DELFT UNIVERSITY OF TECHNOLOGY

REPORT 18-01

POD-based deflation techniques for the solution of two-phase flow problems in large and highly heterogeneous porous media.

G. B. Diaz Cortes, C. Vuik, J. D. Jansen

ISSN 1389-6520

Reports of the Delft Institute of Applied Mathematics

Delft 2018

POD-based deflation techniques for the solution of two-phase flow problems in large and highly heterogeneous porous media.

G. B. Diaz Cortes[1], C. Vuik[1] and J. D. Jansen[2]

[1]Department of Applied Mathematics, TU Delft
[2]Department of Geoscience & Engineering, TU Delft

Abstract

Simulation of two-phase flow through highly heterogeneous porous media results in ill-conditioned large systems of linear equations for the pressure when using, e.g., sequential procedures. Solving the resulting linear system can be particularly time-consuming. Therefore, there have been extensive efforts to find ways to address this issue effectively.

Iterative methods, together with preconditioning techniques [1, 2], are the most commonly chosen techniques to solve these problems. In the literature, we can also find Reduced Order Models (ROM) [3–5] and deflation methods [6, 7], where system information is reused to find a good approximation more quickly. For the deflation techniques, an optimal selection of deflation vectors is crucial for a good performance. The construction of deflation vectors based on information captured with ROM, in particular, Proper Orthogonal Decomposition (POD), was recently presented for a single-phase flow problem [8, 9].

The goal of this work is to further explore and develop the possibilities of combining POD and deflation techniques for a two-phase flow simulation. We propose selecting deflation vectors from a POD basis in two different ways. In the first one, we obtain the basis on-the-fly during the simulation, to accelerate the upcoming time steps. For the second one, the basis is obtained off-line in a training phase, and it is used later to solve slightly different problems.

The convergence properties of the resulting POD-based deflation method is studied for an incompressible two-phase flow problem in heterogeneous porous media, presenting a contrast in permeability coefficients of $\mathcal{O}(10^7)$. We compare the number of iterations required to solve the above-mentioned problem using the Conjugate Gradient method preconditioned with Incomplete Cholesky (ICCG), against the deflated version of the same method (DICCG).

The efficiency of the method is illustrated with the SPE 10 benchmark, our largest test case, containing $\mathcal{O}(10^6)$ cells. For this problem, the DICCG method requires only 20% of the number of ICCG iterations.

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 [5, 10–13], which is easier to solve. These methods show that essential system information can be obtained by computing a set of basis functions from a collection of system solutions (also known as 'snapshots'). Proper Orthogonal Decomposition (POD) is a ROM method that has recently been used to accelerate the solution of the linear pressure equation resulting from reservoir simulation [3, 4, 14–16], among other applications. With POD procedures, only a few basis functions capture most of the system variation [14].

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. A POD basis is obtained from these snapshots and used to solve slightly different problems. This approach is especially suited to solve problems with small changes in the input variables, e.g. the same well configurations but different flow rates or bottom hole pressures (*bhp*) [3, 14, 15]. The basis can also be computed 'on-the-fly', using, e.g., the solution of the latest time steps [3, 4, 8]. 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. For the solution of a large-scale system, Markovinovic et al. [4] 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. [3]. Another approach was developed by Pasetto et al. [5], 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. [8].

For many applications, Krylov subspace iterative methods are used [1, 17][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 Conjugate Gradient (CG) [7, 13, 18–20], for which the incomplete Cholesky factorization is a popular preconditioning choice [2, 18].

In recent years, deflation techniques have been developed to accelerate the convergence of Krylov subspace methods [7, 19–22]. For a good performance of this method, a deflation subspace needs to be found, such that, the smallest eigenvalues of the system are no longer

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

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 for the construction of the above-mentioned deflation subspace. For a single-phase incompressible problem, the resulting linear pressure system is of the form $\mathbf{Ax} = \mathbf{b}$. If the deflation matrix consists in all the linearly-independent (l.i.) systems $\mathbf{Ax_i} = \mathbf{b_i}$, the convergence is achieved in one iteration [8, 23]. In that case, the l.i. systems have the same matrix $\mathbf{A}$ and different *rhs*. For a two-phase flow problem, the system matrix varies for each time step $\mathbf{A^{(n)}x^{(n)}} = \mathbf{b^{(n)}}$. However, in our examples, the change in the $\mathbf{A^{(n)}}$ matrix is small enough that information can be captured with POD and it can be used to accelerate the convergence of the solution of similar problems.

We explore the applicability of this methodology with single and two-phase flow problems 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 the 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.

When modeling two phases, 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 sum of all of the phase permeabilities is less than one, due to interfacial tensions. This can be expressed as:

$$\frac{\sum_\alpha K_\alpha^e}{K} < 1.$$

A variable relating the absolute and effective permeabilities is the saturation-dependent relative permeability, which is defined as:

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

The saturation dependence of the relative permeabilities can be expressed with the Corey model:

$$\begin{aligned} k_{rw} &= (\hat{S}_w)^{n_w} k_w^0, \\ k_{rnw} &= (1 - \hat{S}_w)^{n_{nw}} k_{nw}^0. \end{aligned} \tag{3}$$

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 and Darcy's law. The mass balance equation for a phase α is given by:

$$\frac{\partial(\phi \rho_\alpha S_\alpha)}{\partial t} + \nabla \cdot (\rho_\alpha \mathbf{v}_\alpha) = \rho_\alpha q_\alpha, \tag{4}$$

and the Darcy's law reads:

$$\mathbf{v}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K (\nabla p_\alpha - \rho_\alpha g \nabla d). \tag{5}$$

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

$$\lambda_\alpha(S_\alpha) = \frac{K k_{r\alpha}(S_\alpha)}{\mu_\alpha}. \tag{6}$$

Combining Darcy's law (5), the mass balance equation (4) and using the phase mobilities, the system reads:

$$\frac{\partial(\phi \rho_\alpha S_\alpha)}{\partial t} - \nabla \cdot (\rho_\alpha \lambda_\alpha (\nabla p_\alpha - \rho_\alpha g \nabla d)) = \rho_\alpha q_\alpha, \tag{7}$$

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 elliptical and the transport equation becomes hyperbolic. With this formulation, the pressure and transport equations are solved in separate steps. In the next subsection we describe in more detail the fractional flow formulation.

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 (7) reduces to:

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

Taking a two-phase system with a wetting (w) and a non wetting phase (nw), the resulting governing equations are:

$$\begin{aligned} \phi \frac{\partial S_w}{\partial t} + \nabla \cdot \mathbf{v}_w =& \phi \frac{\partial S_w}{\partial t} + \nabla \cdot (\lambda_w (\nabla p_w - \rho_w g \nabla d)) = q_w, \\ \phi \frac{\partial S_{nw}}{\partial t} + \nabla \cdot \mathbf{v}_{nw} =& \phi \frac{\partial S_{nw}}{\partial t} + \nabla \cdot (\lambda_{nw} (\nabla p_{nw} - \rho_{nw} g \nabla d)) = q_{nw}. \end{aligned} \tag{9}$$

To solve this system, we define the total Darcy's velocity as the sum of the velocity in the wetting and non wetting phases:

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

If we add the two continuity equations (System (9)) and use the relationship $S_{nw} + S_w = 1$, we have:

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

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, together with Equation (10), we get:

$$\begin{aligned} \lambda_{nw} \mathbf{v}_w - \lambda_w \mathbf{v}_{nw} &= \lambda \mathbf{v}_w - \lambda_w \mathbf{v} \\ &= \lambda_w \lambda_{nw} [\nabla p_c + (\rho_w - \rho_{nw}) g \nabla d]. \end{aligned}$$

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

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

We introduce the fractional flow function,

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

which, together with the previously computed velocity $\mathbf{v}_w$, transforms the transport Equation (4) to:

$$\phi \frac{\partial S_w}{\partial t} + \nabla \cdot [f_w(\mathbf{v} + \lambda_{nw} \Delta\rho g \nabla d)] + \nabla \cdot (f_w \lambda_{nw} \nabla p_c) = q_w, \tag{14}$$

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 (14). In the pressure equation, the coupling to saturation is present via the phase mobilities (Equation (6)), 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).

Besides the governing equations, we need to define boundary conditions. These conditions can be prescribed pressures (Dirichlet conditions), flow rates (Neumann conditions) or a combination of these (Robin conditions). A description of the discretization methods used in this work is presented in the next section.

3 Discretization methods

In this work, we use the sequential procedure 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 (14).

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 [24]).

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). Then, the system to solve is Equation (12). This equation contains only spatial derivatives that are approximated using a finite difference scheme, in particular cell central differences. We use a mesh with a uniform grid size Δx, Δy, Δz where (i, j, l) is the center of the cell in the position i in the x direction, j in the y direction, and l in the z direction (x_i, y_j, z_l), and $p_{i,j,l} = p_{nw}^n(x_i, y_j, z_l)$ is the pressure at this point. Using the harmonic average ($\lambda_{i-\frac{1}{2},j,l} = \lambda_{i-\frac{1}{2},j,l}(S^n)$) for the mobility at the interface between cells $(i-1, j, l)$ and (i, j, l), the derivative in the x direction becomes (see, e.g. [25–28]):

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

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

$$T_{i-\frac{1}{2},j,l} = \frac{2\Delta y \Delta z}{\mu \Delta x} \lambda_{i-\frac{1}{2},j,l}, \tag{15}$$

Using the previouslydefined transmissibility, Equation (12), together with boundary conditions, is rewritten as:

$$\mathbf{T}\mathbf{p}_{nw}^{n} = \mathbf{q}, \tag{16}$$

where $\mathbf{T}$ is known as the transmissibility matrix. This system is SPD; therefore, we use the Conjugate Gradient (CG) iterative method to solve it throughout this work. More information about this method is given in Section 4.

