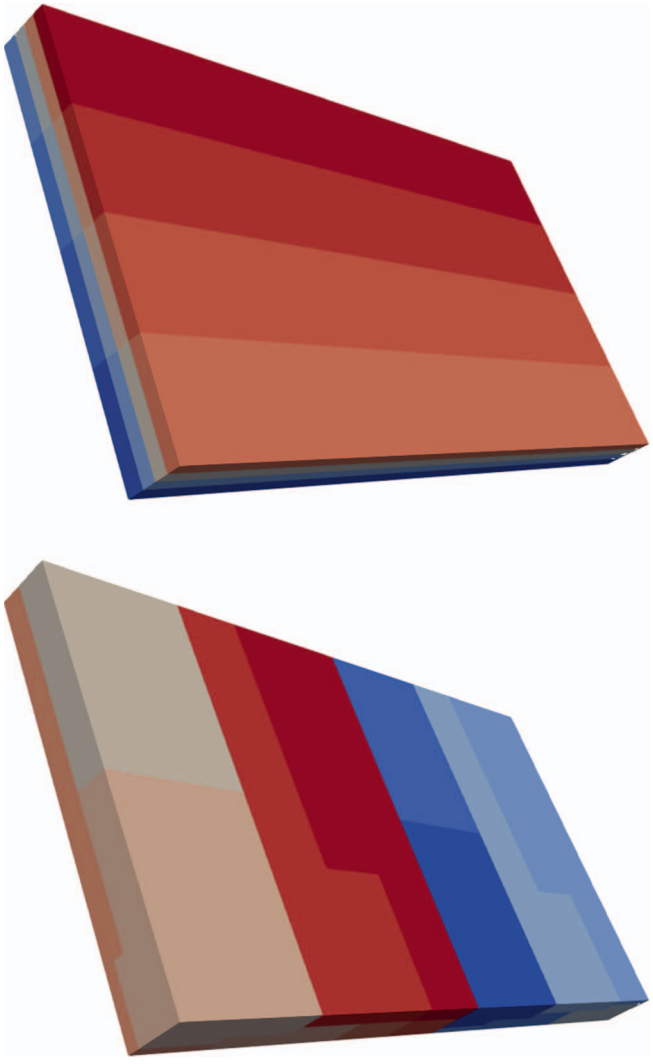

Fig. 7. Two domain decompositions: on the top, the partition is made according to direction X and Z whereas no direction is favoured on the bottom.

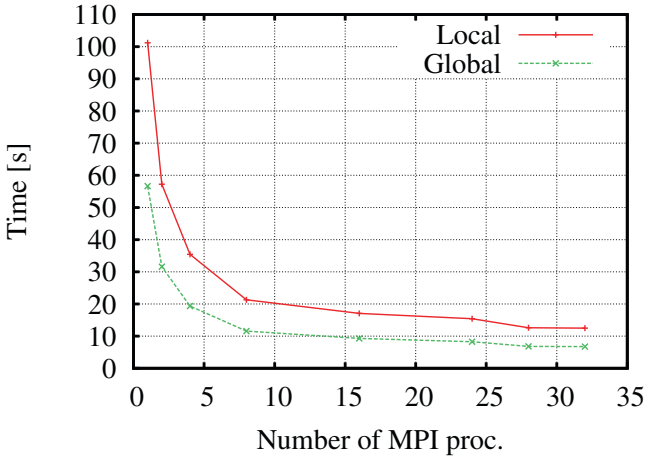

Fig. 8. CPU time used for the calculation of the basis functions.

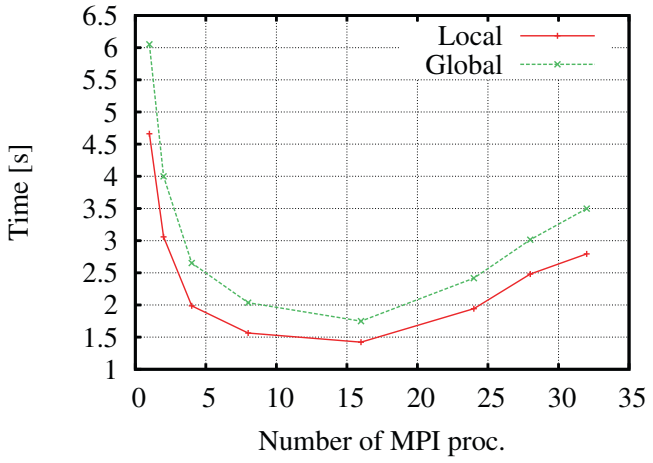

Fig. 9. CPU time used for the resolution of the coarse linear system.

