Adaptive linear solution process for single-phase Darcy flow

Ani Anciaux-Sedrakian, Laura Grigori, Zakariae Jorti, Soleiman Yousef

Oil & Gas Science and Technology – Rev. IFP Energies nouvelles **75**, 54 (2020)

https://doi.org/10.2516/ogst/2020050

Available online at:
ogst.ifpenergiesnouvelles.fr

REGULAR ARTICLE OPEN ACCESS

Adaptive linear solution process for single-phase Darcy flow

Ani Anciaux-Sedrakian[1], Laura Grigori[2], Zakariae Jorti[1,2,*], and Soleiman Yousef[1]

[1] IFP Energies nouvelles, 1 et 4 avenue de Bois-Préau, 92852 Rueil-Malmaison Cedex, France
[2] INRIA Paris, Alpines, and Sorbonne Université, CNRS UMR 7598, Laboratoire Jacques-Louis Lions, 75005 Paris, France

Received: 17 December 2019 / Accepted: 23 June 2020

Abstract. This article presents an adaptive approach for solving linear systems arising from self-adjoint Partial Differential Equations (PDE) problems discretized by cell-centered finite volume method and stemming from single-phase flow simulations. This approach aims at reducing the algebraic error in targeted parts of the domain using *a posteriori* error estimates. Numerical results of a reservoir simulation example for heterogeneous porous media in two dimensions are discussed. Using the adaptive solve procedure, we obtain a significant gain in terms of the number of time steps and iterations compared to a standard solve.

1 Introduction

In this paper, we focus on solving large sparse linear systems of equations that arise from solving partial differential equations stemming from porous media flow models. These linear systems are ill-conditioned due to data anisotropy and heterogeneity. They are in general solved by using iterative Krylov subspace solvers [1] and the solution phase is generally the most memory and time consuming part of the simulation and can even contribute to 80% of the simulation time. With iterative methods, an approximate solution is computed by taking into account the tolerance of the convergence chosen by the user. The solver is stopped when the tolerance threshold is reached. The numerical efficiency of these methods is strongly related to the properties of the linear systems handled. In this article, we explore a faster way to solve those systems during simulations by focusing on the study of *a posteriori* error estimators as an indicator of the complexity of the systems and as a means of providing guidance during the solve. Here, we propose an efficient adaptive procedure for solving linear systems stemming from finite-volume discretization of PDEs. In earlier works, adaptivity has been used for solving algebraic systems in the multigrid context. Adaptive algebraic multigrid methods based on aggregation, which can be applied as a solver or as a preconditioner, have been devised in [2]. In [3], an algorithm based on error estimates in a finite element framework is designed to generate aggregations adaptive to the matrix and the right-hand side as well. In [4], an algebraic multigrid method based on adaptive construction of a multilevel hierarchy is proposed to solve linear systems of weighted graph Laplacians and also discretized second order elliptic partial differential equations. For the ease of presentation, the following section starts by introducing a steady model of single phase flow in porous media, then an unsteady model of the same problem extends the scope of the experimentation (in Sect. 5.2.1).

This article is structured as follows. Section 2 introduces the steady problem of single phase flow in porous media with finite volume discretization, Section 3 formulates the starting hypothesis and presents some elements related to the field of *a posteriori* error analysis that justify the need for an adaptive solving procedure. Section 4 provides details of that error reducing approach. Finally, the numerical results of the procedure are shown in Section 5.

1.1 Some notations

Let $\Omega \subset \mathbb{R}^d$, $1 \leq d \leq 3$ be a polytopal domain (open, bounded and connected set). We denote by $\overline{\Omega}$, Ω^{o}, $\partial\Omega$ and $\mathcal{T}_h$ the closure, interior, boundary and a matching simplicial mesh of Ω, respectively. We consider that the domain Ω can be decomposed into n non-overlapping Lipschitz polyhedra K_i such that $K_i^{\mathrm{o}} \cap K_j^{\mathrm{o}} = \emptyset$ and $\bigcup_i \overline{K}_i = \overline{\Omega}$. For a polyhedron K_i, h_i designates its diameter, $|K_i|$ its d-Lebesgue measure and $V(i)$ the set of indices of neighboring polyhedra. This tessellation of Ω is denoted $\mathcal{T}_h$ with $h = \max_i h_i$.

Let us denote by $\mathcal{F} = \mathcal{F}_{\mathrm{int}} \cup \mathcal{F}_{\mathrm{ext}}$ the set of mesh edges where $\mathcal{F}_{\mathrm{int}}$ (resp. $\mathcal{F}_{\mathrm{ext}}$) is the set of inner (resp. boundary) egdes. For a mesh element K_i, $\mathcal{F}_i$ is the set of all its edges, and for an edge $\sigma \in \mathcal{F}_i$, $|\sigma|$ denotes the $(d-1)$-Lebesgue measure of σ while $\boldsymbol{n}_{\sigma,i}$ and $d_{i,\sigma}$ respectively denote the unit normal vector to the edge σ and outward to K_i, and the distance between the cell center of K_i and the edge center of σ. We use the standard notation $L^2(\Omega)$ for the space of

* Corresponding author: zjorti@hotmail.fr

integrable functions on Ω. For a vector w of length $n \in \mathbb{N}$ and a subset $L \subset [\![1, n]\!]$, we denote by w_L the restriction of w to its components whose indexes belong to L.

2 Cell-centered finite volume discretization

We first introduce the steady single phase flow model problem in a simple cell-centered finite volume discretization, then define the resulting linear system of algebraic equations and present the key assumption that motivates the need for an adaptive solving procedure.

Consider the problem that consists in seeking $\underline{p} : \Omega \to \mathbb{R}$ such that:

$$\begin{cases} -\nabla \cdot (\underline{\mathbf{K}} \nabla \underline{p}) = \underline{f} & \text{in } \Omega, \\ \underline{p} = 0 & \text{on } \partial\Omega, \end{cases} \tag{1}$$

where $\underline{f} : \Omega \to \mathbb{R}$ is a source term in $L^2(\Omega)$, and $\underline{\mathbf{K}} = \kappa \mathbf{I}$ is an uniformly bounded and positive definite permeability tensor. For the sake of simplicity we assume that $\underline{f}$ and κ are cellwise constant with respect to the mesh $\mathcal{T}_h$, and denote by f_i and κ_i their values on a mesh element K_i. The existence of a solution and the convergence of the scheme were proven in [5]. We refer the reader to the latter book for a more rigorous framework of the discretization by finite volume method.

By integrating the law (1) over a mesh element K_i, and applying the Stokes formula we obtain:

$$\int_{\partial K_i} (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_i \mathrm{d}s + \int_{K_i} \underline{f} \mathrm{d}x = \sum_{\sigma \in \mathcal{F}_i} \int_{\sigma} (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_{\sigma,i} \mathrm{d}\sigma + \int_{K_i} \underline{f} \mathrm{d}x = 0. \tag{2}$$

If we introduce the discrete unknowns p_i per mesh element K_i and define a cellwise constant function $\underline{p}_h$ that takes the values p_i on the mesh elements K_i, then the fluxes $\int_\sigma (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_{\sigma,i} \mathrm{d}\sigma$ can be approximated as functions of the discrete unknowns:

$$\int_{\sigma} (\underline{\mathbf{K}} \nabla \underline{p}) \cdot \boldsymbol{n}_{\sigma,i} \mathrm{d}\sigma \approx F_{i,\sigma}(\underline{p}_h),$$

where we have in a finite volume scheme with two point flux approximation for example:

$$F_{i,\sigma}(\underline{p}_h) = \begin{cases} T_{\sigma_i}(p_i) \quad \text{with} \quad T_{\sigma_i} = \dfrac{|\sigma| \kappa_i}{d_{i,\sigma}} \quad \text{if} \quad \sigma = \sigma_i \subset \partial\Omega, \\ T_{\sigma_{ij}}(p_i - p_j) \quad \text{if} \quad \sigma = \sigma_{ij} \not\subset \partial\Omega, \end{cases} \tag{3}$$

with

$$T_{\sigma_{i,j}} = |\sigma| \left(\frac{\kappa_i}{d_{i,\sigma}} + \frac{\kappa_j}{d_{j,\sigma}} \right)^{-1} \left(\frac{\kappa_i}{d_{i,\sigma}} \times \frac{\kappa_j}{d_{j,\sigma}} \right).$$