Well model

In reservoir simulation, besides boundary conditions, we can also have sources. They 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:

$$q_{(i,j,l)} = I_{(i,j,l)}(p_{(i,j,l)} - p_{bh(i,j,l)}), \tag{17}$$

for a cell (i,j,l) that contains the well. In Equation (17), $p_{(i,j,l)}$ 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 (16) with Equation (17) we obtain:

$$\mathbf{T}\mathbf{p}_{nw}^{n} = \mathbf{I}_w(\mathbf{p}_{nw}^{n} - \mathbf{p}_{bh}^{n}), \tag{18}$$

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 (14). 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:

$$\phi \frac{(S_w^{n+1} - S_w^n)}{\Delta t} + \nabla \cdot [f_w(S_w^n)(\mathbf{v}^n + \lambda_{nw}\Delta\rho g \nabla z)] + \nabla \cdot (f_w(S_w^n)\lambda_{nw}(S_w^n)\nabla p_c(S_w^n)) = q_w^{n+1}. \tag{19}$$

For the implicit solution, a backward Euler time discretization scheme can be used. With this scheme, Equation (14) is:

$$\phi\frac{(S_w^{n+1}-S_w^n)}{\Delta t}+\nabla\cdot[f_w(S_w^{n+1})(\mathbf{v}^n+\lambda_{nw}\Delta\rho g\nabla z)]+\nabla\cdot(f_w(S_w^{n+1})\lambda_{nw}(S_w^{n+1})\nabla p_c(S_w^{n+1}))=q_w^n, \tag{20}$$

or:

$$\begin{aligned}S_w^{n+1}-S_w^n-\frac{\Delta t}{\phi}\left(q_w-\nabla\cdot[f_w(S_w^{n+1})(\mathbf{v}^n+\lambda_{nw}\Delta\rho g\nabla z)]\right)\\ +\frac{\Delta t}{\phi}\left(\nabla\cdot(f_w(S_w^{n+1})\lambda_{nw}(S_w^{n+1})\nabla p_c(S_w^{n+1}))\right)=0.\end{aligned} \tag{21}$$

If we use the implicit scheme, the resulting system is nonlinear (Equation (21)) and depends on the saturation at time step 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:

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

where

$$\begin{aligned}F(S^k,S^n)=S_w^k-S_w^n-\frac{\Delta t}{\phi}\left(q_w-\nabla\cdot[f_w(S_w^k)(\mathbf{v}^n+\lambda_{nw}\Delta\rho g\nabla z)]\right)\\ +\frac{\Delta t}{\phi}\left(\nabla\cdot(f_w(S_w^k)\lambda_{nw}(S_w^k)\nabla p_c(S_w^k))\right),\end{aligned} \tag{22}$$

$J(S^k)=\frac{\partial F(S^k,S^n)}{\partial S^k}$ is the Jacobian matrix, and δS^{k+1} is the NR update at iteration step $k+1$. Therefore, the linear system to solve is:

$$J(S^k)\delta S^{k+1}=b(S^k). \tag{23}$$

where $b(S^k)$ is the function evaluated at iteration step k, $b(S^k)=-F(S^k,S^n)$, and δS^{k+1} is the unknown. Once we have computed a δS^{k+1} accurate enough, we update the saturation of the actual time step,

$$S_w^{n+1}=S_w^n+\delta S_w^{n+1}.$$

Then, we compute the pressure for this time step p_{nw}^{n+1}, and we repeat the process for the rest of time steps.

4 Solution methods for linear systems

Solving a large pressure linear system is time-consuming. Therefore, iterative techniques are preferred over direct methods to approximate the solution. When the system matrix is SPD (Symmetric Positive Definite), the Conjugate Gradient (CG) is preferred as iterative method. This method can be accelerated with, e.g., the Incomplete Cholesky preconditioner. 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 Method

The CG method is a Krylov-subspace method used when the matrix of the linear system is SPD. 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 $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$ and set $\mathbf{p}^0 = \mathbf{r}^0$.
for $k = 0, ...,$ until convergence
$\alpha^k = \frac{(\mathbf{r}^k, \mathbf{r}^k)}{(\mathbf{A}\mathbf{p}^k, \mathbf{p}^k)}$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
$\beta^k = \frac{(\mathbf{r}^{k+1}, \mathbf{r}^{k+1})}{(\mathbf{r}^k, \mathbf{r}^k)}$
$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta^k \mathbf{p}^k$
end

Preconditioning

To accelerate the convergence of a Krylov-subspace 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 preconditioned system is:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}. \tag{24}$$

Deflation

Sometimes, there are a few extreme eigenvalues hampering the convergence of an iterative method, with deflation [7], the effect of these eigenvalues can be annihilated. Given an SPD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$the deflation matrix $\mathbf{P}$ is defined as follows [20, 22]:

$$\mathbf{P} = \mathbf{I} - \mathbf{A}\mathbf{Q}, \qquad \mathbf{P} \in \mathbb{R}^{n\times n}, \qquad \mathbf{Q} \in \mathbb{R}^{n\times n},$$

where

$$\mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T, \qquad \mathbf{Z} \in \mathbb{R}^{n\times m}, \qquad \mathbf{E} \in \mathbb{R}^{m\times m},$$

with

$$\mathbf{E} = \mathbf{Z}^T\mathbf{A}\mathbf{Z}.$$

$\mathbf{E}$ is known as the (invertible) *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 [8, 29], subdomain deflation vectors [19] or multigrid and multilevel-based deflation matrices [20, 30].

Proper Orthogonal Decomposition (POD)

In this method, a small set of orthonormal basis vectors $\Psi = [\psi_1\ \psi_2\ ..\ \psi_l]$, $\Psi \in \mathbb{R}^{n\times l}$, is 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' $\{\mathbf{x_i}\}_{i=1,..,m}$, obtained by simulation or experiments [4]. These vectors $\{\psi_j\}_{j=1}^l$ are l eigenvectors corresponding to the l largest eigenvalues $\{\sigma_j\}_{j=1}^l$ of the data snapshot correlation matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$,

$$\mathbf{R} := \frac{1}{m}\mathbf{X}\mathbf{X}^T \equiv \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T, \qquad \mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_m]. \tag{25}$$

If the system is large (n), the matrix $\mathbf{R}$ is also large, and to compute the eigenvalues can be costly. However, it is not necessary to compute the eigenvalues from $\mathbf{X}\mathbf{X}^T$, but instead, it is possible to compute the eigenvalues of the much smaller matrix $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{m\times m}$ (see Appendix E).

Once the basis is computed, the high dimensional variable $\mathbf{x} \in \mathbb{R}^n$ is approximated by a linear combination of l orthonormal basis vectors [3]:

$$\mathbf{x} \approx \sum_{i=1}^{l} c_i\psi_i. \tag{26}$$

The l eigenvectors are chosen such that they contain almost all the variability of the snapshots. Usually, they are chosen as the eigenvectors of the maximal number (l) of eigenvalues satisfying [4]:

$$\frac{\sum_{j=1}^{l}\sigma_j}{\sum_{j=1}^{m}\sigma_j} \leq \alpha, \qquad 0 < \alpha \leq 1, \tag{27}$$

with α close to 1. The eigenvalues σ_j are ordered from large to small with σ_1 beeing the largest eigenvalue of $\mathbf{R}$.

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

5 Numerical experiments

In this section we present a series of experiments were we test the deflation method with a POD basis as deflation matrix. We study two-phase flow problems in a highly heterogeneous reservoir. We study 2D and 3D problems. For the 3D case, we include gravity terms. We study water flooding for immiscible fluids (oil and water) and we study cases with and without capillary pressure involved.

Model problems

We model water flooding into a reservoir initially filled with oil. Therefore, the initial saturation is set as one for the oil and zero for the water (see Table 2).

We study two models with different permeability fields; an academic layered reservoir with a contrast in permeability coefficients up to 10^6, and the SPE 10 benchmark [31] with a contrast of 10^7.

We study the solution of systems of linear equations for the pressure, resulting from the discretization of the partial differential equations describing this process. We use the fractional flow formulation to decouple pressure from saturation and we the resulting system with sequential schemes. The pressure linear system is obtained with MRST. The transport equation is solved with MRST using implicit schemes.

Pressure solver

As mentioned before, we focus on the solution of the pressure Equation (12). We implement the Deflated Preconditioned Conjugate Gradient method, preconditioned with Incomplete Cholesky (DICCG), using a POD basis as deflation vectors. We compare the results with the non-deflated method (ICCG). Defining N as the number of grid cells, d the number of deflation vectors, and s is the number of snapshots, the computational cost of computing a solution with the ICCG method is $31N$ for a 2D case and $39N$ for 3D. For the DICCG method, the cost is $(31+4d)N$ for 2D and $(39+4d)N$ for 3D. For the DICCG method, we also need to perform the SVD decomposition that costs Ns^2 operations.

Deflation procedures

To construct the deflation vectors, we study different approaches. We study a moving window and a training run approach to obtain a POD basis that is used as deflation vectors.

Moving window: With this approach, we start by computing a set of s snapshots and obtaining a POD basis from it. We solve the rest of the time steps using the DICCG method with the vectors of the POD basis as deflation vectors. The basis and, as a consequence, the deflation matrix have to be updated 'on-the-fly' at each time step. The pseudo code is given in Algorithm 3.

We compare the total number of iterations necessary to run the whole DICCG simulation, where the first s time steps are computed with ICCG, with the total number of

iterations necessary to solve the same problem using the ICCG method only.

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

$X_{1:s} = \{x_1, x_2, ..., x_s\}$[a] % Compute the solution of the first s time steps with **ICCG**.

for $t = s+1, \quad ..., \quad$ steps

$\mathbf{R} = \frac{1}{s}\mathbf{X}^T_{(t-s):(t-1)}\mathbf{X}_{(t-s):(t-1)}$ % Construct correlation matrix with the previous s solutions.

$\mathbf{\Sigma}_{1:s} = \{\sigma_1, ..., \sigma_s\}$ % Compute the eigenvalues and

$\mathbf{\Phi}_{1:s} = \{\phi_1, ..., \phi_s\}$ eigenvectors of $\mathbf{R}$[b].

$\mathbf{Z}_{1:l} = \{\phi_1, ..., \phi_l\}$ % Construct the deflation matrix with the POD basis.

$\mathbf{x}_t$ % Compute the next solution with **DICCG**.

end

[a] We define $X_{a:b} := \{x_a, x_{a+1}, ..., x_b\}$.

[b] The POD basis is constructed with the l largest eigenvectors.

Training simulation: For this case, we use a *training phase*, where we run the simulation for all the time steps with the ICCG method. During this *training phase*, we randomly vary the pressure in the production wells. A POD basis is computed from the solutions of the *training phase* and it is used to construct a deflation matrix with 10 or 30 POD basis vectors as deflation vectors. We solve a series of problems with the same conditions as the *training phase*, but with different pressures in the wells, i.e., different *rhs*. The pseudo code is presented in Algorithm 4.

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.

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

Solve $\mathbf{A}_t\mathbf{x}*_t = \mathbf{b}_t$	% Run a training phase with **ICCG** varying the pressure in the wells.
$X_{1:steps} = \{x*_1, x*_2, ..., x*_{steps}\}$	% Save the solutions as a matrix.
$\mathbf{R} = \frac{1}{steps}\mathbf{X}^T_{1:steps}\mathbf{X}_{1:steps}$	% Construct correlation matrix with the solutions.
$\mathbf{\Sigma}_{1:steps} = \{\sigma_1, ..., \sigma_{steps}\}$ $\mathbf{\Phi}_{1:steps} = \{\phi_1, ..., \phi_{steps}\}$	% Compute the eigenvalues and eigenvectors of $\mathbf{R}$.
$\mathbf{Z}_{1:l} = \{\phi_1, ..., \phi_l\}$	% Construct the deflation matrix with the POD basis.
Solve $\mathbf{A}_t\mathbf{x}_t = \mathbf{b}_t$	% The deflation matrix can be used to solve similar problems using the **DICCG** method.