assigned to the coarse linear solver [34]. But then, the question of the optimal number of MPI processes that should be dedicated to this solver brings up and, in case of multiscale methods, this number could be significantly different from the one required by the other parts of the simulator. Keeping too many processes will reduce the performances as it was previously pointed out, since communications through the network will represent a large part of the time related to the iterative process. Conversely a reduced number of processes would penalize the other parts of the simulator. Concerning the implementation, hybrid parallel programming requires some effort since all existing software components of the simulator should be threadsafe. Thus, for these reasons and as suggested in [35], we propose in the next section a two-level MPI approach to reduce the distribution of our coarse linear systems. Note that this approach remains complementary to a hybrid parallel programming model.

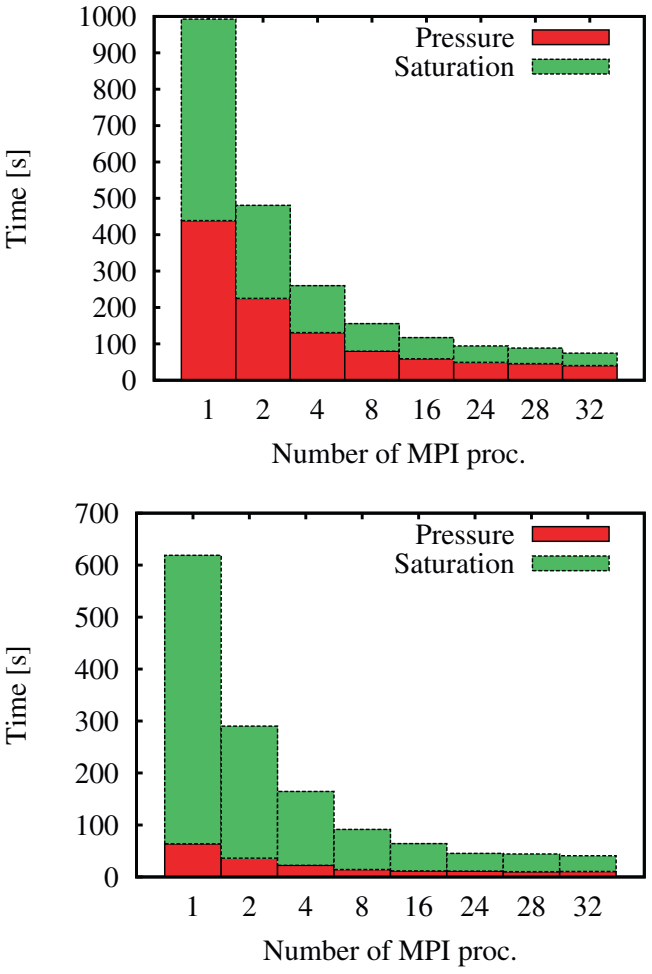

Fig. 10. CPU time used for the resolution of the pressure and saturation equations of the TPFA method on the top and the MMsFE with multiscale global basis functions on the bottom.

5.2 A two-level MPI approach

The scalability problem related to the coarse pressure linear system and mentioned in the previous section is similar to the issue pointed out in [35] for coarse operations in multigrid staging. In this work, the authors resort to a second level MPI communicator set on a small part of the machine to circumvent this difficulty. The role of this second communicator consists in gathering the coarse data, processing them and sending them back to the first level MPI communicator. By that way less computing resources are used. The same strategy was adopted here by using a redistribution library for linear systems. This library, called ALEPH for Linear Algebra Extended to Hybrid Parallelism [36] is said to be "hybrid" in the sense that it uses different MPI communicators. The redistribution is done in a smart way using the topology of the machine. SMP nodes are filled in a compact manner in order to minimize costly operations with the second level MPI-communicator. Moreover, cache optimizations have been performed to minimize memory communications between the communicator levels. For instance, the global indexes of the linear systems are sent only once during the simulation. To solve the coarse pressure system, the ALEPH library collects, on a user-defined subset of MPI processes, all contributions to the matrix and the right-hand side that have been computed by all MPI processes in the first MPI-communicator. Then, the compacted coarse pressure system is solved using only this subset of MPI processes in the second level MPI-communicator. This resolution may be carried out by any MPI-based library such as PETSc, HYPRE, Trilinos, etc. In this work, the PETSc library was used. At the end of the resolution, the solution is distributed over all MPI processes. The ALEPH workflow is summarized in Figure 11 for an example of machine with two nodes composed of four processors.