In (3), $F_{i,\sigma}(\underline{p}_h)$ denotes a flux through the edge σ, whereas $T_{\sigma_{i,j}}$ and T_{σ_i} are transmissivities. Therefore, (2) becomes:

$$\sum_{\sigma_i \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}}} F_{i,\sigma}(\underline{p}_h) + \sum_{\sigma \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} F_{i,\sigma}(\underline{p}_h) = |K_i| f_i$$

$$\left(\sum_{\sigma \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\sigma_{i,j} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}}} T_{\sigma_{i,j}} \right) p_i - \sum_{\sigma_{i,j} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}}} T_{\sigma_{i,j}} p_j = |K_i| f_i. \tag{4}$$

This means that the vector $\mathbf{x} = (p_i)_{1 \le i \le n}$ is the solution of the linear system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ where:

$$\forall i, j \in [\![1, n]\!], i \neq j : \quad \mathbf{b}_i := |K_i| f_i, \tag{5}$$

$$\mathbf{A}_{ii} = \sum_{\sigma_i \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\sigma_{i,j} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}}} T_{\sigma_{i,j}},$$

$$\mathbf{A}_{ij} = \begin{cases} T_{\sigma_{i,j}} & \text{if} \quad j \in V(i), \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Remark 2.1. According to (3), we notice that $T_{\sigma_{i,j}} = T_{\sigma_{j,i}}$ for arbitrary neighbors i and j. Therefore, it should be noted from the definition (6) that the matrix $\mathbf{A}$ is symmetric. In addition, it is diagonally dominant with positive diagonal entries. $\mathbf{A}$ is thus symmetric positive definite.

3 Matrix decomposition and local error reduction

An iterative Krylov solver is used to solve the system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$. At each iteration j of the iterative solver, we denote by $\mathbf{x}^{(j)}$ the approximate solution obtained, and by $\underline{p}_h^{(j)}$ the associated approximate function.

For a finite element discretization, a way to build adaptive preconditioners, based on *a posteriori* error estimators, in order to effectively decrease the energy norm of the error was introduced in [6]. We briefly recall this approach here. First, the initial domain is decomposed into two disjoint open subdomains Ω_1 and Ω_2. The former is an aggregate of mesh elements where the algebraic error estimates are significant, the latter is an aggregate of the remaining mesh elements. This error-based classification allows to first formulate a sum-splitting of the operator:

$$\mathbf{A} = \mathbf{A}_p^{(1)} + \mathbf{A}_p^{(2)}, \tag{7}$$

where $\mathbf{A}_p^{(1)}$ and $\mathbf{A}_p^{(2)}$ are the extensions of the local stiffness matrices for the subdomains Ω_1 and Ω_2 to the whole domain Ω. Then, it is possible to derive a 2×2 block partitioning of the matrix $\mathbf{A} = \left(\begin{array}{c|c} \mathbf{A}_L & \mathbf{A}_{LR} \\ \hline \mathbf{A}_{RL} & \mathbf{A}_R \end{array} \right)$, where the L-part stands for the set of vertices that belong to the closure of Ω_1 and the R-part for the vertices of the interior of Ω_2. The adaptive preconditioners have the following shape $\mathbf{M} = \left(\begin{array}{c|c} \mathbf{A}_L & \mathbf{A}_{LR} \\ \hline \mathbf{A}_{RL} & \mathbf{M}_S + \mathbf{A}_{RL} \mathbf{A}_L^{-1} \mathbf{A}_{LR} \end{array} \right)$, where $\mathbf{M}_S$ is an arbitrary preconditioner whose size is consistent with that of $\mathbf{A}_R$. When applied on a PCG solver with a specific

initial guess $\mathbf{x}^{(O)} = \begin{bmatrix} \mathbf{A}_L^{-1} \cdot (\mathbf{b}_L - \mathbf{A}_{LR} \cdot \mathbf{y}) \\ \mathbf{y} \end{bmatrix}$ where $\mathbf{y}$ is an arbitrary vector whose length is consistent with the size of $\mathbf{A}_R$, this kind of preconditioners cancels the partial residual on the unknowns of the L-part and thus ensures the decay of local high algebraic errors.

For a finite volume scheme, the original cellwise constant finite volume approximation $\underline{p}_h \in \mathbb{P}_0(\mathcal{T}_h)$ is not appropriate for energy-norm type *a posteriori* error estimation as it is only piecewise constant. For this reason, a postprocessed approximation that has more regularity is introduced $\underline{\tilde{p}}_h$ [7, 8]. This time, the *a posteriori* error analysis allows to derive estimates for the classical associated error norm:

$$||\underline{K}^{\frac{1}{2}} \nabla(\underline{p} - \underline{\tilde{p}}_h)||^2_{L^2(\Omega)} \leq \eta_{\text{disc}} + \eta_{\text{alg}},$$

where η_{disc} and η_{alg} are *a posteriori* error estimates of the error components: the former term is supposed to approximate the discretization error and the latter the algebraic error [8]. In addition, such quantities can be obtained locally on each mesh element, therefore, analogous to [6], we use algebraic error estimates at a fixed iteration i of the iterative solver to have a domain decomposition:

$$\overline{\mathbf{\Omega}}_1 \cup \overline{\mathbf{\Omega}}_2 = \overline{\mathbf{\Omega}},$$
$$\mathbf{\Omega}_1^{\text{o}} \cap \mathbf{\Omega}_2^{\text{o}} = \emptyset,$$

where $\mathbf{\Omega}_1$ is the part with the high algebraic error estimates:

$$\sum_{K \in \mathcal{T}_h(\mathbf{\Omega}_1)} (\eta^{(i)}_{\text{alg},K})^2 \gg \sum_{K \in \mathcal{T}_h(\mathbf{\Omega}_2)} (\eta^{(i)}_{\text{alg},K})^2. \tag{8}$$

Remark 3.1. From (6), we can derive formulas for two matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ local to $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ respectively by considering for all $i \neq j$ in $[\![1,\ n]\!]$ such that $K_i \subset \mathbf{\Omega}_1$ and $K_j \subset \mathbf{\Omega}_1$:

$$\mathbf{A}^{(1)}_{ii} = \sum_{\sigma_i \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_1(i)}} T_{\sigma_{i,l}}, \tag{9}$$

$$\mathbf{A}^{(1)}_{ij} = \begin{cases} T_{\sigma_{i,j}} & \text{if} \quad j \in V(i), \\ 0 & \text{otherwise}, \end{cases} \tag{10}$$

and for all $i \neq j$ in $[\![1,n]\!]$ such that $K_i \subset \mathbf{\Omega}_2$ and $K_j \subset \mathbf{\Omega}_2$:

$$\mathbf{A}^{(2)}_{ii} = \sum_{\sigma_i \in \mathcal{F}_i \cap \mathcal{F}_{\text{ext}}} T_{\sigma_i} + \sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_2(i)}} T_{\sigma_{i,l}}, \tag{11}$$

$$\mathbf{A}^{(2)}_{ij} = \begin{cases} T_{\sigma_{i,j}} & \text{if} \quad j \in V(i), \\ 0 & \text{otherwise}, \end{cases} \tag{12}$$

where $V_1(i)$ (resp. $V_2(i)$) denotes the set of indices belonging to $V(i)$ but whose associated polyhedra are included in $\mathbf{\Omega}_1$ (resp. $\mathbf{\Omega}_2$). Therefore, we can express the global matrix $\mathbf{A}$ as a sum:

$$\mathbf{A} = \mathbf{A}^{(1)} + \mathbf{A}^{(2)} + \mathbf{F}, \tag{13}$$

where for all $i \neq j$ in $[\![1,\ n]\!]$:

$$\mathbf{F}_{ii} = \begin{cases} \displaystyle\sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_2(i)}} T_{\sigma_{i,l}} & \text{if} \quad K_i \in \mathbf{\Omega}_1, \\ \displaystyle\sum_{\substack{\sigma_{i,l} \in \mathcal{F}_i \cap \mathcal{F}_{\text{int}} \\ l \in V_1(i)}} & \text{otherwise}, \end{cases} \tag{14}$$

$$\mathbf{F}_{ij} = \begin{cases} T_{\sigma_{i,j}} & \text{if} \quad K_i \in \mathbf{\Omega}_1 \quad \text{and} \quad j \in V_2(i), \\ T_{\sigma_{i,j}} & \text{if} \quad K_i \in \mathbf{\Omega}_2 \quad \text{and} \quad j \in V_1(i), \\ 0 & \text{otherwise}, \end{cases} \tag{15}$$

Formula (13) recalls the sum splitting of the matrix in (7) for the finite element discretization method. The difference is that now for the finite volume method, we have a third term $\mathbf{F}$ that is not local to only one subdomain. This is due to the fact that the finite element scheme is by definition vertex-centered whereas the finite volume scheme considered here is cell-centered. This kind of scheme provides an approximation of the solution that is piecewise constant on a primal mesh. That being said, there exist some vertex-centered finite volume schemes that introduce a so-called dual mesh, which is a conforming triangulation of the domain $\mathbf{\Omega}$, around the vertices (see, e.g. [9]). However, for the current context of the application, as the *a posteriori* error estimates implemented in *IFPEN*'s software were derived for cell-centered schemes, we focus on this latter type of schemes in the sequel.

In finite volume method, we cannot reproduce in the language of matrices an inequality that is equivalent to the main starting hypothesis in the finite element framework, *i.e.* Formula (8) of the article [6] that we have briefly explained earlier, but we have some estimations of the terms involved in that hypothesis. Therefore, instead of the initial hypothesis with the exact algebraic error, we consider a starting hypothesis that involves error estimates. It is expressed in the assumption (8).

Accordingly, since we are filtering the mesh elements with high errors in $\mathbf{\Omega}_1$, we expect an inequality that can be written in the form:

$$\left(\mathbf{x}_L - \mathbf{x}_L^{(i)}\right)^T \cdot \mathbf{A}_L \cdot \left(\mathbf{x}_L - \mathbf{x}_L^{(i)}\right) \gg \left(\mathbf{x}_R - \mathbf{x}_R^{(i)}\right)^T \cdot \mathbf{A}_R \cdot \left(\mathbf{x}_R - \mathbf{x}_R^{(i)}\right) \tag{16}$$

is satisfied where L and R are subsets that include the indices of mesh elements contained in $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ respectively. We denote by n_L and n_R the sizes of $\mathbf{A}_L$ and $\mathbf{A}_R$ respectively. Their sum is equal to the size of $\mathbf{A}$: $n_L + n_R = n$.

We recall that since $\mathbf{A}$ is SPD, $\mathbf{A}_L$ and $\mathbf{A}_R$ are SPD as well and the two terms involved in assumption (16) are a part of the global energy norm of the initial system:

$$||\mathbf{x}-\mathbf{x}^{(i)}||^2_{\mathbf{A}} = \left\langle \mathbf{A}\cdot\left(\mathbf{x}-\mathbf{x}^{(i)}\right),\ \mathbf{x}-\mathbf{x}^{(i)}\right\rangle$$
$$= \underbrace{\left\langle \mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(i)},\ \mathbf{x}-\mathbf{x}^{(i)}\right\rangle_L}_{:=L-\text{term}} + \underbrace{\left\langle \mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(i)},\ \mathbf{x}-\mathbf{x}^{(i)}\right\rangle_R}_{:=R-\text{term}},$$

where

$$\left\langle \mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(i)},\ \mathbf{x}-\mathbf{x}^{(i)}\right\rangle_L = ||(\mathbf{x}-\mathbf{x}^{(i)})_L||^2_{\mathbf{A}_L} + (\mathbf{x}-\mathbf{x}^{(i)})^T_L\cdot\mathbf{A}_{LR}\cdot(\mathbf{x}-\mathbf{x}^{(i)})_R, \quad (17)$$

$$\left\langle \mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(i)},\ \mathbf{x}-\mathbf{x}^{(i)}\right\rangle_R = ||(\mathbf{x}-\mathbf{x}^{(i)})_R||^2_{\mathbf{A}_R} + (\mathbf{x}-\mathbf{x}^{(i)})^T_R\cdot\mathbf{A}_{RL}\cdot(\mathbf{x}-\mathbf{x}^{(i)})_L.$$

This is equivalent to the formulation obtained in the case of finite element methods, described in Section 3.3 of [6] (Formulas (19) and (20)). Due to the symmetry, the coupling terms are identical:

$$\left(\mathbf{x}-\mathbf{x}^{(i)}\right)^T_R\cdot\mathbf{A}_{RL}\cdot\left(\mathbf{x}-\mathbf{x}^{(i)}\right)_L = \left(\mathbf{x}-\mathbf{x}^{(i)}\right)^T_L\cdot\mathbf{A}_{LR}\cdot\left(\mathbf{x}-\mathbf{x}^{(i)}\right)_R.$$

Assumption (16) implies that the $\mathbf{A}_L$-inner product of the error is dominant, and so will be the L-term according to (17). Therefore, reducing them may efficiently reduce the energy norm of the error. As the vectors $(\mathbf{x}-\mathbf{x}^{(i)})_L$ and $\mathbf{A}_L\cdot(\mathbf{x}-\mathbf{x}^{(i)})_L$ cannot be computed, we favor the alternative of reducing the partial residual $(\mathbf{b}-\mathbf{A}\cdot\mathbf{x}^{(i)})_L$ to decrease the L-term. A straightforward manner to bring that partial residual down to zero is by using a Schur complement reduction that we detail in the next section.

4 Adaptive linear solver

In this section, we explain an approach for reducing the dominant part of the algebraic error. This procedure is based on the exact decomposition of the L-block and a Schur complement of the R-block. In the field of linear algebra and the theory of matrices, the literature is rich on these two techniques. For a detailed description of these concepts, we refer the reader to [10, 11] and the references therein. Another popular use for those techniques is the construction of preconditioners. Approximate or inexact factorization is often used as a preconditioner for iterative solvers such as PCG [1]. In domain decomposition methods as well, many research works were made to devise preconditioners for the global matrix $\mathbf{A}$ from techniques that approximately solve the reduced Schur complement system [12, 13]. We also specify that, when a PCG solver is used, the solve procedure described below, which is based on a reduced Schur complement system, is equivalent to iteratively solving the global system $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ with the initial guess and the preconditioner introduced in Section 3. Furthermore, we check if that adaptive solve procedure can be valid for other solvers than PCG.