5.1 Heterogeneous permeability layers

The experiments simulate flow through a porous medium with a constant porosity field of 0.2. In this set of experiments, we study an academic system, which consists of equal-sized layers with different permeability values (see Figure 1). The configuration of these layers consists of a layer with one value of permeability κ_1, followed by a layer with a different permeability value κ_2. The permeability of one set of layers is set to $\kappa_1 = 1mD$, the permeability of the other set, κ_2, is varied. Therefore, the contrast in permeability between the layers ($\frac{\kappa_2}{\kappa_1} = \kappa_2$), depends on the value of κ_2. The permeability κ_2 varies from $\kappa_2 = 1mD$ up to $\kappa_2 = 10^6 mD$. The domain consists of a Cartesian grid of 32 x 32 cells, one meter long in the 2D case, and 24 x 24 x 24 cells for the 3D case. For the relative permeability of the fluids, the Corey model is used. The properties of the fluids are presented in Table 1. No gravity terms and no capillary pressure are taken into account in the first set of experiments.

We repeat the experiments, taking into account capillary pressure. We use a linear capillary relationship, $P_c = C(1 - S)$; the curve is presented in Figure 3. The last set of experiments with the layered system consisting of a 3D problem, where gravity terms are included. We study water flooding, with water injected from the boundary and from wells. 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 3). For the injection through wells, we have two cases. In the first one, we use two wells, one injector (I) and one producer (P) placed on opposite corners of the reservoir. For the second, we place four producers (P_i) on the corners and one injector in the center. The wells are controlled prescribing the *bhp* (see Table 14). The simulation is run during 4800 days with

240 time steps of 20 days (See Table 3). The stopping criterium for the ICCG and DICCG methods is $\epsilon = 1 \cdot 10^{-11}$.

Property	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

Table 1: Fluids properties.

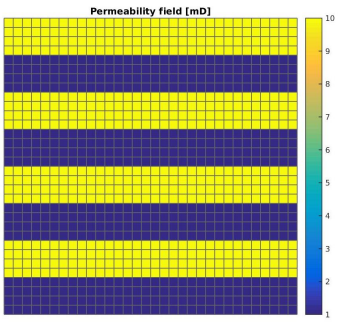

Figure 1: Rock permeability

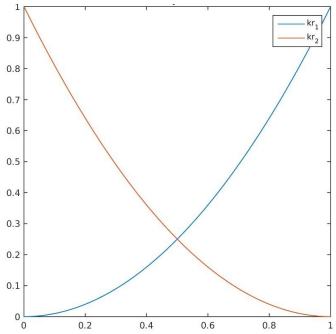

Figure 2: Fluid relative permeability

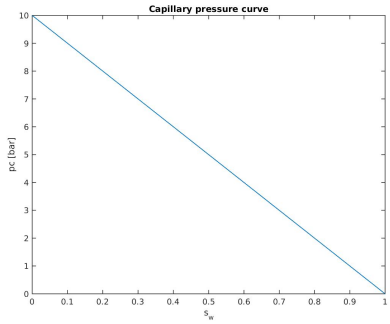

Figure 3: Capillary pressure

5.1.1 Injection through the left boundary

A: No capillary pressure.

Initial saturations		
	Water	Oil
$S_{0,x\neq 0,L_x}$	0	1
$S_{x=0}$	1	0
$S_{x=L_x}$	0	1

Table 2: Initial Saturations.

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

Table 3: Boundary conditions and temporal parameters.

As mentioned before, we simulate flow through a porous medium with water injection through the left boundary in a homogeneous and a layered reservoir. Results are presented

in Table 4. This first column contains the contrast between permeability layers ($\frac{\kappa_1}{\kappa_2}$). The number of iterations necessary to achieve convergence with the ICCG method is presented in the second column (Total ICCG Iterations). The third column shows the number of deflation vectors used (5 or 10 in this case). The number of iterations necessary to compute the snapshots with the ICCG method is presented in the fourth column (ICCG Iterations). In the fifth column, we give the total number of iterations, taking into account the snapshots computed with ICCG and the rest of the iterations computed with DICCG. In the last column, we compute the total number of iterations of the DICCG methods with respect to ICCG.

We observe (see Table 4) that using deflation methods reduces the number of iterations to around 7% of the total ICCG iterations. We also note that the number of iterations does not change dramatically when we vary the contrast between permeability layers or we change the number of deflation vectors. The largest increment in iterations occurs when we have a contrast of 10^6. Comparing this case with the homogeneous case, we see an increase of 10% in the number of iterations, which is a small increment.

For a contrast between the permeability layers of 10^1 or 10^6 we observe five eigenvalues significantly larger than the rest (see Figure 6). Therefore, if we use five POD vectors instead of ten as deflation vectors the results are similar, which is shown in Table 4. For the case of higher contrast, the spectrum is more spread. This could explain the slight increase in the number of iterations when we increase the contrast.

Pressure field and the water saturation appear in Figure 4 and Figure 5 for various times. The pressure value is higher on the boundary where water is injected decreasing towards the right boundary, and it flows easily through the layers with higher permeability (see Figure 5).

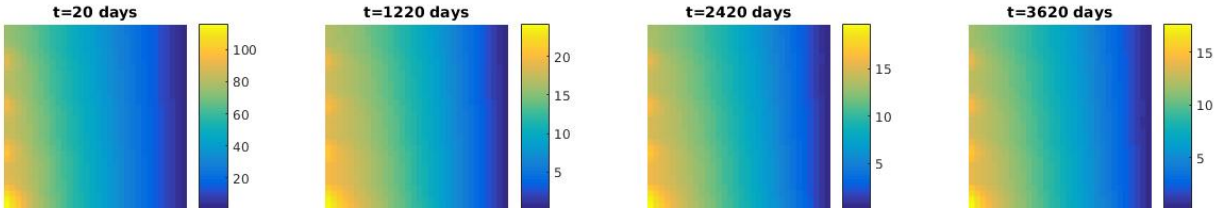

Figure 4: Pressure field [bars] for various times, for a contrast between permeability values of 10^1, 32 x 32 grid cells.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	12210	$DICCG_{10}$	495	295	790	6
10^0	12210	$DICCG_5$	495	384	879	7
10^1	14783	$DICCG_{10}$	605	1270	1875	13
10^1	14783	$DICCG_5$	605	1573	2178	15
10^2	14513	$DICCG_{10}$	624	764	1388	10
10^2	14513	$DICCG_5$	624	919	1543	11
10^3	12714	$DICCG_{10}$	524	700	1224	10
10^3	12714	$DICCG_5$	524	923	1447	11
10^4	11151	$DICCG_{10}$	482	783	1265	11
10^4	11151	$DICCG_5$	482	960	1442	13
10^5	10958	$DICCG_{10}$	469	982	1451	13
10^5	10958	$DICCG_5$	469	1078	1547	14
10^6	9735	$DICCG_{10}$	442	1163	1605	16
10^6	9735	$DICCG_5$	442	1317	1759	18

Table 4: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 32 x 32 cells.

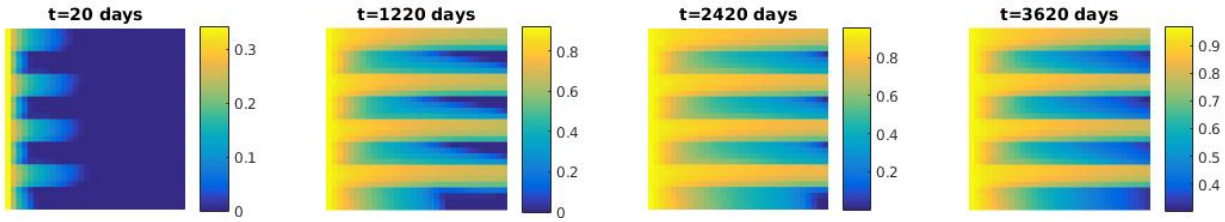

Figure 5: Water saturation for various times, for a contrast between permeability values of 10^1, 32 x 32 grid cells.

B: Capillary pressure included

For this set of experiments, we include capillary pressure. The capillary pressure function used for these experiments is presented in Table 1 and Figure 3. The number of iterations necessary to achieve convergence for various contrast between permeability layers is presented in 5. The pressure field and the water saturation are presented in Figure 7 and Figure 8 for various times.

In Table 5 we observe that the behavior of the DICCG method is similar when we use 5 or 10 POD basis vectors as deflation vectors. The performance of the DICCG method is better without capillary pressure, previous case. However, we observe that for a contrast between permeability layers less than 10^4 we need around 20% ICCG iterations. For a

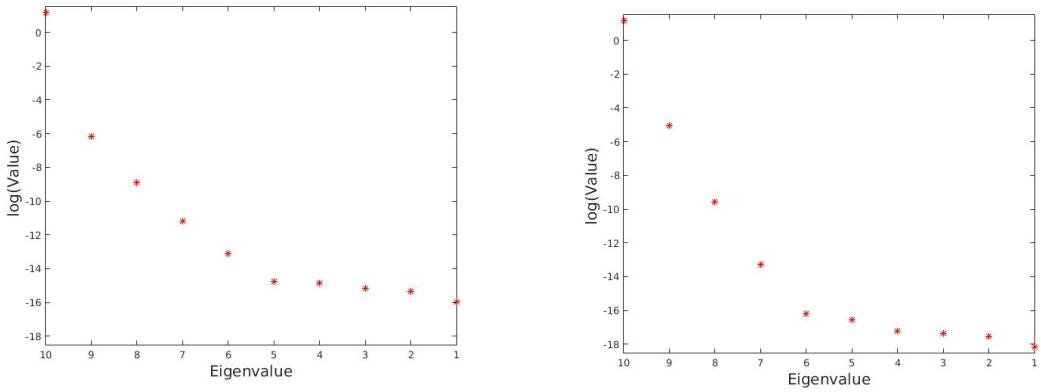

Figure 6: Eigenvalues of the correlation matrix $\mathbf{R} = \frac{1}{m}\mathbf{X}\mathbf{X}^T$ for a contrast between permeability values of 10^1 and 10^6.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	12440	DICCG_{10}	500	2050	2550	20
10^0	12440	DICCG_5	500	1951	2451	20
10^1	14597	DICCG_{10}	600	2843	3443	24
10^1	14597	DICCG_5	600	3072	3672	25
10^2	14897	DICCG_{10}	618	2517	3135	21
10^2	14897	DICCG_5	618	2588	3206	22
10^3	11821	DICCG_{10}	502	2439	2941	25
10^3	11821	DICCG_5	502	2362	2864	24
10^4	10530	DICCG_{10}	465	2491	2956	28
10^4	10530	DICCG_5	465	2464	2929	28
10^5	10030	DICCG_{10}	451	2952	3403	34
10^5	10030	DICCG_5	451	2770	3221	32
10^6	9071	DICCG_{10}	428	3156	3584	40
10^6	9071	DICCG_5	428	2644	3072	34

Table 5: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 32 x 32 cells, capillary pressure included.

larger contrast, we need between 30 to 40% ICCG iterations, which is a good reduction.