For the local approach, Figure 12 shows the CPU time taken by the coarse pressure solver according to the number of MPI processes when the ALEPH library is used or not. Here the redistribution operated by ALEPH enables to keep the CPU time below the limit of 1.5 s when more than 16 MPI processes are used. The same trend can be observed in Figure 13 with the global approach.

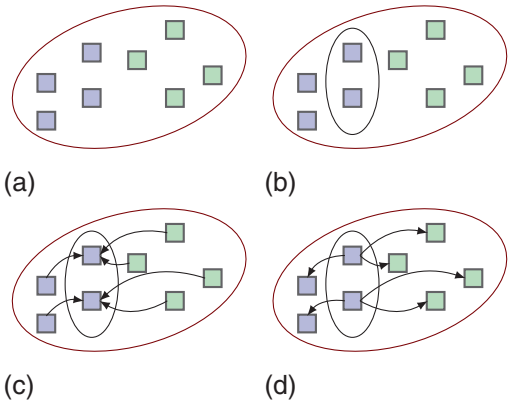

Fig. 11. ALEPH library workflow. (a) First-level MPI, (b) second-level MPI, (c) collect and solve step and d) redistribution step. Here, colors refer to processors over nodes.

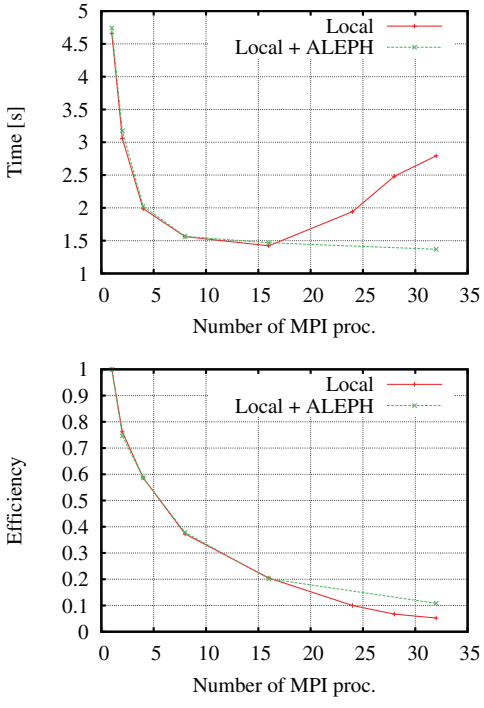

Fig. 12. CPU time and efficiency of the MMsFE method with local basis functions using or not the ALEPH library.

Since this reduction on a small subset of MPI processes is only performed for the resolution of the coarse linear system, this strategy enables the other parts of the resolution process, and in particular the saturation solver, to run on a wider number of MPI processes. Nevertheless, a significant part of the available nodes remains unused during that stage and to make the most of these calculation resources, other programming techniques such as asynchronism should be studied.

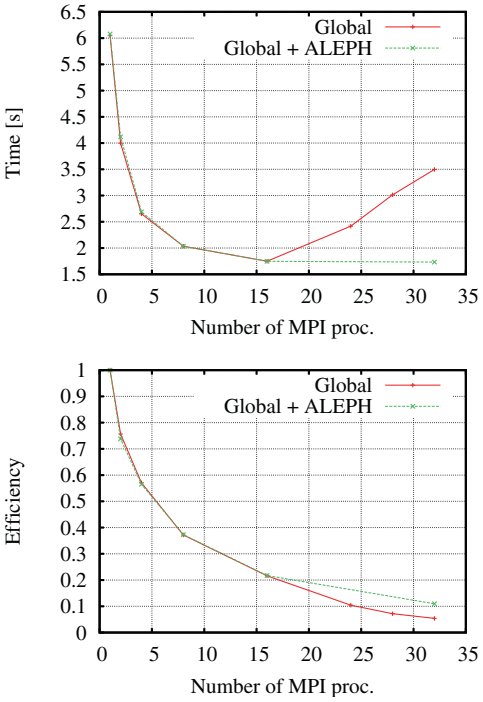

Fig. 13. CPU time and efficiency of the MMsFE method with global basis functions using or not the ALEPH library.

6 Another example of speed-ups obtained with the MMsFE method

The size of the grid of the tenth SPE test case, presented in Section 4.1, is not very large in view of the calculation resources offered by the clusters at the present time. We therefore introduce here a second test case with a larger grid size where the speed-ups provided by the MMsFE method turn out to be more significant.