4.1 Schur complement procedure

Let us define the following matrices:

$$\mathbf{S} = \mathbf{A}_R - \mathbf{A}_{RL}\mathbf{A}_L^{-1}\mathbf{A}_{LR}, \quad \mathbf{A}_L = \mathbf{D}_1\mathbf{D}_2, \quad (18)$$

$$\mathbf{N}_1 = \mathbf{A}_{RL}\mathbf{D}_2^{-1}, \quad \mathbf{N}_2 = \mathbf{D}_1^{-1}\mathbf{A}_{LR}, \quad (19)$$

$$\mathbf{E}_1 = \left(\begin{array}{c|c}\mathbf{D}_1 & 0\\ \hline \mathbf{N}_1 & \mathbf{I}\end{array}\right), \quad \mathbf{E}_2 = \left(\begin{array}{c|c}\mathbf{D}_2 & \mathbf{N}_2\\ \hline \mathbf{0} & \mathbf{S}\end{array}\right), \quad (20)$$

where $\mathbf{I}$ is the identity matrix of the same size as $\mathbf{A}_R$ and $\mathbf{S}$ is referred to as the Schur complement of the R block $\mathbf{A}_R$. Since $\mathbf{A}_L$ is nonsingular (because it is a principal submatrix of $\mathbf{A}$ which is SPD), the Schur complement is well defined. The matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ are exact factors of $\mathbf{A}_L$ where $\mathbf{D}_1$ is the lower triangular factor, and $\mathbf{D}_2$ is the upper triangular one.

With the definitions of (19) and (20), we have the equality:

$$\mathbf{A} = \mathbf{E}_1\mathbf{E}_2, \quad (21)$$

and we can write

$$\mathbf{A}\cdot\mathbf{x}=\mathbf{b} \Longleftrightarrow \begin{cases}\mathbf{E}_1\cdot\mathbf{z}=\mathbf{b}, & \text{(P.1)}\\ \mathbf{E}_2\cdot\mathbf{x}=\mathbf{z}. & \text{(P.2)}\end{cases}$$

Splitting the system $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ into two systems related to L and R domains with the Schur complement is equivalent to solving the system (P′) in the following two stages:
① Solving (P.1): The matrix $\mathbf{E}_1$ is lower triangular, and therefore (P.1) represents a forward substitution:

$$\text{(P.1)} \Longleftrightarrow \begin{cases}\mathbf{D}_1\cdot\mathbf{z}_L = \mathbf{b}_L\\ \mathbf{N}_1\cdot\mathbf{z}_L+\mathbf{z}_R=\mathbf{b}_R\end{cases} \Longleftrightarrow \begin{cases}\mathbf{z}_L=\mathbf{D}_1^{-1}\cdot\mathbf{b}_L,\\ \mathbf{z}_R=\mathbf{b}_R-\mathbf{N}_1\cdot\mathbf{z}_L.\end{cases} \quad (22)$$

② Solving (P.2): We first solve a reduced system with the Schur complement $\mathbf{S}$. Then, we perform a backward substitution with an upper triangular matrix $\mathbf{D}_2$.

$$\text{(P.2)} \Longleftrightarrow \begin{cases}\mathbf{S}\cdot\mathbf{x}_R=\mathbf{z}_R\\ \mathbf{D}_2\cdot\mathbf{x}_L+\mathbf{N}_2\cdot\mathbf{x}_R=\mathbf{z}_L\end{cases}$$
$$\Longleftrightarrow \begin{cases}\mathbf{S}\cdot\mathbf{x}_R=\mathbf{z}_R, & \text{(23a)}\\ \mathbf{x}_L=\mathbf{D}_2^{-1}\cdot(\mathbf{z}_L-\mathbf{N}_2\cdot\mathbf{x}_R). & \text{(23b)}\end{cases}$$

The size of system (23a) is smaller than the size of system $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ because its size is equal to the size of R domain, where no significant algebraic errors were observed. In order to avoid forming the (possibly dense) matrix $\mathbf{S}$, an iterative Krylov solver can be used for solving (23a) as well.

Let $\mathbf{x}_R^{(i)}$ be the approximate solution obtained after i iterations. The associated solution $\mathbf{x}_L^{(i)}$ may be calculated by (23b) as:

$$\mathbf{x}_L^{(i)} = \mathbf{D}_2^{-1} \cdot \left(\mathbf{z}_L - \mathbf{N}_2 \cdot \mathbf{x}_R^{(i)}\right). \tag{24}$$

The adaptive solving procedure can be summarized in two major stages presented above. Algorithm 1a for the setting up of the method (splitting and permuting) and Algorithm 1b for the computation phase.

Remark 4.1. In the first step of Algorithm 1a, the algebraic error estimation can be done in different ways, we cite for example ([8], Sect. 4) where the authors present a simple and practical way to estimate the algebraic error.

4.2 Error reduction properties of the adaptive procedure

We recall in the following lemma some formulas for the residual and the energy norm of the error that hold with the Schur complement method.

Algorithm 1a Error-based domain decomposition

Inputs: Ω.
1: Evaluate the local algebraic error over the mesh elements.
↪ $\eta_l^{(\text{alg})}$.
2: Mark the elements K_l with largest algebraic errors $\eta_l^{(\text{alg})}$.
↪ Ω_1-subdomain.
3: Extract the node indices associated to elements of Ω_1.
↪ L-subset.
4: Find the complementary of L in the set of node indices.
5: Permute the system to obtain an L/R block splitting.
Outputs: Ω_1, L, R.

Algorithm 1b Schur complement procedure

Inputs: L, R, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{M}_S$.
1: Perform an exact factorization on the L-block.
2: Solve (P.1) by simple substitution.
3: Solve the Schur complement system (23a) via an iterative solver with preconditioner $\mathbf{M}_S$.
↪ $\mathbf{x}_R$ solution.
4: Inject the obtained vector in (23b) and proceed by backward and forward substitution.
↪ Updated $\mathbf{x}_L$ solution.
Outputs: $\mathbf{x}_L$, $\mathbf{x}_R$.

Lemma 4.1. *By applying the Schur complement procedure described in Section 4.1, we obtain at each iteration i of an iterative solver for any $\mathbf{x}_R^{(i)}$ approximating $\mathbf{x}_R$ and associated $\mathbf{x}_L^{(i)}$ given by (24):*

$$\begin{cases} \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{z}_R - \mathbf{S} \cdot \mathbf{x}_R^{(i)} \end{bmatrix}, & (25) \\ ||\mathbf{x} - \mathbf{x}^{(i)}||_{\mathbf{A}}^2 = \left\langle \left(\mathbf{x} - \mathbf{x}^{(i)}\right)_{/R},\ \mathbf{z}_R - \mathbf{S} \cdot \mathbf{x}_R^{(i)} \right\rangle. & (26) \end{cases}$$

Proof.

$$\begin{aligned} \left(\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}\right)_{/L} &= \mathbf{A}_L \cdot \left(\mathbf{x} - \mathbf{x}^{(i)}\right)_{/L} + \mathbf{A}_{LR} \cdot \left(\mathbf{x} - \mathbf{x}^{(i)}\right)_{/R} \\ &= \mathbf{b}_L - \mathbf{A}_L \cdot \mathbf{x}_L^{(i)} - \mathbf{A}_{LR} \cdot \mathbf{x}_R^{(i)} \overset{(25)}{=} \mathbf{b}_L \\ &\quad - \left(\mathbf{b}_L - \mathbf{D}_1\mathbf{N}_2 \cdot \mathbf{x}_R^{(i)}\right) - \mathbf{A}_{LR} \cdot \mathbf{x}_R^{(i)} \overset{(19)}{=} \mathbf{0}. \end{aligned}$$

And similarly,

$$\begin{aligned} \left(\mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(i)}\right)_{/R} &= \mathbf{A}_{RL} \cdot \left(\mathbf{x} - \mathbf{x}.^{(i)}\right)_{/L} + \mathbf{A}_R \cdot \left(\mathbf{x} - \mathbf{x}^{(i)}\right)_{/R} \\ &= \mathbf{b}_R - \mathbf{A}_{RL} \cdot \mathbf{x}_L^{(i)} - \mathbf{A}_R \cdot \mathbf{x}_R^{(i)} \overset{(19)}{=} \mathbf{b}_R - \mathbf{N}_1\mathbf{D}_2 \cdot \mathbf{x}_L^{(i)} - \mathbf{A}_R \cdot \mathbf{x}_R^{(i)} \\ &\overset{(25)}{=} \mathbf{b}_R + \mathbf{N}_1 \cdot \left(\mathbf{N}_2 \cdot \mathbf{x}_R^{(i)} - \mathbf{z}_L\right) - \mathbf{A}_R \cdot \mathbf{x}_R^{(i)} \\ &\overset{(23)\,\&\,(18)}{=} \mathbf{z}_R - \mathbf{S} \cdot \mathbf{x}_R^{(i)}. \end{aligned}$$

By looking at the block expression of the residual vector in (25), it can be stated that the L-part of that vector is eliminated and there remains only the R-part. Consequently, in Formula (26) we got rid of the L-term which was dominant and the energy norm now depends only on the scalar product between $\left(\mathbf{x} - \mathbf{x}^{(i)}\right)_{/R}$ the error vector projected on R, and the residual of the solution of (23a).