We note that the eigenvalues of the correlation matrix are more spread when the contrast between permeability layers is 10^6 (see Figure 9). The maximum is similar for both cases, but the minimum for a contrast of 10^1 is around 10^{-14} and for a contrast of 10^6 is around 10^{-16}, which is two orders of magnitude difference. The latter can result in a better

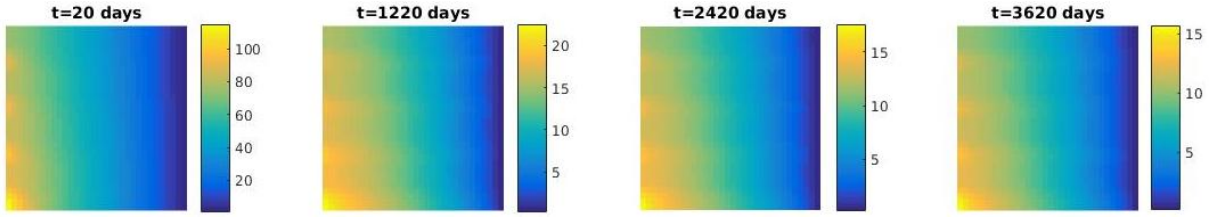

Figure 7: Pressure field [bars] for various times for a contrast between permeability values of 10^1, 32 x 32 grid cells.

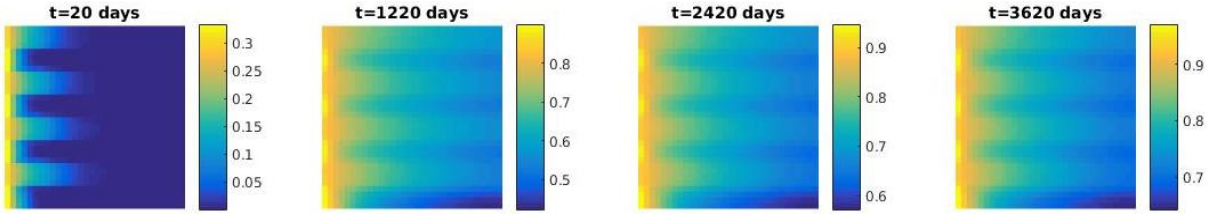

Figure 8: Water saturation for various times for a contrast between permeability values of 10^1, 32 x 32 grid cells.

performance of the DICCG method for the case with smaller contrast.

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) and we use 10 and 5 deflation vectors.

Case 1

The required number of iterations are presented in Table 6 for the first case. From this table, we note that the performance of the DICCG method is similar to the case of 10 deflation vectors. Therefore, we cannot conclude that the number of deflation vectors influences the behavior of the method, when we include capillary pressure. We observe

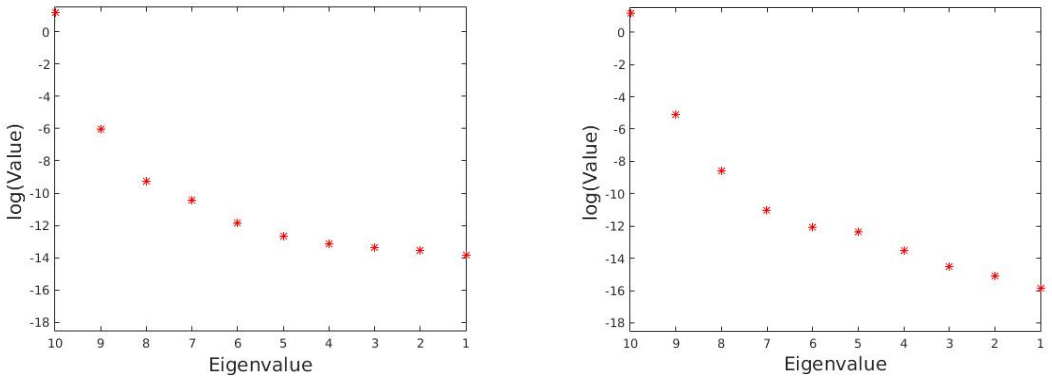

Figure 9: Eigenvalues of the correlation matrix $\mathbf{R} = \frac{1}{m}\mathbf{X}\mathbf{X}^T$ for a contrast between permeability values of 10^1 and 10^6, capillary pressure included.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	12440	$\text{DICCG}_{POD_{20}}$	1010	1815	2825	23
10^0	12440	$\text{DICCG}_{POD_{11}}$	1010	1497	2507	20
10^1	14597	$\text{DICCG}_{POD_{20}}$	1210	2347	3557	24
10^1	14597	$\text{DICCG}_{POD_{11}}$	1210	2486	3696	25
10^2	14897	$\text{DICCG}_{POD_{20}}$	1248	2136	3384	23
10^2	14897	$\text{DICCG}_{POD_{11}}$	1248	2233	3481	23
10^3	11821	$\text{DICCG}_{POD_{20}}$	1002	2082	3084	26
10^3	11821	$\text{DICCG}_{POD_{11}}$	1002	2170	3172	27
10^4	10530	$\text{DICCG}_{POD_{20}}$	954	2272	3226	31
10^4	10530	$\text{DICCG}_{POD_{11}}$	954	2312	3266	31
10^5	10030	$\text{DICCG}_{POD_{20}}$	928	2761	3689	37
10^5	10030	$\text{DICCG}_{POD_{11}}$	928	2875	3803	38
10^6	9071	$\text{DICCG}_{POD_{20}}$	837	3229	4066	45
10^6	9071	$\text{DICCG}_{POD_{11}}$	837	2927	3764	41

Table 6: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 32 x 32 cells, capillary pressure included, 20 deflation vectors.

that the range of the spectrum of the correlation matrix for this case is similar to the case when we have 10 deflation vectors (see Figure 10), which can be the cause of the similar behavior.

Case 2

For this case, we use 480 time steps, instead of the 240 of the previous cases, which implies that the change in the matrix is less than in the previous case. In Table 7 we present the number of iterations of the ICCG and DICCG methods. We observe that the DICCG method performs better if we have smaller time step, i.e., less change in the $\mathbf{A}^m$ matrix. In Figure 11, we note that the smallest eigenvalue is 10^{-15} for a contrast of 10^1 and 10^{-17} for a contrast of 10^6. 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 appear to lead to a better performance.

From these experiments, we observe that the performance of the DICCG method depends on the time step. 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, for the case when we have capillary pressure involved, it is necessary to take into account smaller changes produced in the system, which can be done by taking smaller time steps.

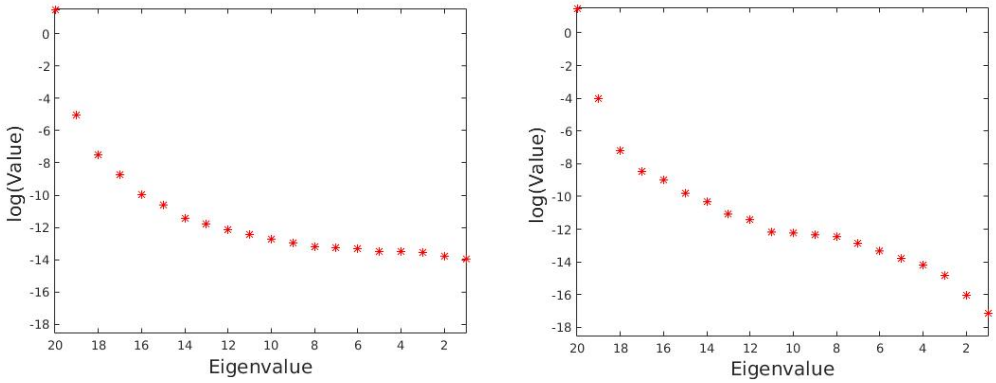

Figure 10: Eigenvalues of the correlation matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$ for a contrast between permeability values of 10^1 and 10^6, capillary pressure included, 20 deflation vectors.

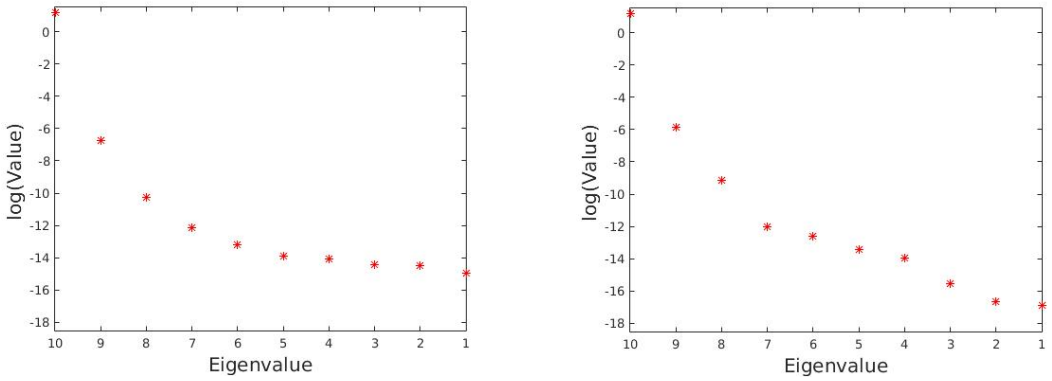

Figure 11: Eigenvalues of the correlation matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$ for a contrast between permeability values of 10^1 and 10^6, 480 time steps.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	24882	$DICCG_{10}$	495	3362	3857	16
10^0	24882	$DICCG_5$	495	3324	3819	15
10^1	29187	$DICCG_{10}$	585	4166	4751	16
10^1	29187	$DICCG_5$	585	4463	5048	17
10^2	29795	$DICCG_{10}$	617	3598	4215	14
10^2	29795	$DICCG_5$	617	3777	4394	15
10^3	23617	$DICCG_{10}$	513	3434	3947	17
10^3	23617	$DICCG_5$	513	3445	3958	17
10^5	20047	$DICCG_{10}$	413	4230	4643	23
10^5	20047	$DICCG_5$	413	4000	4413	22
10^6	18400	$DICCG_{10}$	393	4623	5016	27
10^6	18400	$DICCG_5$	393	4005	4398	24

Table 7: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 32 x 32 cells, capillary pressure included (480 time steps).