6.1 Description

That case takes up again the main parameters of the tenth SPE example. Only the dimensions of the domain and the permeability field change. Here, the fine-grid's size is $256 \times 256 \times 256$. A geostatistical facies realization was created on that grid. Two facies were used to model high and low permeable regions within the reservoir with proportions 0.6 and 0.4 respectively. Figure 14 shows the obtained permeability field. The size of the coarse grid is here equal to $32 \times 32 \times 32$. The global approach is used to compute the cell problems.

6.2 Results

The pressure solutions given by the TPFA and MMsFE methods are represented in Figure 15. The water saturations obtained with the fluxes related to these two methods are shown in Figure 16. Here the relative L^2 distance between these two solutions is equal to 3%.

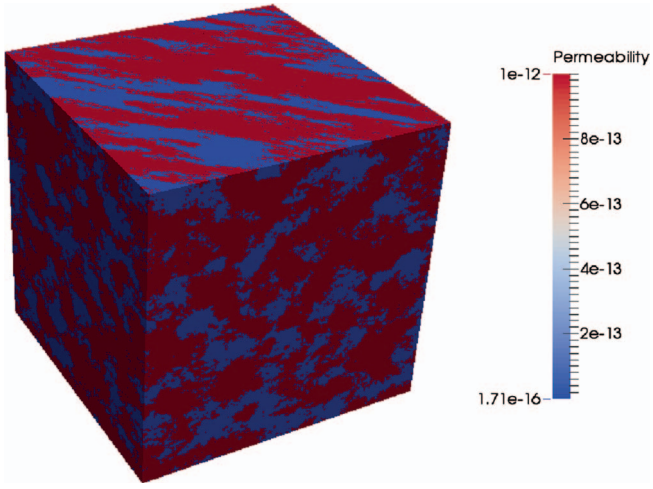

Fig. 14. Permeability values of the second example (in m^2).

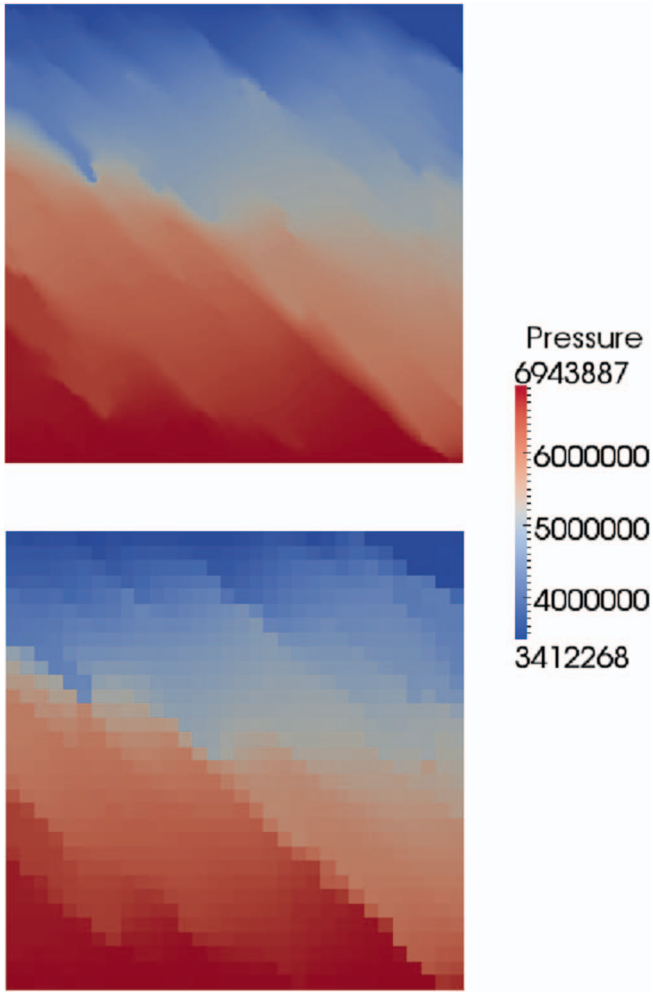

Fig. 15. On the plane $z = z_{max}$, pressure solutions (in Pa) of the TPFA method on the top and the MMsFE method on the bottom.