We would add that in practice, the only components we actually need to compute, for the whole solving process, are the factors $\mathbf{D}_1$ and $\mathbf{D}_2$. The benefit of solving (23a) by some Krylov method is that it only requires matrix-vector products $\mathbf{S}$ times a vector $\mathbf{w}$ which can be performed without computing the entries of the Schur complement matrix $\mathbf{S}$. This way, the solving process combines a direct solver in the subdomain L with an iterative solver for the subdomain R. Thus, it can be seen as an hybrid direct/iterative solver. As for the choice of the preconditioning to be used during the iterative solve, we refer to [14–16] for a range of preconditioners for the Schur complement. Sometimes, a preconditioner $\mathbf{M}_R$ of the submatrix $\mathbf{A}_R$ can be used to precondition the Schur complement $\mathbf{S}$.

So far, we have shown that decreasing the algebraic error by reducing the residual on the L-part to zero is achievable for other solvers than PCG. This represents an extension of the adaptive procedure: In the finite element framework, only PCG solver could be used as stated in Section 3, whereas for a finite volume scheme, any iterative solver is valid.

5 Numerical results

In this section, we present numerical results of tests where we apply the adaptive solve procedure in a reservoir simulation for heterogeneous porous media. The model problem is a single phase flow model, for which we consider two types of problems: steady (as introduced in Sect. 2) and unsteady (see Sect. 5.2.1). We first validate the procedure in the simpler steady case with an initial prototype. Once the method is validated, we assess its performance on the real simulator with the unsteady case. The simulations are run on *IFPEN*'s prototype for reservoir simulation.

In the sequel, we solve the linear systems stemming from the PDEs by the adaptive Schur procedure and compare it to the classical solve procedure. Both procedures employ the same Krylov solver and the same incomplete factorization. The classical solve is performed with an incomplete factorization preconditioner for the global matrix $\mathbf{A}$, whereas the adaptive Schur procedure uses an incomplete factorization of the submatrix $\mathbf{A}_R$ to precondition the reduced Schur complement system. We also consider a stopping threshold value of 10^{-6} for the euclidean norm of the residual.

5.1 Steady problem: heterogeneous media and uniform mesh refinement

For the first test case, we deal with a steady problem. As it generates one single system, we extract data (matrix, right hand side vector and error estimates) from the simulator and solve the linear system with a prototype of the adaptive approach that uses the no-fill Incomplete Cholesky factorization and a PCG solver.

In this test, we use the same configuration described in ([17], Sect. 6.3). We consider a heterogeneous porous media with domain $(0, 1200) \times (0, 2200)$ partitioned by a grid of 60×220 rectangular cells. The permeability tensor corresponds to that of the layer 85 of the tenth SPE comparative solution project model field [18]. Figure 1 shows the permeability field (on top) and the pressure field (on bottom). The source term is $f = 0$. We have tested four values (0.10, 0.15, 0.25 and 0.33) for the size percentage $\delta := \frac{n_L}{n}$ (see Sect. 3). The evolution of the energy norm is displayed in the graph of Figure 2.

We notice from Figure 2 that by initially taking a subset L which is ten times smaller than the size of the global system, we get a decrease of almost 30% in the number of iterations for the adaptive solve with respect to the standard one. The convergence is still accelerated by a wider coverage of the high error areas. This is reflected in the decrease of the number of iterations when we progressively increase the parameter δ, until we obtain a speedup of more than 50% with respect to the standard solve for $\delta = 0.33$.

5.2 Unsteady problem: heterogeneous media and uniform mesh refinement

For the second test case, we handle an unsteady problem of a single phase flow model.

5.2.1 The model

Often used in reservoir simulation, this unsteady model describes the flow of a single fluid through a porous medium $\mathbf{\Omega} \subset \mathbb{R}^d$, $d \in \{2, 3\}$, over a certain time interval. On the one hand, we consider the characteristics of the fluid that follow. We denote by $\underline{p}$ the pressure of the fluid, by ρ its mass density, by μ its viscosity, by c_f its compressibility and by $\mathbf{v}$ the fluid velocity. On the other hand, the physical characteristics of the porous medium are its porosity ϕ, and its absolute permeability tensor $\underline{\mathbf{K}}$. This latter measures the ability of the porous medium to transmit fluid in each direction. We also denote by c_R the rock compressibility and by ρ^0 the fluid density at a reference pressure $\underline{p}^0$.

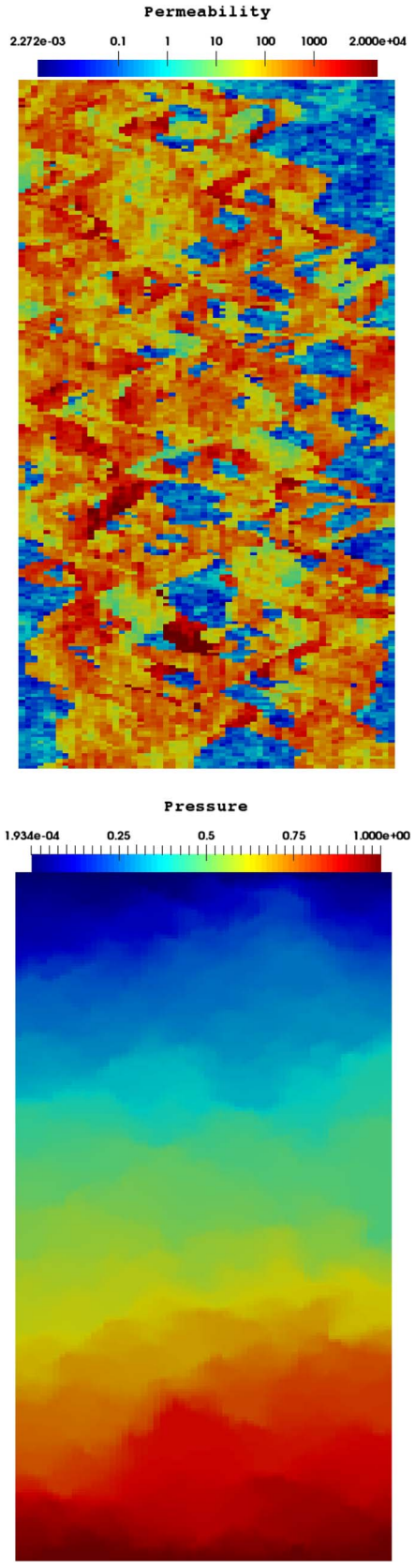


Fig. 1. SPE10 permeability (*top*) and pressure field (*bottom*). Section 5.1.

