Journal of Computational Physics (2018)

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Accelerating the solution of linear systems with POD-based deflation methods*

Gabriela B. Diaz Cortes[a,*], Cornelis Vuik[a,1], Jan Dirk Jansen[b]

[a]*Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Van Mourik Broekmanweg 6, 2628 XE Delft, the Netherlands.*
[b]*Delft University of Technology, Faculty of Civil Engineering and Geosciences, Stevinweg 1, 2628 CN Delft, the Netherlands.*

ARTICLE INFO

Article history:

ABSTRACT

We explore and develop a POD-based deflation methodology for the solution of ill-conditioned linear systems appearing in two-phase flow simulations. We accelerated the convergence of a Preconditioned Conjugate Gradient (PCG) achieving speed-ups of factors two to five. The computational cost of the proposed method depends on the number of deflation vectors, p, as $1 + \frac{p}{10}$. The POD-based deflation method was tested for a particular problem and linear solver; nevertheless, it can be applied to various transient problems, and multiple solvers, e.g., Krylov subspace and Multigrid methods.

© 2018 Elsevier Inc. All rights reserved.

1. Introduction

Solution of systems of linear equations are required when simulating flow through porous media. Solving the pressure equation is the most time-consuming part, especially for large and ill-conditioned systems. Furthermore, if we have a time-varying problem, it is required to compute a large number of simulations, which makes the solution of this problem expensive. Some techniques have been developed to improve the linear solver speed.

Among others, Reduced Order Models (ROM) are used to capture relevant information of a high-dimensional system and to project it into a lower-dimension space [1, 2, 3, 4, 5], which is easier to solve. With these methods, essential system information can be obtained by computing a small set of basis functions from a collection of system solutions (also known as 'snapshots'). Proper Orthogonal Decomposition (POD) is an ROM method that has recently been used to accelerate the solution of the linear pressure equation resulting from reservoir simulation [6, 7, 8, 9, 10], among other applications.

For the computation of the POD basis, two main approaches are used. In the first one, a training simulation is run and the solutions are stored as snapshots, which are collected to obtain a POD basis. This methodology is especially suited to solve problems with small changes in the input variables, e.g. the same well configurations but different flow

*This is an example for title footnote coding.
*Corresponding author: Tel.: +31 (0) 152787290;
e-mail: g.b.diazcortes@tudelft.nl (Gabriela B. Diaz Cortes)

rates or bottom hole pressures (*bhp*) [9, 6, 8]. The basis can also be computed on-the-fly, using, e.g., the solution of the latest time steps [7, 6, 11]. With this approach, the basis has to be adapted during the simulation.

Once the basis is obtained, various POD-based strategies can be used to solve the system. In the future, we will refer to the fist approach as *training phase* approach, and the second as *moving window* approach. For the solution of a large-scale system, Markovinovic et al. [7] proposed using POD techniques to compute a good initial guess that accelerates the iterative method. Solving the problem in the small-scale domain and projecting it back to the large-scale system was also approached by Astrid et al. [6]. Another approach was developed by Pasetto et al. [2], who suggested constructing a preconditioner based on the POD basis vectors. The use of the POD basis within a deflation operator was introduced by Diaz Cortes et al. [11].

For many applications, Krylov subspace iterative methods are used [12, 13][1]. The speed of convergence of these methods depends on the condition number and the right-hand side (*rhs*) of the system. If the condition number is very large, generally, preconditioning techniques are needed to transform the original system into a better conditioned one. If the system is Symmetric Positive (Semi) Definite (SP(S)D), a commonly used Krylov-subspace method is the Conjugate Gradient (CG) [14, 15, 16, 17, 5]. For CG, the Incomplete Cholesky (IC) factorization is a popular preconditioning choice [14, 18].

In recent years, deflation techniques have been developed to accelerate the convergence of Krylov subspace methods [15, 16, 19, 20, 17]. For this technique to be effective, a deflation subspace needs to be found. This subspace is such that the smallest eigenvalues of the system are no longer hampering the convergence of the iterative method.

In this work, we introduce the capture of information via POD methods with a *training phase* and a *moving window* approach. The acquired information is used for the construction of the above-mentioned deflation subspace. We explore the applicability of this methodology for the simulation of two-phase flow in large-scale, highly-heterogeneous porous media.

In Section 2, we present the governing equations used for the simulation of a two-phase flow problem. In Section 3, we describe the models used in this work. Later, in Section 4, we give a brief overview of the methods we use to solve the linear systems. Section 5 is devoted to the numerical experiments, where we give some examples and present some results. Finally, we formulate the conclusions.

2. Two phase flow through porous media

For simulation of two-phase flow through a porous medium, we can consider the phases as separated, i.e., they are immiscible and there is no mass transfer between them. The contact area between the phases is known as interface. We usually consider one of the fluids as the wetting phase (w), which is more attracted to the mineral particles than the other phase, known as non-wetting phase (nw). In the case of a water-oil system, water is considered the wetting phase.

The saturation of a phase (S_α) is the fraction of void space filled with that phase in the medium, where a zero saturation indicates that the phase is not present. Fluids inside a reservoir are usually filling completely the empty space, this property is expressed by the following relation for a two-phase system,

$$S_{nw} + S_w = 1. \tag{1}$$

The surface tension and the curvature of the interface between the fluids causes a difference in pressure between the two phases. This difference is known as the capillary pressure (p_c) which depends on the saturation:

$$p_c(S_w) = p_{nw} - p_w. \tag{2}$$

The pressure of the non-wetting fluid is higher than the pressure of the wetting fluid; therefore, the capillary pressure is always a positive quantity. The relation between the capillary pressure and the saturation is an empirical model based on experiments. The capillary curve depends on the difference in pore-size distributions, porosity, and permeability of the medium.