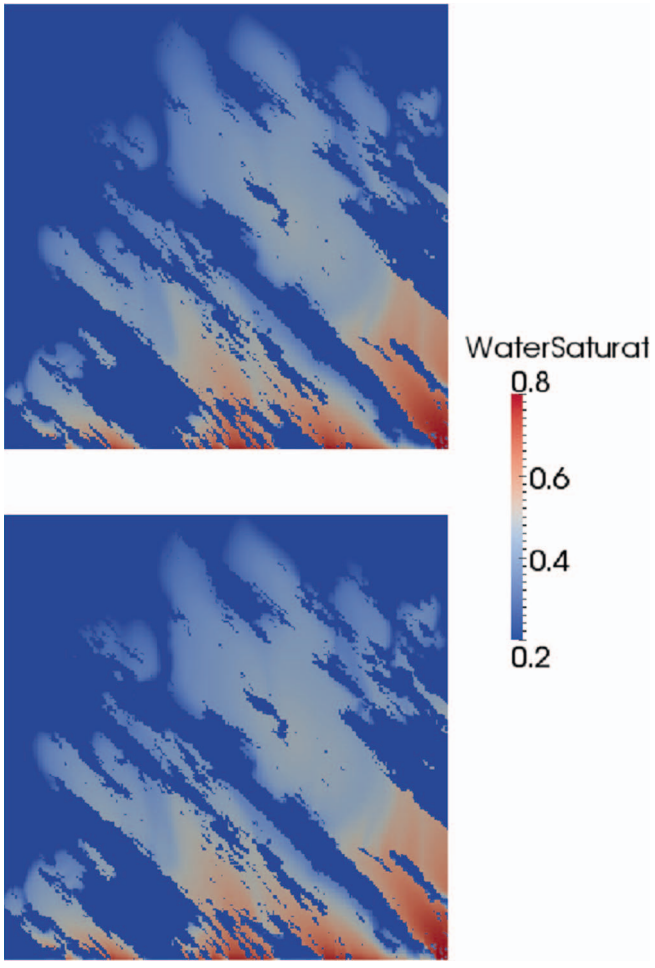

Fig. 16. On the plane $z = z_{max}$, water saturation solutions of the TPFA method on the top and the MMsFE method on the bottom.

All simulations were performed using 256 MPI processes over 16 nodes. Table 1 gives the CPU times obtained with the TPFA and MMsFE pressure solvers. Without the use of the ALEPH library, all MPI processes are used for the resolution of the coarse linear systems. Several ALEPH configurations have been tested up to the case where 128 processes are dedicated to the resolution of these systems. The processors used for each configuration are selected taking into account the topology of the nodes. For instance, the configuration using 128 processes uses exactly 8 nodes, the configuration using 32 processes uses 2 nodes and so on. The best configuration leads to a speedup of nearly 37 with 16 processes, which corresponds to solve coarse problem on one single node. These results tend to suggest that significant accelerations can be achieved during the resolution of the coarse system by limiting extra-node communications, which was carried out here by reducing the numbers of MPI processes and nodes at the same time.

Table 2 gives the CPU times of the ALEPH solver using 16 processes spread over one to 16 nodes. We show that the

Table 1. CPU times of the TPFA and MMsFE pressure solvers using several ALEPH configurations.

Method	Time (s)	Speedup
TPFA	1137	1
MMsFE	155.8	7.3
MMsFE ALEPH(1)	192.8	5.9
MMsFE ALEPH(2)	108.2	10.5
MMsFE ALEPH(4)	67.7	16.8
MMsFE ALEPH(8)	47.3	24
MMsFE ALEPH(16)	30.4	37.4
MMsFE ALEPH(32)	31.2	36.4
MMsFE ALEPH(64)	52.8	21.5
MMsFE ALEPH(128)	9.6	11.4

Table 2. CPU times of the ALEPH solver for various distributions of the 16 processes over the nodes.

Distribution	Time (s)	Overload
1 node/16 procs	25.69	1
2 nodes/8 procs	28.45	1.11
4 nodes/4 procs	33.36	1.30
8 nodes/2 procs	36.03	1.40
16 nodes/1 proc	39.51	1.54

best configuration leads to use only one node and to maximize the intra-node MPI communications and that performances decrease when more nodes are used.

The total simulation time, including the pressure and saturation resolution times, is equal to 1255 s with the fine TPFA scheme (1137 s for the pressure equation, 118 s for the saturation equation) and to 156 s with the MMsFE solver (30 s for the pressure equation, 126 s for the saturation equation), leading to a global speedup of 8.

7 Conclusion

As shown in this work, two factors can limit the scalability of the MMsFE method: the calculations of the basis functions which may be unbalanced between the subdomains and the size of the global coarse linear system which can be small compared to the number of allocated calculation resources. Different solutions have been mentioned in Section 5.1.1 to tackle the first problem. But our works were mainly focused on the second one for which we propose a two-level MPI approach.