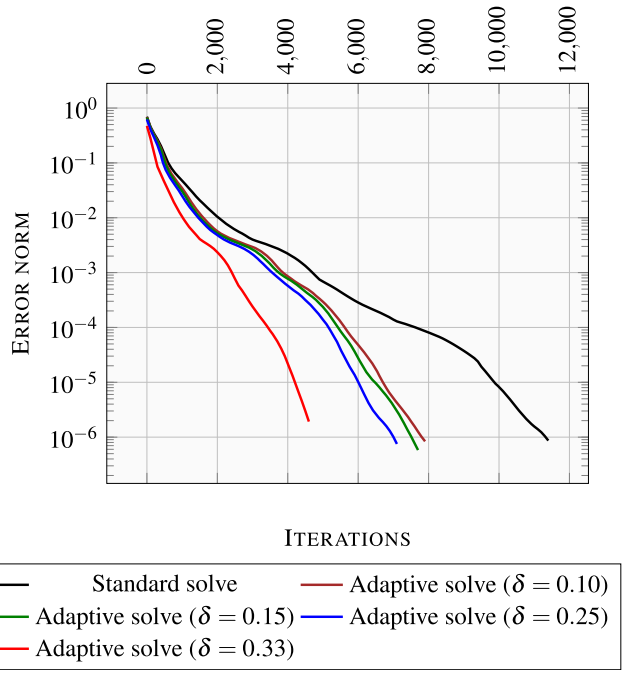


Fig. 2. Evolution of the energy norm of the error with the standard and adaptive solve procedures (steady case).

It is assumed that the mass diffusion and mass dispersion fluxes are negligible and that no fluid mass can cross the fluid–solid interface. Then, the conservation of mass is expressed by the following equation:

$$\frac{\partial(\phi\rho)}{\partial t} = -\nabla\cdot(\rho\mathbf{v}) + q, \tag{27}$$

where q is the source or sink term that is square integrable in time and space.

In addition, Darcy's law gives the expression of the fluid velocity:

$$\mathbf{v} = -\frac{1}{\mu}\underline{\mathbf{K}}(\nabla\underline{p} - \rho g\nabla z), \tag{28}$$

where g is the magnitude of the gravitational acceleration and z is the depth.

An equation of state gives the relationship between the fluid compressibility and the partial derivative of the density with respect to the pressure evaluated at a fixed temperature T:

$$c_f = \frac{1}{\rho}\frac{\partial\rho}{\partial\underline{p}}\bigg|_T. \tag{29}$$

Similarly, the rock compressibility is defined by:

$$c_R = \frac{1}{\phi}\frac{\mathrm{d}\phi}{\mathrm{d}\underline{p}}. \tag{30}$$

The time differentiation of $\phi\rho$ in (27) yields:

$$\left(\phi\frac{\partial\rho}{\partial\underline{p}} + \rho\frac{\mathrm{d}\phi}{\mathrm{d}\underline{p}}\right)\frac{\partial\underline{p}}{\partial t} = -\nabla\cdot(\rho\mathbf{v}) + q.$$

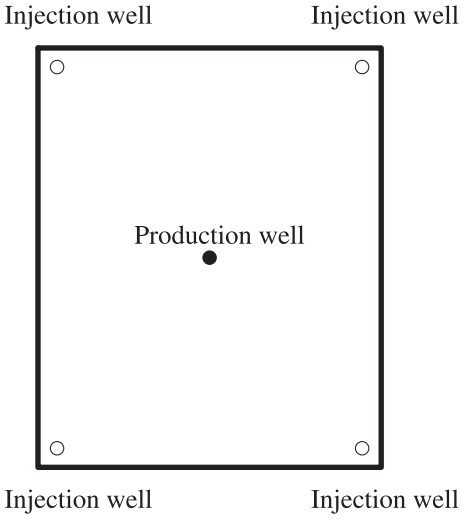


Fig. 3. Configuration for the numerical test case of Section 5.2.

By injecting the compressibility formulae (29), (30) and the momentum conservation's law (28), in the mass conservation's law (27), we obtain:

$$\rho(c_f + c_R)\phi\frac{\partial\underline{p}}{\partial t} = \nabla\cdot\left(\frac{\rho}{\mu}\underline{\mathbf{K}}(\nabla\underline{p} - \rho g\nabla z)\right) + q. \tag{31}$$

We consider that the medium contains a single fluid (oil or gas) that is slightly compressible. Thus, the fluid compressibility stays constant when the pressure varies within a certain range of values. In this case, integrating (29) yields:

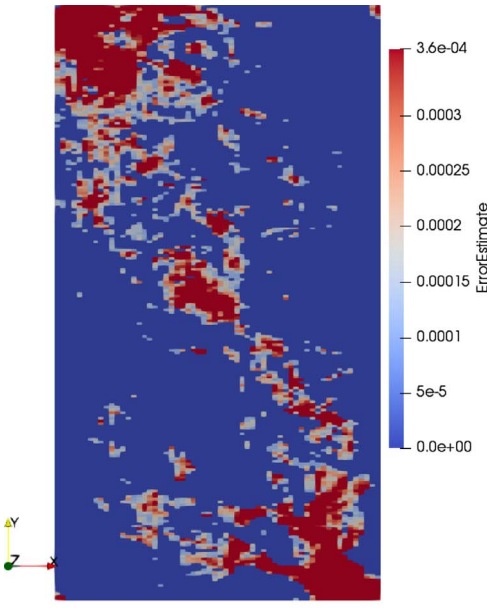
$$\rho = \rho^0 e^{c_f(\underline{p}-\underline{p}^0)}. \tag{32}$$

Hence, with (32), the governing PDE (31) is a parabolic equation for the main unknown which is the pressure $\underline{p}$. For proofs of the existence, uniqueness and regularity of a solution to this system, and for discretization and linearization approaches, we refer to [19] and references therein.

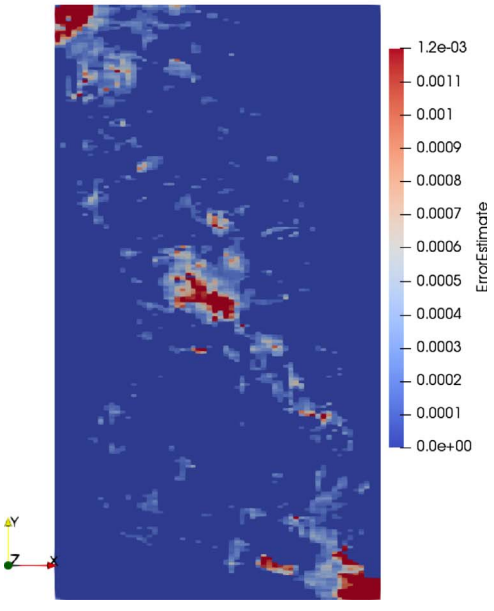
5.2.2 Simulation results

As far as the computing framework is concerned, we have implemented GMRES solver and the adaptive solve procedure on MCGSolver [20, 21]. For exact and inexact LU factorizations, we used the library Eigen [22]. The transfer of error estimates between the linear solver (MCGSolver) and the simulator is operated by the ALIEN interface which provides a modern, uniform and generic API for a wide range of linear solver libraries including Petsc [23], IFPSolver [24] and MCGSolver.

Note that, as we consider a horizontal 2D case, gravitational effects are not taken into account in the numerical tests. A simulation over a 24-h period was conducted in this study. We consider a 2D cartesian grid (60×220). At each time t_i, an initial time step $\Delta t_i^{(0)}$ is set, a linear system is generated and solved with GMRES solver. Note that usually the non-linear solver does not converge when the linear system's solution did not converge. Note also that when the non-linear solver does not converge, the time step is halved

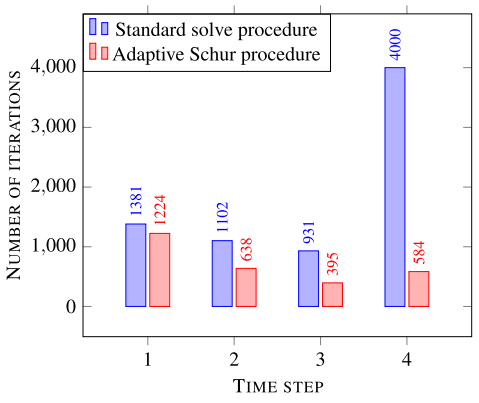


(a) A posteriori error estimates at the beginning of the simulation

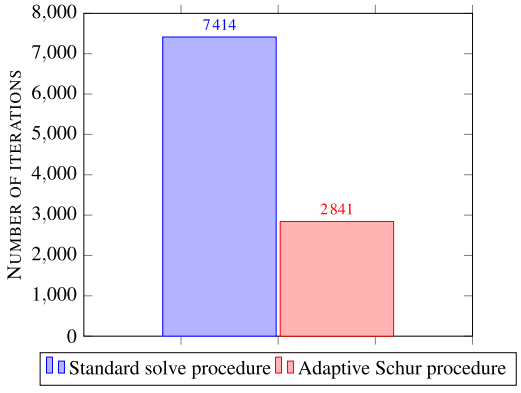


(b) A posteriori error estimates at the end of the simulation

Fig. 4. Distribution of *a posteriori* error estimates during the single phase flow simulation.