[1] Given a linear system $\mathbf{Ax} = \mathbf{b}$, and the initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0$, with $\mathbf{x}^0$ an initial guess of $\mathbf{x}$, we define the Krylov subspace as $\mathcal{K}_k(\mathbf{A}, \mathbf{r}^0) = span\{\mathbf{r}^0, \mathbf{Ar}^0, \ldots, \mathbf{A}^{k-1}\mathbf{r}^0\}$. That is, the set of linear combinations of powers of $\mathbf{A}$ times $\mathbf{r}^0$.

When modeling two-phase flow, the permeability of each phase, α, will be affected by the presence of the other phase. Therefore, an effective permeability K_α has to be used instead of the absolute permeability K. The absolute and effective permeabilities are realted via the relative permeability, defined as:

$$k_{r\alpha}(S_\alpha) = K_\alpha^e / K, \tag{3}$$

that depends on the saturation, the Corey model gives a relation between these two quatities:

$$k_{rw} = (\hat{S}_w)^{n_w} k_w^0, \qquad k_{rnw} = (1 - \hat{S}_w)^{n_{nw}} k_{nw}^0. \tag{4}$$

where $n_w > 1$, $n_{nw} > 1$ and k_α^0 are fitting parameters.

As in the single-phase case, the governing equations for two-phase flow in a porous medium are the mass conservation principle and Darcy's law. The mass balance equation for a phase α is given by:

$$\frac{\partial(\phi \rho_\alpha S_\alpha)}{\partial t} + \nabla \cdot (\rho_\alpha \mathbf{v}_\alpha) = \rho_\alpha q_\alpha, \tag{5}$$

and the Darcy's law reads:

$$\mathbf{v}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K(\nabla p_\alpha - \rho_\alpha g \nabla d), \tag{6}$$

where ρ_α, μ_α, q_α and p_α are the density, viscosity, sources and pressure of each phase, g is the gravity constant, and d is the depth of the reservoir.

To simplify notation, we introduce the phase mobilities

$$\lambda_\alpha(S_\alpha) = \frac{K k_{r\alpha}(S_\alpha)}{\mu_\alpha}. \tag{7}$$

Combining Darcy's law (6), the mass balance equation (5) and using the phase mobilities, the system reads:

$$\frac{\partial(\phi \rho_\alpha S_\alpha)}{\partial t} - \nabla \cdot (\rho_\alpha \lambda_\alpha (\nabla p_\alpha - \rho_\alpha g \nabla d)) = \rho_\alpha q_\alpha, \tag{8}$$

which is a parabolic equation for pressures and saturations.

The previously-mentioned equations can be separated into a pressure equation and a saturation or transport equation via the fractional flow formulation. For an immiscible, incompressible flow, the pressure equation becomes elliptic and the transport equation becomes hyperbolic. With this formulation, the pressure and transport equations are solved in separate steps in a sequential procedure. In the next subsection we describe in more detail this formulation.

2.1. Fractional flow formulation

In the case of incompressible flow, the porosity ϕ and the densities ρ_α do not depend on the pressure. Therefore, Equation (8) reduces to:

$$\phi \frac{\partial S_\alpha}{\partial t} - \nabla \cdot (\lambda_\alpha (\nabla p_\alpha - \rho_\alpha g \nabla d)) = q_\alpha. \tag{9}$$

Considering a two-phase system with a wetting (w) and a non wetting phase (nw), we need to solve Equation (8) for each phase. To solve it, we define the total Darcy's velocity as the sum of the velocity in both phases:

$$\mathbf{v} = \mathbf{v}_w + \mathbf{v}_{nw} = -(\lambda_{nw} + \lambda_w)\nabla p_{nw} + \lambda_w \nabla p_c + (\lambda_{nw}\rho_{nw} + \lambda_w \rho_w) g \nabla d. \tag{10}$$

If we add the two continuity equations, together with Equation (1) we obtain:

$$\phi \frac{\partial (S_w + S_{nw})}{\partial t} + \nabla \cdot (\mathbf{v}_w + \mathbf{v}_{nw}) = \nabla \cdot \mathbf{v} = q, \tag{11}$$

where $q = q_{nw} + q_w$ is the total source term. Defining the total mobility as $\lambda = \lambda_{nw} + \lambda_w$, and using Darcy's law, Equation (11) becomes:

$$-\nabla \cdot (\lambda \nabla p_{nw}) = q - \nabla [\lambda_w \nabla p_c + (\lambda_{nw}\rho_{nw} + \lambda_w \rho_w) g \nabla d], \tag{12}$$

which is an equation for the pressure of the non wetting phase. This equation depends on the saturation via the capillary pressure p_c and the total mobility λ.