Our approach consists in reducing the number of MPI processes to a few ones which are concentrated on a few nodes when solving the coarse pressure system. This strategy, which was carried out here by using the ALEPH library, enables to decrease temporarily the number of MPI processes for that step of the MMsFE method while using the whole set of MPI processes for the other parts of the simulator. Compared to a hybrid programming model which may affect other parts of the application and thus may require more efforts in terms of implementation, this reduction and clustering of the MPI processes is only performed within the MMsFE method.

This technique leads to significant speed-ups for the two-phase model and for the mixed multiscale method considered in this work. But other mixed methods, like the ones introduced in [37] and [38] and based on the Generalized Multiscale Finite Element Method, could be used as well. These last two methods feature, in particular, the possible use of more basis functions in each coarse element and can ensure spectral and mesh convergences.

The next step will be to test the dynamic load balancing for the calculation of the basis functions along with the two-level MPI strategy. In view of reducing computing times for oil and gas reservoir simulations, well boundary conditions as well as a more complicated three-phase Black-Oil model will be considered for the following of this study. The results of these tests will thus turn out the potential of acceleration that the MMsFE method can provide for industrial applications when used in combination with these parallelism techniques.

References

1 Farmer C.L. (2002) Upscaling: a review, *Int. J. Numer. Methods Fluids* **40**, 1–2, 63–78 https://doi.org/10.1002/fld.267.
2 Durlofsky L.J. (2005) Upscaling and gridding of fine scale geological models for flow simulation, in: *8th International Forum on Reservoir Simulation Iles Borromees, Stresa, Italy*, Vol. 2024.
3 Correia M.G., Maschio C., Schiozer D.J. (2018) Flow simulation using local grid refinements to model laminated reservoirs, *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles* **73**, 5.
4 Hou T.Y., Wu X.-H. (1997) A multiscale finite element method for elliptic problems in composite materials and porous media, *J. Comput. Phys.* **134**, 1, 169–189.
5 Chen Z., Hou T.Y. (2003) A mixed multiscale finite element method for elliptic problems with oscillating coeffcients, *Math. Comput.* **72**, 541–576.
6 Jenny P., Lee S.H., Tchelepi H.A. (2003) Multi-scale finite-volume method for elliptic problems in subsurface flow simulation, *J. Comput. Phys.* **187**, 1, 47–67.
7 Moyner O., Lie K.-A. (2014) A multiscale two-point fluxapproximation method, *J. Comput. Phys.* **275**, 273–293.
8 Franc J., Jeannin L., Debenest G., Masson R. (2017) FV-MHMM method for reservoir modeling, *Comput. Geosci.* **21**, 5, 895–908.
9 Efendiev Y., Hou T.Y. (2009) *Multiscale finite element methods: theory and applications*, Vol. 4, Springer Science & Business Media
10 Kippe V., Aarnes J.E., Aarnes J.E., Lie K.-A. (2008) A comparison of multiscale methods for elliptic problems in porous media flow, *Comput. Geosci.* **12**, 3, 377–398.
11 Aziz K., Settari A. (1979) *Petroleum reservoir simulation*, Chapman & Hall.