(a) Per time step



(b) Total

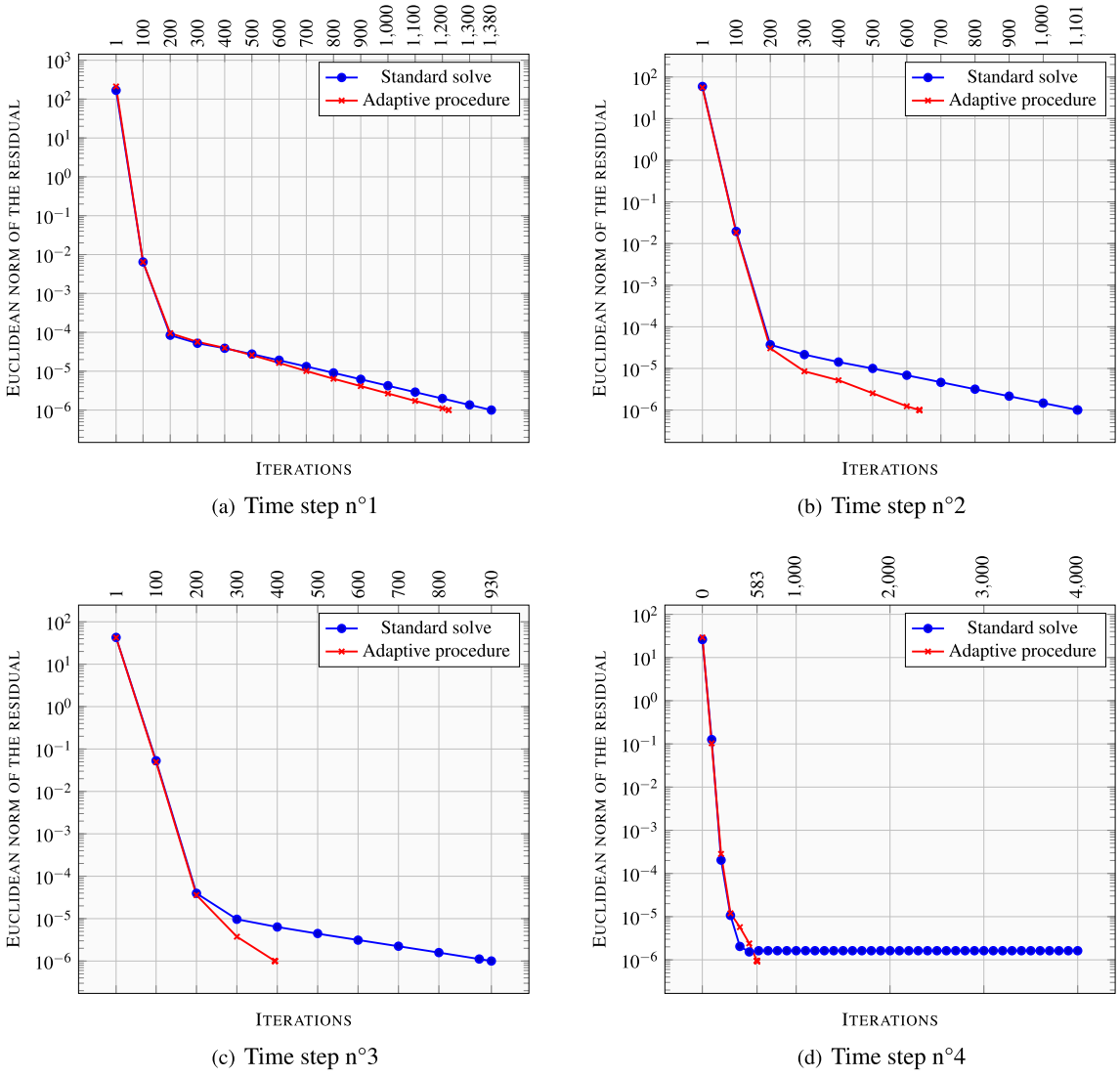
Fig. 5. Number of iterations during the first four time steps after initialization.

$\Delta t_i^{(j+1)} := \frac{\Delta t_i^{(j)}}{2}$ until reaching convergence with $j = j_c$. In this case, $\left(\Delta t_i^{(j)}\right)_{j<j_c}$ (resp. $\Delta t_i^{(j_c)}$) are called failed (resp. accepted) time steps at the time t_i. The next time is set from the accepted time step $t_{i+1} := t_i + \Delta t_i^{(j_c)}$.

The size of the system matrix $\mathbf{A}$ is 12 997 × 12 997, whereas the size of the submatrix $\mathbf{A}_L$ is 417 × 417. For the sake of comparison, the resolutions in the simulation are carried out according to two procedures. The first is the standard solution process using an ILU(0) factorization of the matrix $\mathbf{A}$ as a preconditioner. The second one is the adaptive procedure described in Section 4.1.

The locations of the wells at the reservoir are indicated on Figure 3. The wells are arranged in a five-spot pattern, with the production well positioned at the center. The distribution of *a posteriori* error estimates during the beginning and the end of the simulation is plotted on Figure 4.

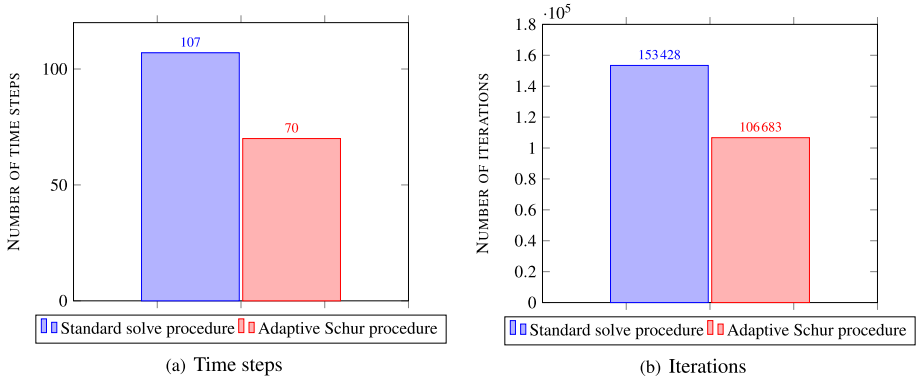
We are interested in the number of iterations needed at every time step of the simulation for the standard solve procedure and the adaptive Schur procedure respectively. Whenever that number reaches 4000, which is the maximum number of iterations allowed, this indicates a failed time step. One can observe that the first failed time step occurs at the sixth time t_6 for the standard procedure. Therefore, the subsequent times $\left(t_j\right)_{j>6}$ differ between the standard and adaptive procedure. Yet, we still can compare the two procedures during the first time steps that are identical and common for them both. The number of iterations for the times in question is displayed in Figure 5. The evolution of the norm of the residual over iterations with both procedures is plotted for each time step in Figure 6. With respect to the standard solve procedure, we observe a speedup with the adaptive Schur procedure in the first common time steps.



(a) Time step n°1 (b) Time step n°2

(c) Time step n°3 (d) Time step n°4

Fig. 6. Convergence of the solve procedures during the first four time steps after initialization.