Multiplying each phase velocity by the relative mobility of the other phase and subtracting the result we get:

$$\lambda_{nw}\mathbf{v}_w - \lambda_w\mathbf{v}_{nw} = \lambda_w\lambda_{nw}[\nabla p_c + (\rho_w - \rho_{nw})g\nabla d]. \tag{13}$$

Therefore, for the wetting phase velocity, $\mathbf{v}_w$, we have:

$$\mathbf{v}_w = \frac{\lambda_w}{\lambda}\mathbf{v} + \frac{\lambda_w\lambda_{nw}}{\lambda}[\nabla p_c + (\rho_w - \rho_{nw})g\nabla d]. \tag{14}$$

We introduce the fractional flow function,

$$f_w(S_w) = \frac{\lambda_w(S_w)}{\lambda_w(S_w) + \lambda_{nw}(S_{nw})}, \tag{15}$$

which, together with the previously computed velocity $\mathbf{v}_w$, transforms the transport Equation (5) to:

$$\phi\frac{\partial S_w}{\partial t} + \nabla\cdot[f_w(\mathbf{v}_w + \lambda_{nw}\Delta\rho g\nabla d)] + \nabla\cdot(f_w\lambda_{nw}\nabla p_c) = q_w, \tag{16}$$

where $\Delta\rho = \rho_w - \rho_{nw}$.

With this approach, the system is expressed in terms of the non wetting phase pressure, Equation (12), and the saturation of the wetting phase, Equation (16). In the pressure equation, the coupling to saturation is present via the phase mobilities, Equation (7), and the derivative of the capillary function. For the saturation, we have an indirect coupling with the pressure through the total Darcy velocity, Equation (10).

To solve the system, besides the governing equations, we need to define boundary conditions. The boundary conditions can be prescribed pressures (Dirichlet conditions), flow rates (Neumann conditions) or a combination of these (Robin conditions). Once we have the complete description of our system, we need to discretize it, the discretization methods used in this work are presented in the next section.

3. Discretization methods

In this work, we use the sequential scheme to simulate two-phase flow. With this approach, an unknown is fixed, e.g. the saturation of the wetting phase (S_w), and the resulting elliptic Equation (12) is solved for the pressure of the non-wetting phase (p_{nw}). Once p_{nw} is computed, we update the total velocity ($\mathbf{v}$), Equation (10), and solve the parabolic transport equation for S_w, Equation (16).

The resulting system depends on space and time. The space derivatives are discretized using finite differences; for the temporal discretization we use the backward Euler method. Both discretization methods are presented in this section. In the examples presented in Section 5, the discretization is performed with the Matlab Reservoir Simulation Toolbox (MRST [21]).

Spatial discretization. Using the sequential scheme, for a given time step n, we fix the wetting-phase saturation (S_w^n) and we compute the non-wetting phase pressure (p_{nw}^n), Equation (12). The resulting equation contains only spatial derivatives, that are approximated using cell central differences scheme in a mesh with uniform size Δx, Δy, Δz. We compute the pressure $p_{i,j,l} = p^n(x_i, y_j, z_l)$ at the center of the cell (i, j, l), and the harmonic average of the mobility, $\lambda_{i-\frac{1}{2},j,l} = \lambda_{i-\frac{1}{2},j,l}(S^n)$, at the interface between cells $(i-1, j, l)$ and (i, j, l). The derivative in the x direction becomes (see, e.g. [22, 23, 24, 25]):

$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial p_{nw}}{\partial x}\right) = \frac{\lambda_{i+\frac{1}{2},j,l}(p_{i+1,j,l} - p_{i,j,l}) - \lambda_{i-\frac{1}{2},j,l}(p_{i,j,l} - p_{i-1,j,l})}{(\Delta x)^2} + \mathcal{O}(\Delta x^2). \tag{17}$$

Defining the *transmissibility*, $T_{i-\frac{1}{2},j,l}$, between grid cells $(i-1, j, l)$ and (i, j, l) as:

$$T_{i-\frac{1}{2},j,l} = \frac{2\Delta y\Delta z}{\mu\Delta x}\lambda_{i-\frac{1}{2},j,l}, \tag{18}$$

together with boundary conditions, Equation (12) is rewritten as:

$$\mathbf{T}\mathbf{p}_{nw}^n = \mathbf{q}, \tag{19}$$

where $\mathbf{T}$ is known as the transmissibility matrix. This system is SPD; therefore, we use the CG method to solve it throughout this work. More information about CG is given in Section 5.

Well model. In reservoir simulation, besides boundary conditions, we can also have sources, that are fluids injected or extracted through wells or through boundaries. To describe the injection or production through wells, we use the Peaceman well model. This model gives a linear relationship between the *bhp* and the flow rate via the productivity or injectivity index $I_{(i,j,l)}$ of the well. This relationship is given by:

$$q_{(i,j,l)} = I_{(i,j,l)}(p_{(i,j,l)} - p_{bh(i,j,l)}), \tag{20}$$

for a cell (i, j, l) that contains the well. In Equation (20), $p_{(i,j,l)}$ is the reservoir pressure in the cell containing the well and $p_{bh(i,j,l)}$ is a prescribed pressure inside the well.

Incompressible fluid. Combining Equation (19) with Equation (20) we obtain:

$$\mathbf{T}\mathbf{p}^n_{nw} = \mathbf{I}_w(\mathbf{p}^n_{nw} - \mathbf{p}^n_{bh}), \tag{21}$$

where $\mathbf{I}_w$ is a diagonal matrix containing the productivity or injectivity indices of the wells present in the reservoir.

Temporal discretization. Once we have computed the pressure of the non-wetting phase (p_{nw}), we update the Darcy velocity ($\mathbf{v}^n$), Equation (10). This velocity is then update in the transport Equation (16). This equation depends on time; thus, we need to discretize the temporal derivative. This discretization can be performed using two schemes: implicit and explicit.