12 Ho A. (2012) A parallel multiscale mixed finite-element method for the matlab reservoir simulation toolbox, *Master's Thesis*, NTNU – Trondheim.
13 Lie K.-A., Krogstad S., Ligaarden I.S., Natvig J.R., Nilsen H.M., Skaflestad B. (2012) Open-source Matlab implementation of consistent discretisations on complex grids, *Comput. Geosci.* **16**, 2, 297–322.
14 Manea A.M., Sewall J., Tchelepi H.A. (2015) Parallel multiscale linear solver for highly detailed reservoir models, *SPE Reservoir Simulation Symposium.*
15 Bastian P., Engwer C., Göddeke D., Iliev O., Ippisch O., Ohlberger M., Turek S., Fahlke J., Kaulmann S., Müthing S., Ribbrock D. (2014) *Euro-Par 2014: Parallel Processing Workshops,* Vol. 8806, chapter, Flexible PDE Solvers, *Numerical Methods and Applications*, **8806**, 530–541.
16 Leverett M.C. (1939) Flow of oil-water mixtures through unconsolidated sands, *Trans. AIME* **132**, 1, 149–171.
17 Farkas E. (1998) Linearization techniques of reservoir-simulation equations: fully implicit cases, *SPE J.* **3**, 4, 316–323.
18 Coats K.H. (2000) A note on IMPES and some IMPES-based simulation models, *SPE J.* **5**, 3, 245–251.
19 Coats K.H. (2003) IMPES stability: selection of stable timesteps, *SPE J.* **8**, 2, 181–187.
20 Whittaker E.T., Robinson G. (1967) The calculus of observations: a treatise on numerical mathematics, Dover Publications.
21 Eymard R., Gallouët T., Herbin R. (2000) Finite volume methods, *Handbook of numerical analysis* **7**, 713–1018.
22 Aavatsmark I., Eigestad G., Mallison B., Nordbotten J. (2008) A compact multipoint flux approximation method with improved robustness, *Numer Methods Partial Differ. Equ.* **24**, 5, 1329–1360.
23 Gyrya V., Lipnikov K. (2008) High-order mimetic finite difference method for diffusion problems on polygonal meshes, *J. Comput. Phys.* **227**, 20, 8841–8854.
24 Ponting D.K. (1989) Corner point geometry in reservoir simulation, in: *ECMOR I – 1st European Conference on the Mathematics of Oil Recovery.*
25 Hauge V.L., Lie K.A., Natvig J.R. (2012) Flow-based coarsening for multiscale simulation of transport in porous media, *Comput. Geosci.* **162**, 391–408.
26 Aarnes J.E. (2004) On the use of a mixed multiscale finite element method for greater flexibility and increased speed or improved accuracy in reservoir simulation, *Multiscale Model. Simul.* **2**, 3, 421–439.
27 Aarnes J.E., Efendiev Y., Jiang L. (2008) Mixed multiscale finite element methods using limited global information, *Multiscale Model. Simul.* **72**, 655–676.
28 Jenny P., Lee S.H., Tchelepi H.A. (2004) Adaptive multiscale finite-volume method for multiphase flow and transport in porous media, *Multiscale Model. Simul.* **3**, 1, 50–64.
29 Grospellier G., Lelandais B. (2009) The ARCANE development framework, in: *8th Workshop on Parallel/High-Performance Object-Oriented Scientific Computing*, ACM, 1–11.
30 Balay S., Gropp W.D., McInnes L.C., Smith B.F. (1997) Efficient management of parallelism in object oriented numerical software libraries, in: *Modern software tools in scientific computing*, Springer, pp. 163–202.
31 Chow E., Cleary A.J., Falgout R.D. (1998) Design of the hypre preconditioner library, in: *SIAM Workshop on Object-Oriented Methods for Intero-Operable Scientific and Engineering, Computing.*
32 Davis T.A. (2004) Algorithm 832: Umfpack v4.3 – an unsymmetricpattern multifrontal method, *ACM Trans. Math. Softw.* **30**, 2, 196–199.
33 Christie M.A., Blunt M.J. (2001) Tenth SPE comparative solution project: a comparison of upscaling techniques, *SPE Reserv. Evalu. Eng.* **4**, 4, 308–317.
34 Ouaki F. (2013) Etude de schémas multi-échelles pour la simulation de réservoir, *PhD Thesis*, Ecole Polytechnique.
35 Jaure S., Moncorge A., de Loubens R. (2014) Reservoir simulation prototyping platform for high performance computing, in: *SPE Large Scale Computing and Big Data Challenges in Reservoir Simulation Conference and Exhibition.*
36 Camier J.S. (2015) Improving performance portability and exascale software software productivity with the ∇ numerical programming language, *Exascale Applications and Software Conference.*
37 Chung E.T., Efendiev Y., Lee C.S. (2015) Mixed generalized multiscale finite element methods and applications, *Multiscale Model. Simul.* **13**, 1, 338–366.
38 Chung E.T., Efendiev Y., Leung W.T. (2000) *Constraint energy minimizing generalized multiscale finite element method in the mixed formulation.* Preprint arXiv:1705.05959.

Appendix A

Fine scale discretization

Finite volume methods (21) are widely used for groundwater and reservoir simulations due to the easiness of their implementation and the mass conservation property satisfied by the computed fluxes. In the following, we first remind the reader of the two-point flux approximation used in this work to approximate the pressure gradient in problem (4) and then present the discretization of the saturation equation (5).