C: No capillary pressure, gravity included (3D problem)

In this section, we repeat the experiments performed in the 2D case for a 3D problem with 24 x 24 x 24 cells. As in the previous case, each cell is one meter long. We studied two layered problems. In the first one, the layers are placed vertically (see Figure 12) and in the second, an horizontal configuration is used (see Figure 13). The time step parameters are the same as in the previous studies (see Table 3).
In Tables 8 and 9, the number of iterations necessary to achieve convergence are presented for various contrast between permeability layers for the ICCG and DICCG methods. The pressure field and the water saturation are presented in Figures 14 and 16 for various times for the first case, and Figures 18 and 19 for the second.

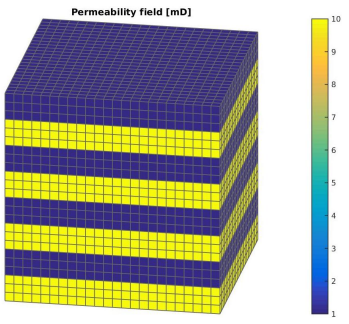

Figure 12: Rock permeability

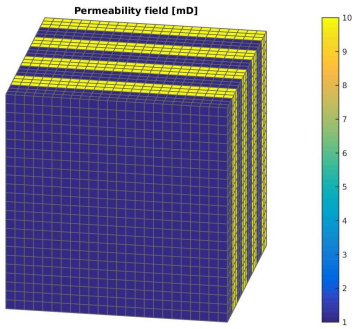

Figure 13: Rock permeability

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	15438	$DICCG_{10}$	574	1616	2190	14
10^0	15438	$DICCG_5$	574	1871	2445	16
10^1	17496	$DICCG_{10}$	656	1800	2456	14
10^1	17496	$DICCG_5$	656	2170	2826	16
10^2	20587	$DICCG_{10}$	791	2023	2814	14
10^2	20587	$DICCG_5$	791	2251	3042	15
10^3	20044	$DICCG_{10}$	602	1888	2490	12
10^3	20044	$DICCG_5$	602	2236	2838	14
10^4	17563	$DICCG_{10}$	530	1782	2312	13
10^4	17563	$DICCG_5$	530	2140	2670	15
10^5	16944	$DICCG_{10}$	513	1874	2387	14
10^5	16944	$DICCG_5$	513	2124	2637	16
10^6	15720	$DICCG_{10}$	486	1972	2458	16
10^6	15720	$DICCG_5$	486	2121	2607	17

Table 8: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 24 x 24 x 24 cells, no capillary pressure, gravity included.

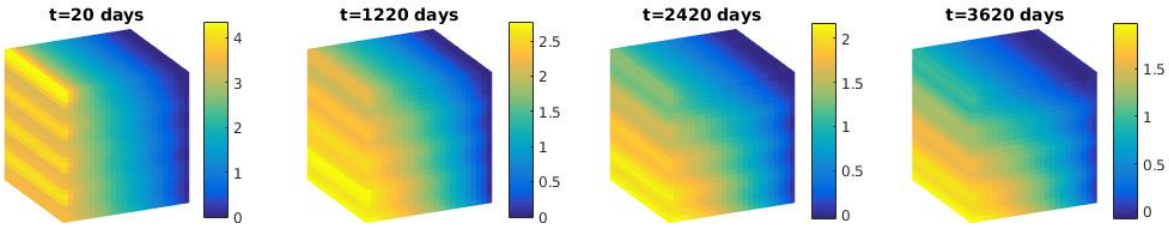

Figure 14: Pressure field for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, horizontal layers.

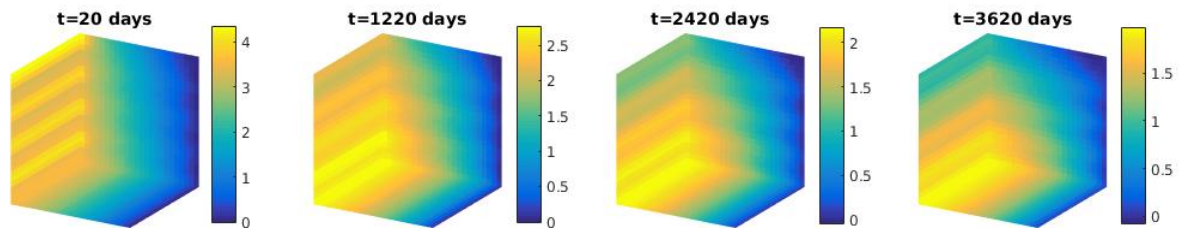

Figure 15: Pressure field for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, horizontal layers, lower view.

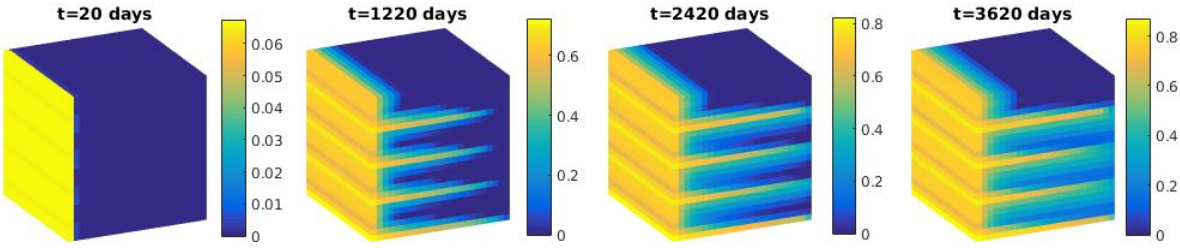

Figure 16: Water saturation for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, horizontal layers.

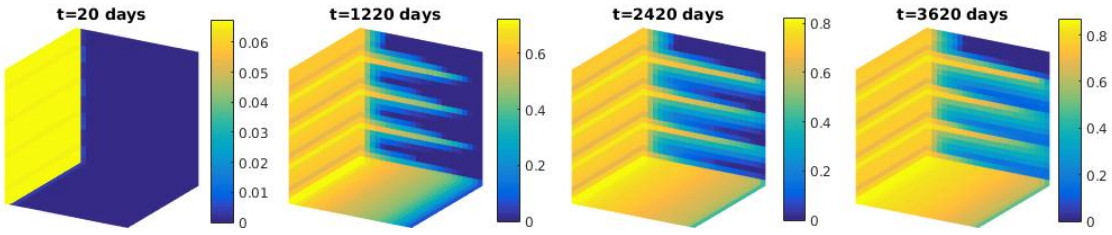

Figure 17: Water saturation for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, horizontal layers, lower view.

In Tables 8 and 9, the number of DICCG iterations is reduced to around 15% ICCG iterations. We note that the results are similar while using ten or five deflation independent of the contrast in permeability or positioning of the layers.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	15438	$DICCG_{10}$	574	1616	2190	14
10^0	15438	$DICCG_5$	574	1871	2445	16
10^1	19426	$DICCG_{10}$	654	1969	2623	14
10^1	19426	$DICCG_5$	654	2258	2912	15
10^2	22577	$DICCG_{10}$	762	2340	3102	14
10^2	22577	$DICCG_5$	762	2714	3476	15
10^3	21832	$DICCG_{10}$	594	2086	2680	12
10^3	21832	$DICCG_5$	594	2406	3000	14
10^4	18483	$DICCG_{10}$	529	1868	2397	13
10^4	18483	$DICCG_5$	529	2121	2650	14
10^5	17808	$DICCG_{10}$	513	1819	2332	13
10^5	17808	$DICCG_5$	513	2065	2578	14
10^6	17152	$DICCG_{10}$	486	1955	2441	14
10^6	17152	$DICCG_5$	486	2115	2601	15

Table 9: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 24 x 24 x 24 cells, no capillary pressure, gravity included, vertical layers.

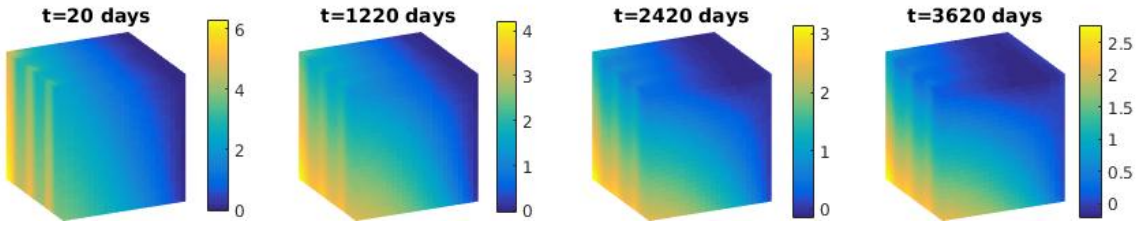

Figure 18: Pressure field for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, vertical layers.

D: Capillary pressure and gravity included (3D problem).

As in the previous case, we study a 3D problem with gravity terms included, but in this case, we also include capillary pressure. The capillary pressure function is the same as in the 2D problem (see Table 1 and Figure 3). The number of iterations necessary to achieve convergence is presented for various contrast between permeability layers for the ICCG and DICCG methods in Table 10 . The pressure field and the water saturation are presented in Figures 20 and 21 for various times.

In Table 10 and Table 11, we observe that for the DICCG method we need around 20% of the ICCG iterations. This percentage increases slightly when we increase the contrast

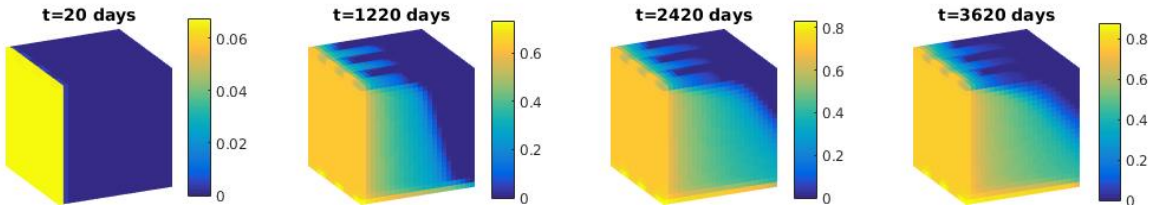

Figure 19: Water saturation for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, vertical layers.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	14136	DICCG_{10}	588	2357	2945	21
10^0	14136	DICCG_5	588	2520	3108	22
10^1	17224	DICCG_{10}	660	3431	4091	24
10^1	17224	DICCG_5	660	3658	4318	25
10^2	20562	DICCG_{10}	763	3468	4231	21
10^2	20562	DICCG_5	763	3596	4359	21
10^3	18514	DICCG_{10}	605	2894	3499	19
10^3	18514	DICCG_5	605	2943	3548	19
10^4	15397	DICCG_{10}	520	2801	3321	22
10^4	15397	DICCG_5	520	2925	3445	22
10^5	14955	DICCG_{10}	505	3025	3530	24
10^5	14955	DICCG_5	505	3206	3711	25
10^6	13228	DICCG_{10}	469	3531	4000	30
10^6	13228	DICCG_5	469	2992	3461	26

Table 10: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 24 x 24 x 24 cells, capillary pressure and gravity included, vertical layers.