In the explicit scheme, the time derivative is approximated using the fractional flow, mobilities, capillary pressure and Darcy velocity computed in the previous time step. After the update, the system reads:

$$\phi\frac{(S_w^{n+1} - S_w^n)}{\Delta t} + \nabla\cdot[f_w(S_w^n)(\mathbf{v}^n + \lambda_{nw}\Delta\rho g\nabla z)] + \nabla\cdot(f_w(S_w^n)\lambda_{nw}(S_w^n)\nabla p_c(S_w^n)) = q_w^{n+1}. \tag{22}$$

For the implicit solution we use the backward Euler time discretization scheme that transforms Equation (16) into:

$$\phi\frac{(S_w^{n+1} - S_w^n)}{\Delta t} + \nabla\cdot[f_w(S_w^{n+1})(\mathbf{v}^n + \lambda_{nw}\Delta\rho g\nabla z)] + \nabla\cdot(f_w(S_w^{n+1})\lambda_{nw}(S_w^{n+1})\nabla p_c(S_w^{n+1})) = q_w^n, \tag{23}$$

or:

$$S_w^{n+1} - S_w^n - \frac{\Delta t}{\phi}\left(q_w - \nabla\cdot[f_w(S_w^{n+1})(\mathbf{v}^n + \lambda_{nw}\Delta\rho g\nabla z)]\right) + \frac{\Delta t}{\phi}\left(\nabla\cdot(f_w(S_w^{n+1})\lambda_{nw}(S_w^{n+1})\nabla p_c(S_w^{n+1}))\right) = 0. \tag{24}$$

If we use the implicit scheme, the resulting system is nonlinear, Equation (24), and depends on the saturation at time steps n and $n + 1$. The nonlinear system can be solved using, e.g., the Newton-Raphson (NR) method. With this method, for the $(k + 1)$-th iteration we have:

$$J(S^k)\delta S^{k+1} = -F(S^k, S^n), \qquad S^{k+1} = S^k + \delta S^{k+1}, \tag{25}$$

where $J(S^k) = \frac{\partial F(S^k, S^n)}{\partial S^k}$ is the Jacobian matrix, δS^{k+1} is the NR update at iteration step $k + 1$ and $F(S^k, S^n)$ is given by Equation 24. We solve Equation 25 for δS^{k+1} and we update the saturation of the actual time step. Then, we compute the pressure for this time step p^{n+1}_{nw}, and we repeat the process for the rest of the time steps.

4. Solution methods for linear systems

Iterative techniques are preferred over direct methods to approximate the solution of ill-conditioned and large linear systems, being the CG preconditioned with IC a popular choice to solve SP(S)D systems. In this work, we study a further acceleration with deflation and POD techniques. In this section, we give a brief overview of these methods.

Conjugate Gradient (CG) Method. is a Krylov-subspace method used to solve systems with SPD matrices. The pseudo code for CG is given in Algorithm 2.

Algorithm 2. Conjugate Gradient (CG) method, solving $\mathbf{Ax} = \mathbf{b}$.

Give an initial guess $\mathbf{x}^0$.
Compute $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0$ and set $\mathbf{p}^0 = \mathbf{r}^0$.
for $k = 0, ...,$ until convergence
$\alpha^k = \frac{(\mathbf{r}^k, \mathbf{r}^k)}{(\mathbf{Ap}^k, \mathbf{p}^k)}$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{Ap}^k$
$\beta^k = \frac{(\mathbf{r}^{k+1}, \mathbf{r}^{k+1})}{(\mathbf{r}^k, \mathbf{r}^k)}$
$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta^k \mathbf{p}^k$
end

Preconditioning. To accelerate the convergence of an iterative method, the linear system is multiplied by a matrix $\mathbf{M}^{-1}$ such that the iteration matrix has a better spectrum and $\mathbf{M}^{-1}\mathbf{b}$ is cheap to compute, the resulting preconditioned system is:

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}. \tag{26}$$

Deflation. Sometimes, there are a few extreme eigenvalues hampering the convergence of an iterative method, with deflation [15], the effect of these eigenvalues can be annihilated.

Given an *SPD* matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, the deflation matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ is defined as follows [20, 17]:

$$\mathbf{P} = \mathbf{I} - \mathbf{AQ}, \quad \mathbf{Q} = \mathbf{ZE}^{-1}\mathbf{Z}^T \quad \text{and} \quad \mathbf{E} = \mathbf{Z}^T\mathbf{AZ}, \tag{27}$$

where $\mathbf{E} \in \mathbb{R}^{m \times m}$ is known as the *Galerkin* or *coarse* matrix, the full rank matrix $\mathbf{Z} \in \mathbb{R}^{n \times m}$ is called the *deflation − subspace* matrix, and it's columns are the *deflation* vectors or *projection* vectors. These vectors have to be selected and, usually, a good selection depends on the problem. The selection of deflation vectors is mainly based on approximated eigenvectors, recycling solutions [26, 11], subdomain deflation vectors [16], or multigrid and multilevel-based deflation matrices [17, 27].