A.1 Pressure equation

Integrating (8) over a cell $k \in \mathscr{K}_h$ and applying the divergence theorem leads to

$$\begin{aligned}\int_k \nabla \cdot \mathbf{v}\,\mathrm{d}x &= \sum_{\sigma\in\mathscr{F}_{h,k}} \int_\sigma \mathbf{v}\cdot\mathbf{n}_{\sigma,k}\,\mathrm{d}s \\ &= \sum_{\sigma\in\mathscr{F}^i_{h,k}} \int_\sigma \mathbf{v}\cdot\mathbf{n}_{\sigma,k}\,\mathrm{d}s + \sum_{\sigma\in\mathscr{F}^D_{h,k}} \int_\sigma \mathbf{v}\cdot\mathbf{n}_{\sigma}\,\mathrm{d}s.\end{aligned}$$

Let $k \in \mathscr{K}_h$, $\sigma \in \mathscr{F}^i_{h,k}$ and $l \in \mathscr{K}_{h,\sigma}$ be the cell facing k next to σ. The diffusive fluxes are approximated using a two-point scheme for the gradient term. Thus

$$\int_\sigma \mathbf{v}(P^{n+1},S^n)\cdot\mathbf{n}_{\sigma,k}\,\mathrm{d}s;\ \lambda^i_\sigma\, T^i_\sigma \left(P^{n+1}_k - P^{n+1}_l\right) = \Phi^i_{k,\sigma},$$

where the transmissivity T^i_σ is computed using a harmonic average

$$T^i_\sigma = |\sigma| \left(\frac{|x_k - x_\sigma|}{\mathbf{K}_k} + \frac{|x_l - x_\sigma|}{\mathbf{K}_l}\right)^{-1}$$

and the total mobility λ_σ^i is equal to the cell mobility which is upstream to σ with respect to the pressure gradient:

$$\lambda_\sigma^i = \begin{cases} \lambda\left(S_k^n\right) & \text{if } P_k^{n+1} \geq P_l^{n+1}, \\ \lambda\left(S_l^n\right) & \text{otherwise.} \end{cases}$$

Similarly, for each boundary face $\sigma \in \mathscr{F}_h^D$ and each cell k such that $k \in \mathscr{K}_{h,\sigma}$, the flux is given by

$$\int_\sigma \mathbf{v} \cdot \mathbf{n}_\sigma \, \mathrm{d}s \simeq \lambda_\sigma^b \, T_\sigma^b \left(P_k^{n+1} - P_b(x_\sigma)\right) = \Phi_{k,\sigma}^b,$$

where

$$T_\sigma^b = |\sigma| \, \frac{\mathbf{K}_k}{|x_k - x_\sigma|}$$

and the total mobility λ_σ^b is such that

$$\lambda_\sigma^b = \begin{cases} \lambda\left(S_k^n\right) & \text{if } P_k^{n+1} \geq P_b(x_\sigma), \\ \lambda(S_b(x_\sigma)) & \text{otherwise.} \end{cases}$$

Finally, the discrete pressure equation for one cell $k \in \mathscr{K}_h$ reads

$$\sum_{\sigma \in \mathscr{F}_{h,k}^i} \Phi_{k,\sigma}^i + \sum_{\sigma \in \mathscr{F}_{h,k}^D} \Phi_{k,\sigma}^b = 0. \tag{14}$$

This set of equations forms a linear system whose matrix has a strictly dominating diagonal, thus admitting a unique solution P^{n+1}.

A.2 Saturation equation

An upwind approximation of the fluxes is also used for equation (9). By integrating this equation over one cell $k \in \mathscr{K}_h$ and one time step, we obtain

$$S_k^{n+1} = S_k^n - \frac{\Delta t}{|k|\phi_k} \left(\sum_{\sigma \in \mathscr{F}_{h,k}^i} F_{w,\sigma}^{i,\star} + \sum_{\sigma \in \mathscr{F}_{h,k}^D} F_{w,\sigma}^{b,\star} \right), \tag{15}$$

where the discrete fluxes $F_{w,\sigma}^{i,\star}$ and $F_{w,\sigma}^{b,\star}$ are defined using an upstream scheme defined as follows:

$$F_{w,\sigma}^{i,\star} = f_w\left(S_k^\star\right) \left(\Phi_{k,\sigma}^i\right)^+ + f_w\left(S_l^\star\right) \left(\Phi_{k,\sigma}^i\right)^-$$

and

$$F_{w,\sigma}^{b,\star} = f_w\left(S_k^\star\right) \left(\Phi_{k,\sigma}^b\right)^+ + f_w(S_b(x_\sigma)) \left(\Phi_{k,\sigma}^b\right)^-$$

where $a^+ = \max(a,\ 0)$ and $a^- = \min(a,\ 0)$. Here, the exponent $\star$ refers either to time step n or $n+1$ depending on the use of an IMPES or IMPIMS scheme.