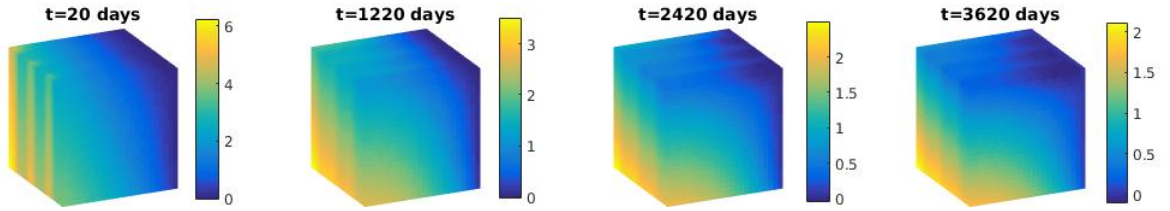

Figure 20: Pressure field for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, capillary pressure and gravity included.

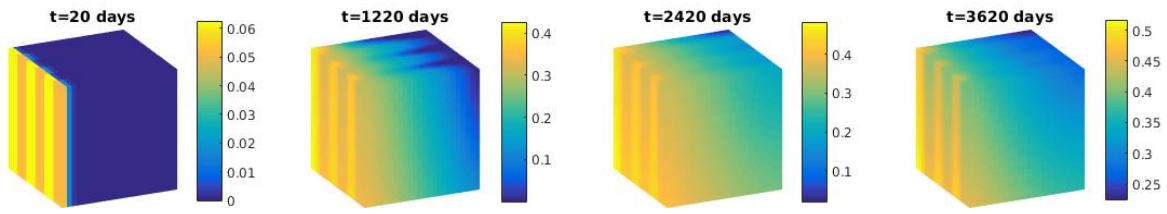

Figure 21: Water saturation for various times for a contrast between permeability values of 10^1, 24 x 24 x 24 grid cells, capillary pressure and gravity included.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	14136	$DICCG_{10}$	588	2357	2945	21
10^0	14136	$DICCG_5$	588	2520	3108	22
10^1	16614	$DICCG_{10}$	672	2786	3458	21
10^1	16614	$DICCG_5$	672	3130	3802	23
10^2	20037	$DICCG_{10}$	786	3290	4076	20
10^2	20037	$DICCG_5$	786	3773	4559	23
10^3	18640	$DICCG_{10}$	610	2918	3528	19
10^3	18640	$DICCG_5$	610	3265	3875	21
10^4	15437	$DICCG_{10}$	522	2712	3234	21
10^4	15437	$DICCG_5$	522	3107	3629	24
10^5	14371	$DICCG_{10}$	505	2899	3404	24
10^5	14371	$DICCG_5$	505	3273	3778	26
10^6	13176	$DICCG_{10}$	470	3361	3831	29
10^6	13176	$DICCG_5$	470	2940	3410	26

Table 11: Number of iterations for various contrast between permeability layers. Injection through the left boundary, domain 24 x 24 x 24 cells, capillary pressure and gravity included, horizontal layers.

between permeability layers. Comparing the cases with and without capillary pressure, we observe that the performance is better without. As mentioned for the 2D case, this could be caused by the lack of information obtained with the deflation vectors. The performance can be improved by decreasing the time step.

5.1.2 Injection through wells.

We study water flooding with injection through wells. For the first set of experiments, two wells are placed in the reservoir, one injection (I) and one production well (P). In the second set, we use five wells in the reservoir, four producers (P_i) and one injector (I). The wells are controlled changing the *bhp*. The pressure in the wells is presented for each experiment. We impose homogeneous Neumann boundary conditions (no flux). We study a problem with layered permeability values and the SPE 10 benchmark with different number of layers. The characteristics of the fluids are the same as in the previous case (see 1). No gravity terms and no capillary pressure are taken into account in the first set of experiments. In the second part of this section, we study 3D problem with gravity and we also include capillary pressure.

A: Heterogeneous permeability layers, two wells, *bhp* controlled

In this section, we study water flooding with two wells. Water is injected in one well placed at one corner (I) and produced in a well located at the opposite corner (P). The domain consist layers with different permeability and 35 x 35 cells (see Figure 22). The contrast between permeability layers varies from 10^0 to 10^3. The pressure in the injector is 200 bars and in the producer is -200 bars. As in the previous experiments, we study deflation method with five and ten POD basis vectors as deflation vectors. The number of iterations necessary to achieve convergence are presented in Table 17.

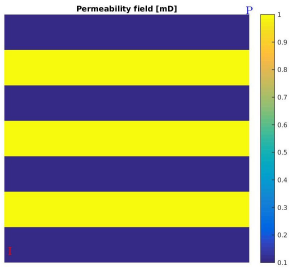

Figure 22: Rock permeability

The total time of the simulation is the time when a volume of water corresponding to 1.2 of the pore volume of the reservoiris injected. We simulate 480 time steps. The oil flux rate is presented in figures 23 and 29 for the homogeneous problem and layers with a contrast of 10^1 and 10^3 in the permeability layers. We are injecting water through the injector, therefore we don't have oil production in the injection well (I). For the production well (P), in both cases, we observe a decrease in the oil rate after $\sim 1.5 \times 10^5$ days for the first case, and $\sim 1.5 \times 10^7$ days for the second case. If we observe the water flux rate for the same cases (Figures 30 and 32), we note a decrease in the injector well when we have an increase in the production well. This shows that at this moment, the water has reached the production well; therefore, we are producing oil and water. Thus, the oil production decreases. We also note that the rate decreases almost linearly with the contrast between permeability layers.

The deflated method reduces the number of ICCG iterations to around 17% of the total, in the homogeneous problem (see Table 17). For the heterogeneous problems, the reduction ais around 13%. This result is independent on the contrast in permeability layers.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	18888	$DICCG_{10}$	395	2739	3134	17
10^0	18888	$DICCG_5$	395	3705	4100	22
10^1	28481	$DICCG_{10}$	595	3198	3793	13
10^1	28481	$DICCG_5$	595	4093	4688	16
10^2	32412	$DICCG_{10}$	681	3385	4066	13
10^2	32412	$DICCG_5$	681	4114	4795	15
10^3	34911	$DICCG_{10}$	730	4875	5605	16
10^3	34911	$DICCG_5$	730	4973	5703	16

Table 12: Number of iterations for various contrast between permeability layers.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	18916	$DICCG_{10}$	395	3524	3919	21
10^0	18916	$DICCG_5$	395	3933	4328	23
10^1	28463	$DICCG_5$	595	4258	4853	17
10^1	28463	$DICCG_{10}$	595	3913	4508	16
10^2	32635	$DICCG_{10}$	683	4662	5345	16
10^2	32635	$DICCG_5$	683	4956	5639	17
10^3	34911	$DICCG_{10}$	730	4875	5605	16
10^3	34911	$DICCG_5$	730	4973	5703	16

Table 13: Number of iterations for various contrast between permeability layers, capillary pressure included.

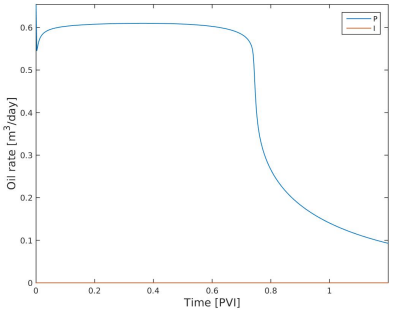

Figure 23: Oil Rate, homogeneous problem.

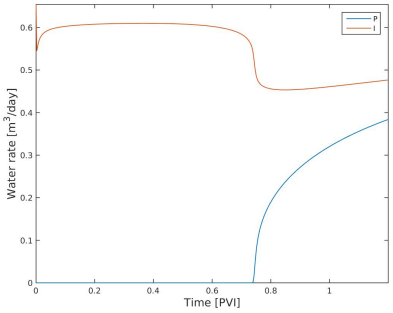

Figure 24: Water Rate, homogeneous problem.

B: Heterogeneous permeability layers, five wells, *bhp* controlled

For the first set of experiments, we study a layered system (see Figure 1). We use five layers of the same size, 3 layers with one value of permeability κ_1, followed by a layer with a different permeability value κ_2. The permeability of one set of layers is $\kappa_1 = 1mD$, the permeability of the other set (κ_2) is varied. Therefore, the contrast in permeability between the layers ($\frac{\kappa_2}{\kappa_1} = \kappa_2$), depends on the value of κ_2. The permeability κ_2 varies from $\kappa_2 = 1mD$ to $\kappa_2 = 10^{-3}mD$. The domain consists of a Cartesian grid of 35 x 35 cells with a length of ten meters per cell.

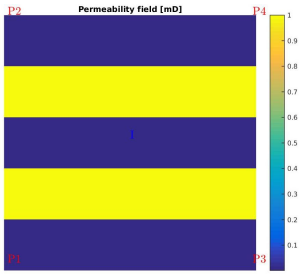

Figure 33: Rock permeability

One injector is placed in the center and four producers on the corners of the reservoir (see Figure 33). The wells are controlled via the bottom hole pressure (*bhp*). Water is injected into a reservoir initially filled with oil. The values of the wells are presented in Table 14. The first simulation has constant permeability through all the reservoir. The

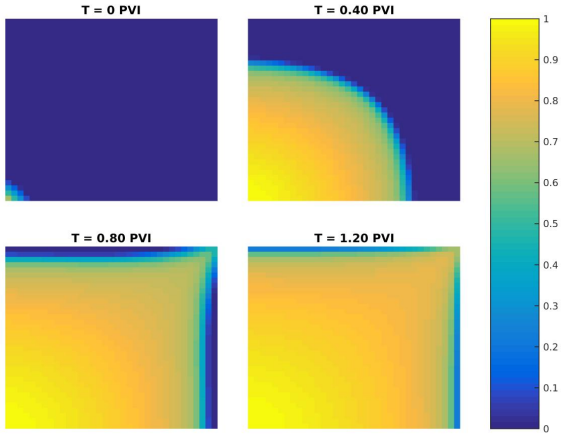

Figure 25: Water saturation, homogeneous problem.