Proper Orthogonal Decomposition (POD). Essential system information is captured with POD in a small set of orthonormal basis vectors $\mathbf{\Psi} = [\psi_1\ \psi_2\ ..\ \psi_l]$, $\mathbf{\Psi} \in \mathbb{R}^{n \times l}$. This basis can be used to project a high-order model onto a space spanned by this basis. The basis vectors $\psi_i \in \mathbb{R}^n$ are computed from a set of 'snapshots', obtained by simulation or experiments [7], $\mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_m]$. These vectors $\{\psi_j\}_{j=1}^l$ are l eigenvectors corresponding to the largest eigenvalues $\{\sigma_j\}_{j=1}^l$ of the data snapshot correlation matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$,

$$\mathbf{R} := \frac{1}{m}\mathbf{XX}^T \equiv \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^T. \tag{28}$$

Ordering the eigenvalues σ_j from large to small, σ_1 being the largest, the basis consist on the eigenvalues of the l eigenvalues satisfying [7]:

$$\frac{\sum_{j=1}^{l} \sigma_j}{\sum_{j=1}^{m} \sigma_j} \leq \alpha, \qquad 0 < \alpha \leq 1, \tag{29}$$

with α close to 1. This basis contains almost all the system's variability; Therefore, the high-dimensional variable $\mathbf{x} \in \mathbb{R}^n$ can be approximated by a linear combination of these basis vectors [6][2]:

$$\mathbf{x} \approx \sum_{i=1}^{l} c_i \psi_i. \tag{30}$$

[2] In this study, we normalize the snapshots, so that $\|\mathbf{x}_i\|_2 = 1$.

POD-based deflation method. We propose the reuse of a POD basis, $\mathbf{\Psi}$, as subspace-deflation matrix, $\mathbf{Z}$, in a deflation procedure. We implement the Deflated Preconditioned Conjugate Gradient method preconditioned with Incomplete Cholesky (DICCG). With deflation, we remove some of the eigenvalues of the system's matrix $\mathbf{A}_t$ that cause a slow convergence by making use the system's information contained in the POD basis. To obtain the basis, a set of snapshots, $\mathbf{X}$, is required. We propose a *moving window* and a *training simulation* approaches to obtain the snapshots.

Moving window We start the simulation by computing a set of s snapshots and obtaining a POD basis from it. We solve the rest of the time steps with the DICCG method using the vectors of the POD basis as deflation vectors. The basis and, as a consequence, the deflation matrix is updated at each time step. The pseudo code is given in Algorithm 3. Note that, with this approach, the first s time steps are computed using the ICCG method.

Training simulation. We run the simulation for all the time steps with the ICCG method. During this simulation (*training phase*), we randomly vary the pressure in the production wells. A POD basis $\mathbf{\Psi}$ is computed from the solutions of the *training phase*. Later, $\mathbf{\Psi}$ is used as deflation-subspace matrix to solve a series of problems with the same conditions, but with different pressures in the wells, i.e., different *rhs*. The pseudo code is presented in Algorithm 4.

Algorithm 3. Deflation, moving window variant, solving $\mathbf{A}_t\mathbf{x}_t = \mathbf{b}_t$.

% Compute the solution of the first s time steps with ICCG.
for $t = 1, \quad ..., \quad s$
$\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}$
end
$X_{1:s} = \{x_1, x_2, ..., x_s\}^a$
% Compute the POD basis from the correlation matrix $\mathbf{R}^b$.
$\mathbf{\Psi}_{1:s} = \{\psi_1, ..., \psi_s\}$
% Compute the next solutions with DICCG.
for $t = s + 1, \quad ..., \quad$ steps
$\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}$
end

Algorithm 4. Deflation, training variant, solving $\mathbf{A}_t\mathbf{x}_t = \mathbf{b}_t$.

% Run a training phase simulation with ICCG varying the pressure in the wells.
for $t = 1, \quad ..., \quad steps$
$\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}_t, \qquad \mathbf{b}_t = rand$
end
$X_{1:steps} = \{x_1, x_2, ..., x_{steps}\}$
% Compute the POD basis from the correlation matrix $\mathbf{R}$
$\mathbf{\Psi}_{1:s} = \{\psi_1, ..., \psi_s\}$
% Run the simulation with fixed pressures DICCG.
for $t = 1, \quad ..., \quad$ steps
$\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}_t$
end

[a] We define $X_{a:b} := \{x_a, x_{a+1}, ..., x_b\}$.
[b] $\mathbf{R} := \frac{1}{s}\mathbf{X}\mathbf{X}^T$

We use this methodology to compute the solution of the pressure, Equation (12). To analyze the method's performance, we compare the total number of iterations necessary to run the DICCG simulation with the total number of iterations necessary to solve the same problem using the non-deflated method (ICCG).

The computational cost to solve one time step with the ICCG method is $31N$ for a 2D and $39N$ for a 3D problem of size N. For the DICCG method using d deflation vectors the cost is $(31 + 4d)N$ for 2D and $(39 + 4d)N$ for 3D. Which implies that the DICCG method requires $\sim 1 + \frac{4d}{30}$ of ICCG operations for the 2D case, and $\sim 1 + \frac{d}{10}$ for the 3D case (see [28]).

As tolerance or stopping criterion we use the relative residual, defined as the 2-norm of the residual of the k^{th} iteration divided by the 2-norm of the right-hand side of the preconditioned system, $\|\mathbf{M}^{-1}r^k\|_2/\|\mathbf{M}^{-1}b\|_2 \leq \epsilon$. The tolerance of the solvers is presented for each case.