(a) Time steps (b) Iterations

Fig. 7. The total number of time steps and iterations needed for the whole simulation with the standard and adaptive procedures. The maximum number of iterations allowed per solve is set to 4000.

Table 1. The solve times (in seconds) for the first four time steps of the simulation with the standard and adaptive procedures.

	Standard solve	Adaptive Schur
Time step no. 1	7.15	14.26
Time step no. 2	5.59	6.91
Time step no. 3	4.68	3.84
Time step no. 4	21.06	5.84

Moreover, this trend is confirmed on a larger scale for a whole simulation. The total numbers of time steps and GMRES iterations are reported in Figure 7. The charts of this latter indeed indicate that more time steps and iterations are needed for the simulation when the standard solve procedure is employed compared to the adaptive one.

The solve times are collected in Table 1. We notice that the standard solve takes less time than the adaptive procedure in the first two time steps, and more time on the third and fourth time steps.

6 Conclusion

In this article, we have adapted the *a posteriori* error estimates hypothesis for the finite volume discretization of the model problem. Then we presented the adaptive Schur procedure to exploit this hypothesis in order to effectively reduce the algebraic error and accelerate the convergence. Lastly, we showed the numerical results of the method applied to the framework of reservoir simulation. The results of the initial tests are encouraging. The comparison with the standard solve illustrates the performance of the adaptive procedure and in particular reveals that a significant speedup in terms of the number of time steps and iterations can be achieved. Yet, the results for the time gain of this method are not as conclusive as for the number of iterations and time steps because the implementation of the approach was not optimized, making the computations with the Schur complement costly in time. There is certainly room for improvement in this regard. For future perspectives, we could rely on hierarchical matrices [25] and the relevant techniques that are efficient for data-sparse representations of certain densely populated matrices such as the Schur complement in the elliptic models to reduce the costs in floating point operations (the complexity) and time. In addition, further test cases are envisaged, such as two-phase or multiphase flow models. Furthermore, extended research is necessary to address the issue of adaptive solution procedures based on *a posteriori* error estimators for complex models and in three dimensions. On the basis of previous studies, we expect that the feasibility of this method might depend on some essential factors:

- The matrix in the linearized system has to be symmetric.
- The evaluation of the algebraic error estimates at each step should not be too costly.
- The regions with high errors should not be too scattered, as this would yield a larger subdomain $\mathbf{\Omega}_1$ and a denser Schur complement $\mathbf{S}$, making the application of this matrix to vectors more costly. And of course, in three dimensions, the spread of the error is allowed and possible in an extra direction with respect to a two-dimension configuration. Thus, dealing with such cases is certainly more challenging for the proposed method.

References

1 Saad Y. (2000) *Iterative methods for sparse linear systems*, 2nd edn, SIAM.
2 D'ambra P., Filippone S., Vassilevski P.S. (2018) Bootcmatch: A software package for bootstrap amg based on graph weighted matching, *ACM Trans. Math. Softw.* **44**, 39:1–39:25. doi: 10.1145/3190647.
3 Xu W., Zikatanov L.T. (2018) Adaptive aggregation on graphs, *J. Comput. Appl. Math.* **340**, 718–730. doi: 10.1016/j.cam.2017.10.032.
4 Hu X., Lin J., Zikatanov L.T. (2019) An adaptive multigrid method based on path cover, *SIAM J. Sci. Comput.* **410**, 5, S220–S241. doi: 10.1137/18M1194493.
5 Eymard R., Gallouët T., Herbin R. (2000) Finite volume methods, *Handbook Numer. Anal.* **7**, 713–1018.
6 Anciaux-Sedrakian A., Grigori L., Jorti Z., Papež J., Yousef S. (2020) Adaptive solution of linear systems of equations based on *a posteriori* error estimators, *Numer. Algorithms* **84**, 331–364. doi: 10.1007/s11075-019-00757-z.
7 Vohralík M. (2008) Residual flux-based *a posteriori* error estimates for finite volume and related locally conservative methods, *Numer. Math.* **1110**, 1, 121–158.
8 Ern A., Vohralík M. (2013) Adaptive inexact Newton methods with *a posteriori* stopping criteria for nonlinear diffusion PDEs, *SIAM J. Sci. Comput.* **350**, 4, A1761–A1791. doi: 10.1137/120896918.
9 Erath C., Praetorius D. (2016) Adaptive vertex-centered finite volume methods with convergence rates, *SIAM J. Numer. Anal.* **540**, 4, 2228–2255.
10 Zhang F. (2006) *The Schur complement and its applications*, Vol. **4**, Springer Science & Business Media.
11 Trefethen L.N., Bau D. (1997) *Numerical linear algebra*, Vol. **50**, SIAM.
12 Dolean V., Jolivet P., Nataf F. (2015) *An introduction to domain decomposition methods: Algorithms, theory, and parallel implementation*, SIAM.
13 Li R., Xi Y., Saad Y. (2016) Schur complement-based domain decomposition preconditioners with low-rank corrections, *Numer. Linear Algebr. Appl.* **230**, 4, 706–729.
14 Saad Y., Sosonkina M. (1999) Distributed schur complement techniques for general sparse linear systems, *SIAM J. Sci. Comput.* **210**, 4, 1337–1356. doi: 10.1137/S1064827597328996.
15 Chan T.F., Mathew T.P. (1994) Domain decomposition algorithms, in: *Acta Numerica 1994*, Cambridge University Press, pp. 61–143.
16 Li Z., Saad Y., Sosonkina M. (2003) parms: a parallel version of the algebraic recursive multilevel solver, *Numer. Linear Algebr. Appl.* **100**, 5–6, 485–509. doi: 10.1002/nla.325.
17 Mallik G., Vohralík M., Yousef S. (2020) Goal-oriented *a posteriori* error estimation for conforming and nonconforming approximations with inexact solvers, *J. Comput. Appl. Math.* **366**, 112367. doi: 10.1016/j.cam.2019.112367.

18 Christie M.A., Blunt M.J. (2001) Tenth SPE comparative solution project : A comparison of upscaling techniques, *SPE Reserv. Evalu. Eng.* **40**, 4, 308–317. doi: 10.2118/66599-MS.

19 Chen Z., Huan G., Ma Y. (2006) *Computational methods for multiphase flows in porous media (Computational Science and Engineering 2)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

20 Anciaux-Sedrakian A., Eaton J., Gratien J.-M., Guignon T., Havé P., Preux C., Ricois O. (2015) Will GPGPUs be finally a credible solution for industrial reservoir simulators? in: *SPE Reservoir Simulation Symposium*, Vol. **1**, Society of Petroleum Engineers. doi: 10.2118/173223-MS.

21 Anciaux-Sedrakian A., Gottschling P., Gratien J.-M., Guignon T. (2014) Survey on efficient linear solvers for porous media flow models on recent hardware architectures, *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles* **69**, 4, 753–766. doi: 10.2516/ogst/2013184.

22 Guennebaud G., Jacob B., *et al.* (2010) *Eigen v3*. http://eigen.tuxfamily.org.

23 Balay S., Abhyankar S., Adams M., Brown J., Brune P., Buschelman K., Dalcin L.D., Eijkhout V., Gropp W., Kaushik D., Knepley M., May D., McInnes L., Curfman L., Munson T., Rupp K., Sanan P., Smith B., Zampini S., Zhang H., Zhang H. (2017) Petsc users manual revision 3.8, *Technical Report*, Argonne National Lab. (ANL), Argonne, IL (United States).

24 Gratien J.-M., Guignon T., Magras J.-F., Quandalle P., Ricois O. (2006) How to improve the scalability of an industrial parallel reservoir simulator, in: *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 114–121.

25 Hackbusch W. (1999) A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part i: Introduction to $\mathcal{H}$-matrices, *Computing* **620**, 2, 89–108.