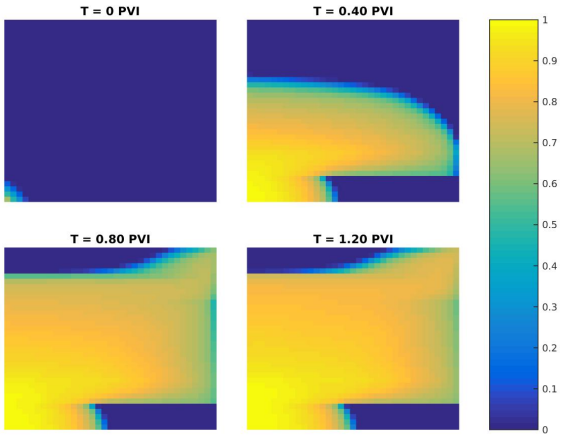

Figure 26: Water saturation, heterogeneous problem, contrast between permeability layers 10^3.

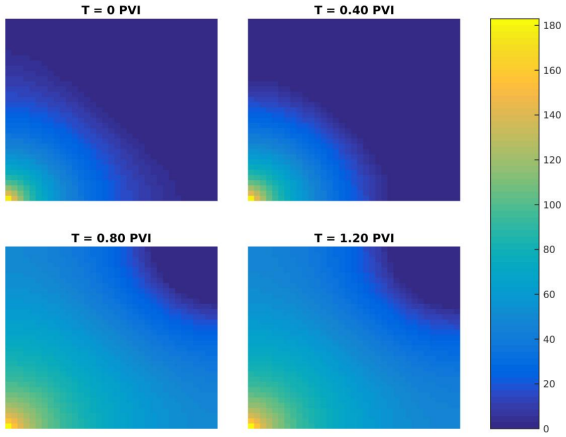

Figure 27: Pressure field, homogeneous problem.

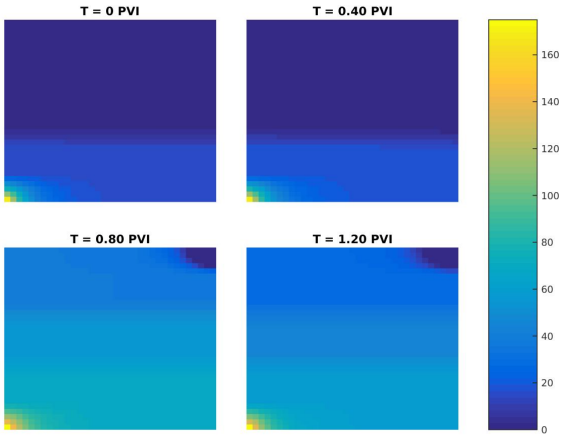

Figure 28: Pressure field, heterogeneous problem, contrast between permeability layers 10^3

simulation is run until we inject 1.2 times the pore volume of the reservoir. We use 480 time steps (see Table 15).

The number of iterations necessary to achieve convergence for the ICCG and DICCG methods are presented in Table 16 for a problem without capillary pressure and Table 17 for a problem including capillary pressure.

In Table 17, we observe that the percentage of DICCG iterations necessary to achieve convergence is reduced to 9% ICCG iterations when using ten and five POD basis vectors as deflation vectors.

We observe, in both cases, that five eigenvalues are larger than the rest, which implies that most of the system's information is contained in these eigenvalues. We observe a

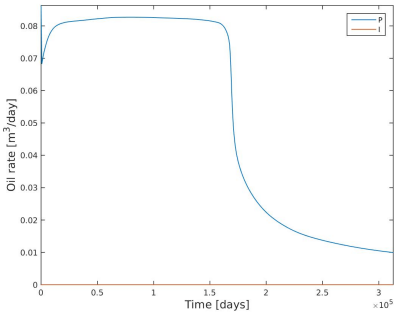

Figure 29: Oil Rate, contrast between permeability layers of 10^1.

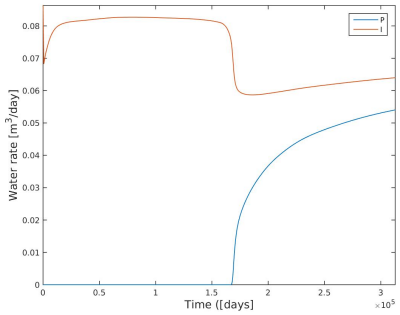

Figure 30: Water Rate, contrast between permeability layers of 10^1.

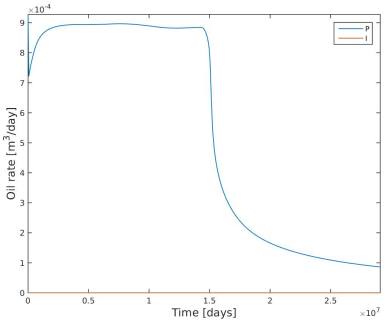

Figure 31: Oil Rate, contrast between permeability layers of 10^3.

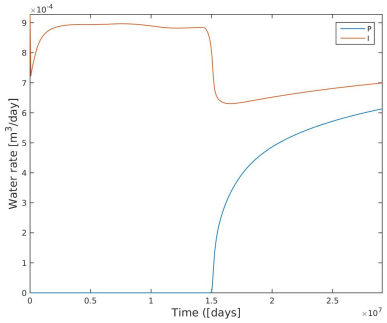

Figure 32: Water Rate, contrast between permeability layers of 10^3.

Well	Water Sat	Oil Sat	Pressure [bars]
P_1	0	1	-50
P_2	0	1	-50
P_3	0	1	-50
P_4	0	1	-50
I	1	0	200

Table 14: Wells properties.

Property	Value
T_{total}	1.2 PV
Steps	480

Table 15: Time.

similar behavior for the DICCG method with five and ten deflation vectors (see Table 16 and Table 17). The correlation matrix shows that the main information is contained in the five largest eigenvalues, which corresponds to the observed behavior of the method (see Figure 34).

Results homogeneous permeability

As mentioned before, we vary the contrast between the permeability layers. The first case that we study the case with homogeneous permeability, i.e. the layers have the same value.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	17506	$DICCG_{10}$	351	1824	2175	12
10^0	17506	$DICCG_5$	351	2312	2663	15
10^1	24394	$DICCG_{10}$	502	1869	2371	10
10^1	24394	$DICCG_5$	502	2477	2979	12
10^2	27364	$DICCG_{10}$	551	1906	2457	9
10^2	27364	$DICCG_5$	551	2583	3134	11
10^3	27092	$DICCG_{10}$	529	2033	2562	9
10^3	27092	$DICCG_5$	529	2430	2959	11

Table 16: Number of iterations for various contrast between permeability layers, five wells.

$\frac{\kappa_2}{\kappa_1}$	Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
10^0	17383	$DICCG_{10}$	351	2655	3006	17
10^0	17383	$DICCG_5$	351	2642	2993	17
10^1	23810	$DICCG_{10}$	502	2641	3143	13
10^1	23810	$DICCG_5$	502	2683	3185	13
10^2	27629	$DICCG_{10}$	551	2719	3270	12
10^2	27629	$DICCG_5$	551	2793	3344	12
10^3	23962	$DICCG_{10}$	517	2872	3389	14
10^3	23962	$DICCG_5$	517	2744	3261	14

Table 17: Number of iterations for various contrast between permeability layers, capillary pressure included, five wells.

In Figure 35 we present the flux rate of water in each well for the homogeneous problem. We observe that the injector (I) starts with a high rate, this is maintained for some time until the water reaches the producers (P), then it decreases. In Figure 36 we observe the saturation of water inside the reservoir. In this case, as the reservoir has constant permeability, we observe a symmetric expansion of water in the reservoir. In Figure 37 we have the oil's flux rate in the wells. This rate is constant until water break through (wbt), when it starts to decrease. We also note, in Figure 38, that the pressure increases from the injector, the center of the domain to the corners with time.

Results heterogeneous permeability

In this section the contrast between permeability layers is of 10^1 and 10^3. We present the rate of water and oil injected or extracted from each well in Figures 39, 41, 43 and 45. As in the previous case, we observe that the injector starts with a high rate, this is

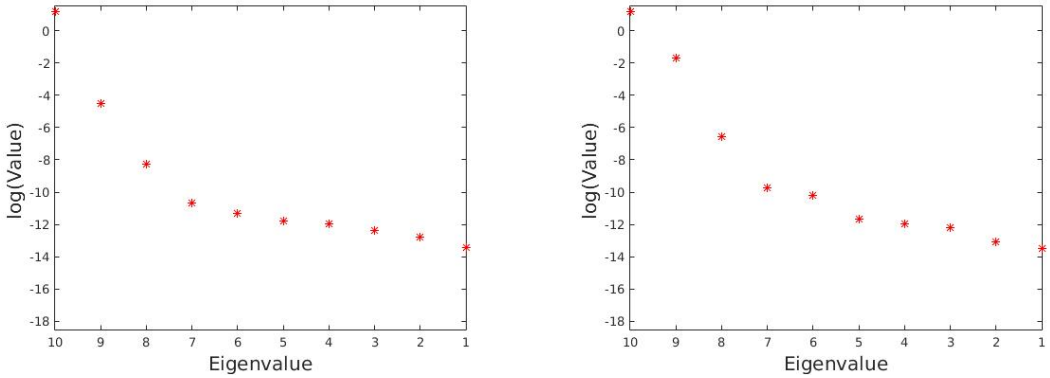

Figure 34: Eigenvalues of the correlation matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$ for a homogeneous problem and a contrast between permeability values of 10^3, 480 time steps.

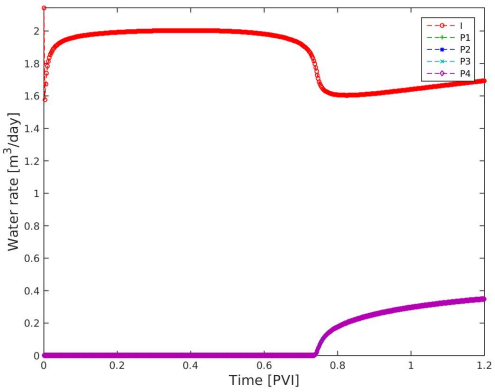

Figure 35: Water Rate, homogeneous permeability.