5. Numerical experiments: results and discussion

We study the solution of systems of linear equations for the pressure resulting from water flooding for immiscible fluids (oil and water) flowing in highly heterogeneous 2D and 3D reservoirs. We include capillary pressure and gravity terms in some of the case studies. We use the fractional flow formulation to decouple pressure from saturation and we solve the resulting system with sequential schemes. The linear pressure system is obtained with MRST and solved using the proposed POD-based deflation procedure. The transport equation is solved with MRST using implicit schemes. The models studied in this work are: an academic layered reservoir with a contrast in permeability coefficients up to 10^6, and the SPE 10 benchmark with a contrast of 10^7 [29].

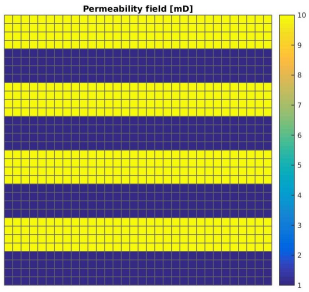

Fig. 1: Rock permeability

Table 1: Fluids properties.

	Water	Oil	Units
μ	1	10	cp
ρ	1000	700	kg/m^3
k_r	$(S_w)^2$	$(1-S_w)^2$	
C_p	$10 * (1-S)$		bars

5.1. Heterogeneous permeability layers

In this section, we study water injection in an academic system consisting of equal-sized layers with a constant porosity field of 0.2 and different permeability values (see Figure 1). A set of layers with permeability $\kappa_1 = 1mD$ is followed by layers with permeability $\kappa_2 = 1$ or $6mD$. The domain consists of a Cartesian grid of 32 x 32 cells in the 2D case, and 24 x 24 x 24 cells for the 3D case. For the relative permeability we use the Corey model, with exponents $n_w = n_{nw} = 2$ (see Table 1). The first set of experiments do not consider gravity terms and capillary pressure. Later, we include capillary pressure, the relationship used is $P_c = C(1-S)$. Finally, we study a 3D problem with gravity terms included.

We study water flooding with injection through the boundary and through wells. When using wells, one setup consists of one injector (I) and one producer (P) placed on opposite corners of the reservoir. For the second setup we place four producers (P_i) on the corners and one injector in the center. The wells are controlled prescribing the *bhp*.

The simulation is run during 4800 days with 240 time steps of 20 days (See Table 2). The stopping criterium for the ICCG and DICCG methods is $\epsilon = 5 \cdot 10^{-8}$.

Table 2: Boundary conditions and temporal parameters.

Temporal parameters			Boundary conditions		
T_{total}	4800	days	$Q_{x=0}$	0.4	m^3/day
T_{steps}	240		$P_{0,x\neq(0,L_x)}$	100	$bars$
dT	20	days	$P_{x=L_x}$	0	$bars$

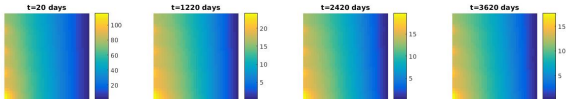

Fig. 2: Pressure field [bars] for various times, for a contrast between permeability values of 10^1, 32 x 32 grid cells.

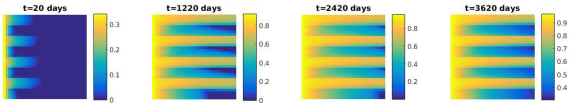

Fig. 3: Water saturation for various times, for a contrast between permeability values of 10^1, 32 x 32 grid cells.

5.1.1. Injection through the left boundary

Injection is performed through the left boundary at a rate of 0.4 m^3/day. The pressure is set as zero at the right boundary and 100 bars inside the reservoir (See Table 2).

The pressure field and the water saturation are presented in Figure 2 and Figure 3 for various times. We observe that the pressure is larger on the boundary where water is injected and it decreases towards the right boundary. We note that the water flows easily through the layers with higher permeability.

The number of iterations necessary to achieve convergence is summarized in Table 3. The first column contains the contrast between permeability layers (κ_1/κ_2). In the second, we present the number of deflation d vectors used. The third column shows the number of iterations necessary to achieve convergence with the ICCG method only. The number of iterations necessary to compute the snapshots with the DICCG method is presented in the 4th and 5th columns (DICCG). For this examples we use the *moving window* approach. Therefore, it is required to compute the first d snapshots with ICCG (fourth column), the rest of the time steps are computed with DICCG (fifth column). The total number of iterations needed to perform the DICCG method (d time steps computed with ICCG + Total-d computed with DICCG) are presented in the sixth column. In the last column, we compute the percentage of DICCG iterations with respect to ICCG.

A reduction to around 15% the number of ICCG iterations was achieved with the DICCG method for most of the cases (see Table 3). This reduction is similar when we use a contrast in permeability layers of 10^1 or 10^6.

In Figure 4, we note that a contrast on the permeability layers of 10^1 or 10^6 results in five eigenvalues significantly larger than the rest. Therefore, if we use five instead of ten POD basis vectors as deflation vectors the results are similar, as noted in Table 3. For the case with higher contrast, the spectrum of the correlation matrix is slightly more spread, which could explain the minor increase in the number of iterations.

Capillary pressure case.. The results show that the capillary pressure influences the performance of the DICCG method increasing the number of iterations required to find an approximate solution (see Table 3). To further investigate the performance of the DICCG method, we study two cases. In the first one, we increase the number of deflation vectors to 20. For the second one, we use a smaller time step (half of the previous) using five and ten deflation vectors. The results are presented in Table 4.

If we increase the number of deflation vectors, the performance of the DICCG method does not change considerably (see Table 4). Therefore, we cannot conclude that the number of deflation vectors has a direct influence on

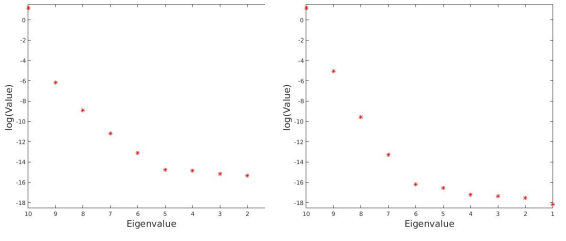

Fig. 4: Eigenvalues of the correlation matrix $\mathbf{R} = \frac{1}{m}\mathbf{X}\mathbf{X}^T$ for a contrast between permeability values of 10^1 and 10^6.

Table 3: Number of iterations. Injection through the left boundary.

$\frac{\kappa_2}{\kappa_1}$	d	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
No capillary pressure included, 2D						
10^1	10	14783	605	1270	1875	13
10^1	5	14783	605	1573	2178	15
10^6	10	9735	442	1163	1605	16
10^6	5	9735	442	1317	1759	18
Capillary pressure included, 2D						
10^1	10	14597	600	2843	3443	24
10^1	5	14597	600	3072	3672	25
10^6	10	9071	428	3156	3584	40
10^6	5	9071	428	2644	3072	34
No capillary pressure included, 3D						
10^1	10	17496	656	1800	2456	14
10^1	5	17496	656	2170	2826	16
10^6	10	15720	486	1972	2458	16
10^6	5	15720	486	2121	2607	17
Capillary pressure included, 3D						
10^1	10	17224	660	3431	4091	24
10^1	5	17224	660	3658	4318	25
10^6	10	13228	469	3531	4000	30
10^6	5	13228	469	2992	3461	26

Table 4: Number of iterations. Capillary pressure examples.

$\frac{\kappa_2}{\kappa_1}$	d	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
Capillary pressure included, 2D						
10^1	10	14597	600	2843	3443	24
10^1	5	14597	600	3072	3672	25
10^6	10	9071	428	3156	3584	40
10^6	5	9071	428	2644	3072	34
Capillary pressure included, 2D, 20 dv						
10^1	20	14597	1210	2347	3557	24
10^1	11	14597	1210	2486	3696	25
10^6	20	9071	837	3229	4066	45
10^6	11	9071	837	2927	3764	41
Capillary pressure included, 2D, smaller time step						
10^1	10	29187	585	4166	4751	16
10^1	5	29187	585	4463	5048	17
10^6	10	18400	393	4623	5016	27
10^6	5	18400	393	4005	4398	24

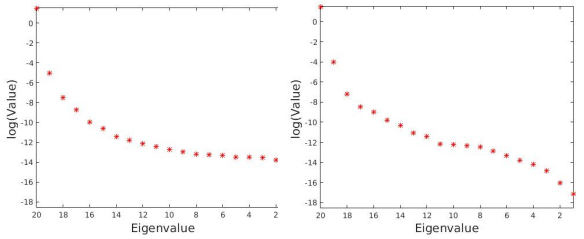

Fig. 5: Eigenvalues of the correlation matrix $\mathbf{R} = \frac{1}{m}\mathbf{X}\mathbf{X}^T$ for a contrast between permeability values of 10^1 and 10^6, 20 deflation vectors.

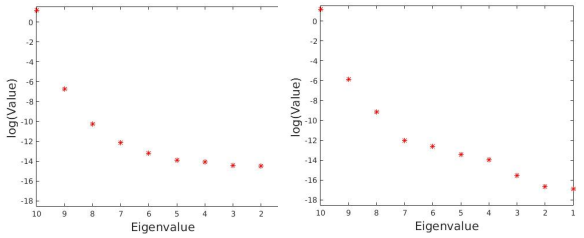

Fig. 6: Eigenvalues of the correlation matrix $\mathbf{R} = \frac{1}{m}\mathbf{X}\mathbf{X}^T$ for a contrast between permeability values of 10^1 and 10^6, 480 time steps.

the behavior of the method. This can be explained from the spectrum of the correlation matrix (see Figure 5). The range of the spectrum is similar for ten or twenty basis vectors. Therefore, the information captured in both cases is basically the same and the performance of the method does not differ.

However, if we use 480 time steps instead of the 240 of the previous cases, we observe that the DICCG method performs better. If the time step is smaller, the matrix $\mathbf{A}$ has less variation during each time step. The smallest eigenvalue of the correlation matrix for this problem is 10^{-15} for a contrast of 10^1 and 10^{-17} for a contrast of 10^6 (see Figure 6). Therefore, the difference between these values is the same as in the previous cases, but the smallest value is smaller than in the previous cases, which appears to result in a better performance.

From these experiments, we observe that the performance of the DICCG method depends on the time step, i.e., the variation in the matrix $\mathbf{A}$. This can be linked to the information acquired with the snapshots. For the case without capillary pressure, the time step used is enough to capture the system behavior. On the contrary, including capillary pressure, smaller changes produced in the system have to be considered, which can be done by taking smaller time steps.

Acknowledgments

References

Reference style

Text: All citations in the text should refer to:

1. Single author: the author's name (without initials, unless there is ambiguity) and the year of publication;
2. Two authors: both authors' names and the year of publication;
3. Three or more authors: first author's name followed by 'et al.' and the year of publication.

Citations may be made directly (or parenthetically). Groups of references should be listed first alphabetically, then chronologically.

References

[1] P.T.M. Vermeulen, A.W. Heemink and C.B.M. Te Stroet, Reduced models for linear groundwater flow models using empirical orthogonal functions., Advances in Water Resources 27 (2004) 57–69.

[2] D. Pasetto, M. Ferronato and M. Putti, A reduced order model-based preconditioner for the efficient solution of transient diffusion equations, International Journal for Numerical Methods in Engineering 109 (2017) 1159–1179.
[3] W. Schilders, H. Van der Vorst and J. Rommes, Model order reduction: theory, research aspects and applications, volume 13, Springer, 2008.
[4] A. Quarteroni and G. Rozza, Reduced order methods for modeling and computational reduction, volume 9, Springer, 2014.
[5] K. Carlberg, V. Forstall and R. Tuminaro, Krylov-subspace recycling via the POD-augmented conjugate-gradient method, SIAM Journal on Matrix Analysis and Applications 37 (2016) 1304–1336.
[6] P. Astrid, G. Papaioannou, J.C. Vink and J.D. Jansen, Pressure Preconditioning Using Proper Orthogonal Decomposition., in: 2011 SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 2011, pp. 21–23.
[7] R. Markovinović and J.D. Jansen, Accelerating iterative solution methods using reduced-order models as solution predictors., International journal for numerical methods in engineering 68 (2006) 525–541.
[8] M.A. Cardoso, L.J. Durlofsky and P. Sarma, Development and application of reduced-order modeling procedures for subsurface flow simulation, International journal for numerical methods in engineering 77 (2009) 1322–1350.
[9] T. Heijn, R. Markovinovic, J.D. Jansen et al., Generation of low-order reservoir models using system-theoretical concepts, SPE Journal 9 (2004) 202–218.
[10] J. van Doren, R. Markovinović and J.D. Jansen, Reduced-order optimal control of water flooding using proper orthogonal decomposition, Computational Geosciences 10 (2006) 137–158.
[11] G.B. Diaz Cortes, C. Vuik and J.D. Jansen, On POD-based Deflation Vectors for DPCG applied to porous media problems, Journal of Computational and Applied Mathematics 330 (2018) 193 – 213.
[12] Y. Saad, Iterative Methods for Sparse Linear Systems., Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2nd ed., SIAM, 2003.
[13] H.A. Van der Vorst, Iterative Krylov methods for large linear systems, volume 13, Cambridge University Press, 2003.
[14] H.A. Van der Vorst, C. Dekker, Conjugate gradient type methods and preconditioning, Journal of Computational and Applied Mathematics 24 (1988) 73–87.
[15] C. Vuik, A. Segal and J.A. Meijerink, An Efficient Preconditioned CG Method for the Solution of a Class of Layered Problems with Extreme Contrasts in the Coefficients., Journal of Computational Physics 152 (1999) 385.
[16] C. Vuik, A. Segal, L. Yaakoubi and E. Dufour, A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients., Applied Numerical Mathematics 41 (2002) 219–233.
[17] J.M. Tang, R. Nabben, C. Vuik and Y. Erlangga, Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods, Journal of scientific computing 39 (2009) 340–370.
[18] M. Benzi, Preconditioning techniques for large linear systems: a survey, Journal of computational Physics 182 (2002) 418–477.
[19] J. Tang and C. Vuik, On deflation and singular symmetric positive semi-definite matrices, Journal of computational and applied mathematics 206 (2007) 603–614.
[20] J. Tang, Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems, Ph.D. thesis, Delft University of Technology, 2008.
[21] K.A. Lie, An Introduction to Reservoir Simulation Using MATLAB: User guide for the Matlab Reservoir Simulation Toolbox (MRST)., SINTEF ICT, 2013.
[22] K. Aziz and A. Settari, Petroleum reservoir simulation, Chapman & Hall, 1979.
[23] Z. Chen, G. Huan and Y. Ma, Computational methods for multiphase flows in porous media, SIAM, 2006.
[24] J.D. Jansen, A systems description of flow through porous media., Springer, 2013.
[25] G.B. Diaz Cortes, C. Vuik and J.D. Jansen, Physics-based pre-conditioners for large-scale subsurface flow simulation, Technical Report, Delft University of Technology, Department of Applied Mathematics, 2016.
[26] M. Clemens, M. Wilke, R. Schuhmann and T. Weiland, Subspace projection extrapolation scheme for transient field simulations., IEEE Transactions on Magnetics 40 (2004) 934–937.
[27] B. Smith, P. Bjorstad and W. Gropp, Domain decomposition: parallel multilevel methods for elliptic partial differential equations., Cambridge University Press New York, 1996.
[28] G.B. Diaz Cortes, C. Vuik and J.D. Jansen, POD-based deflation techniques for the solution of two-phase flow problems in large and highly heterogeneous porous media., Report 18-1, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2018.
[29] M.A. Christie and M.J. Blunt, Tenth SPE Comparative Solution Project: a Comparison of Upscaling Techniques., SPE Reservoir Engineering and Evaluation 4 (2001) 308–317.

Supplementary Material

Supplementary material that may be helpful in the review process should be prepared and provided as a separate electronic file. That file can then be transformed into PDF format and submitted along with the manuscript and graphic files to the appropriate editorial office.