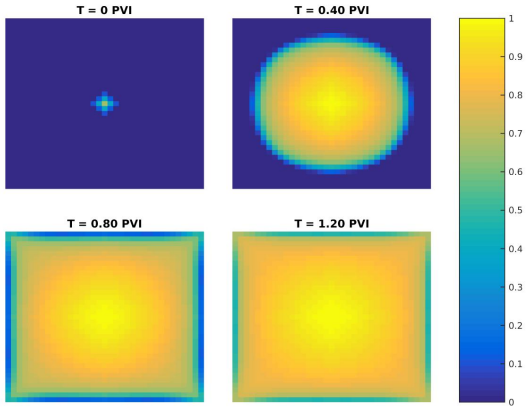

Figure 36: Water Saturation, homogeneous permeability.

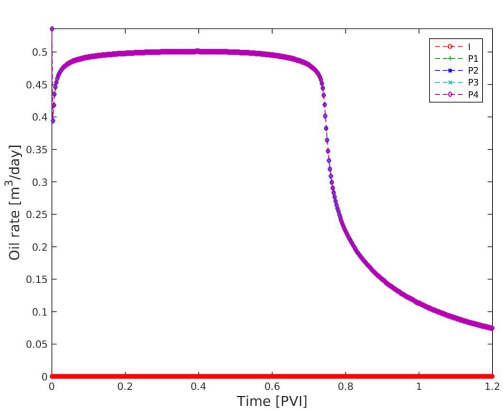

Figure 37: Oil rate, homogeneous permeability.

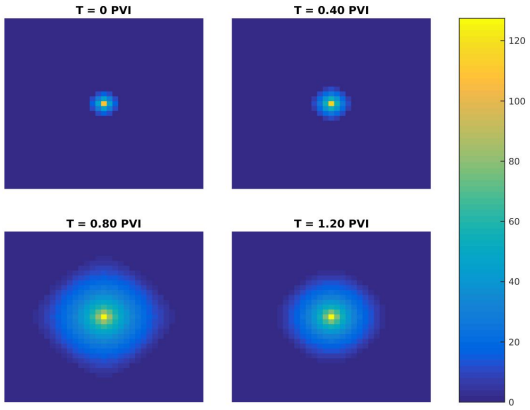

Figure 38: Pressure, homogeneous permeability.

maintained for some time until the water reaches the producers, then it decreases. For the

production of oil we observe a similar behavior, it begins with a constant rate and it drops after *wbt*. Water saturation in the reservoir is presented in Figures 40 and 44. We note that the high permeability layers hinder the movement of the water. For the pressure field (see Figures 42 and 46), we observe an increment in the pressure from the center outwards.

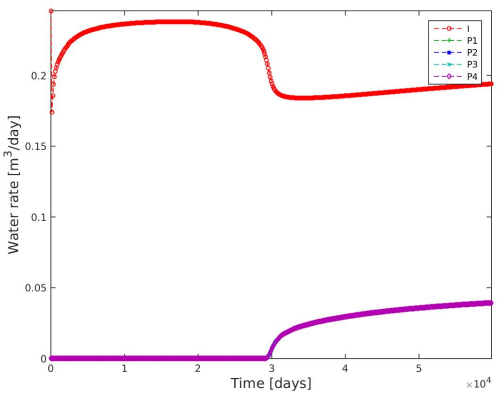

Figure 39: Water Rate, heterogeneous permeability (contrast 10^1).

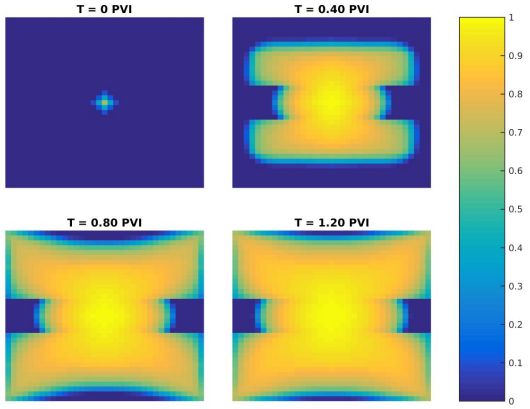

Figure 40: Water Saturation, heterogeneous permeability (contrast 10^1).

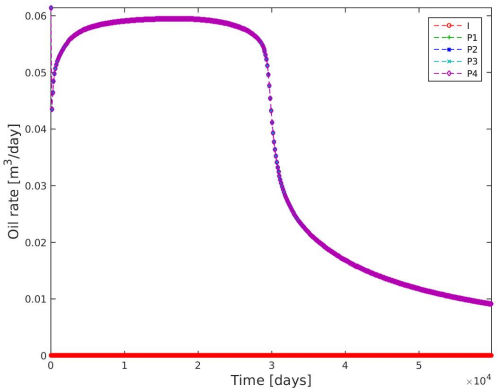

Figure 41: Oil rate, heterogeneous permeability (contrast 10^1).

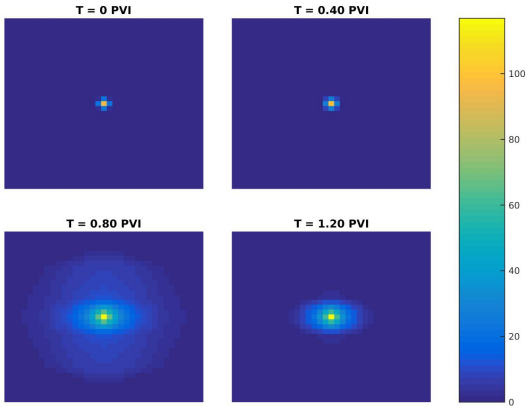

Figure 42: Pressure, heterogeneous permeability (contrast 10^1).

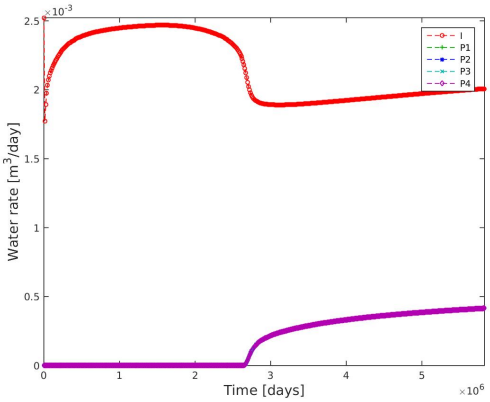

Figure 43: Water Rate, heterogeneous permeability (contrast 10^3).

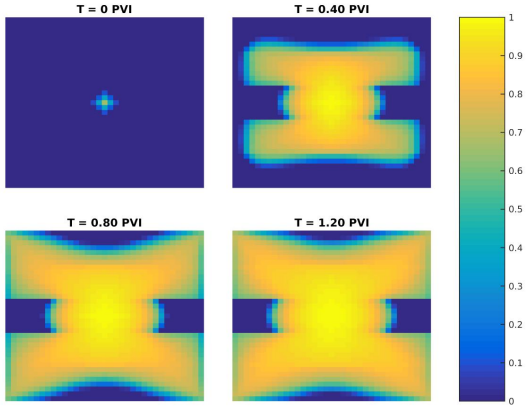

Figure 44: Water Saturation, heterogeneous permeability (contrast 10^3).

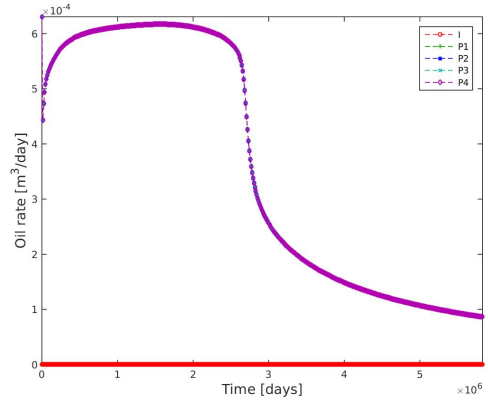

Figure 45: Oil rate, heterogeneous permeability (contrast 10^3).

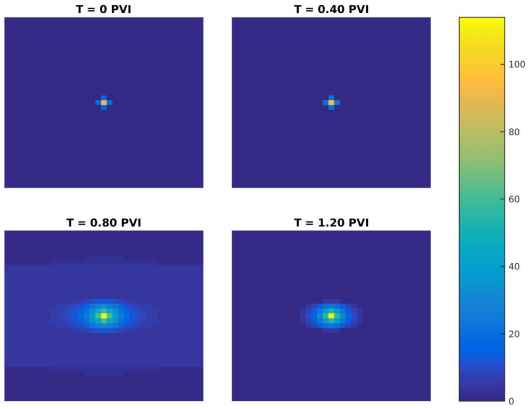

Figure 46: Pressure, heterogeneous permeability (contrast 10^3).

5.2 SPE 10

The SPE 10 model consists on 60 x 220 x 85 cells. In this work we study 3 cases. The first one containing one layer, the second with 35 layers and the last one containing 85 layers. We consider injection through the boundary for the 2D problem and injection through wells for the 3D cases. The wells setup consist on one injector and four producers and (see Figure 47). For the collection of snapshots, we use two criteria: *moving window* for the 2D case, and a *training phase* for the 3D cases. A detailed description of the number of deflation vectors, times, and pressures is presented for each case.

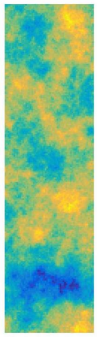

Figure 47: Rock permeability, SPE 10.

Injection through left boundary

Water is injected through the left boundary at a constant rate of 4 m^3/day to a reservoir initially filled with oil. The initial pressure of the reservoir is 100 bars, and the pressure in the right boundary is set to zero bars. We run the simulation for 100 time steps, with a step of 20 days (see Tables 18 and 19). We study the DICCG method with 30 and 10 deflation vectors. The stopping criterium for the ICCG and DICCG methods is $\epsilon = 5 \cdot 10^{-8}$. The eigenvalues of the correlation matrix are shown in Figure 48. The results are presented in Table 20 and the pressure and water saturation at diverse time steps are shown in Figures 49 and 50.

	Water	Oil
$S_{0,x\neq 0,L_x}$	0	1
$S_{x=0}$	1	0
$S_{x=L_x}$	0	1

Table 18: Initial Saturations.

Temporal Parameters		
dT	20	days
$Steps$	100	
T_{total}	2000	days
Boundary conditions		
$Q_{x=0}$	400	m^3/day
$P_{0,x\neq(0,L_x)}$	100	$bars$
$P_{x=L_x}$	0	$bars$

Table 19: Boundary conditions and temporal parameters.

Total ICCG Iterations	DICCG Method	ICCG Iterations (Snapshots)	DICCG Iterations	Total ICCG +DICCG	% of total ICCG Iterations
88502	DICCG_{10}	2018	25129	27147	31
88502	$\text{DICCG}_{POD_{30}}$	6120	16274	22394	25

Table 20: Number of iterations.

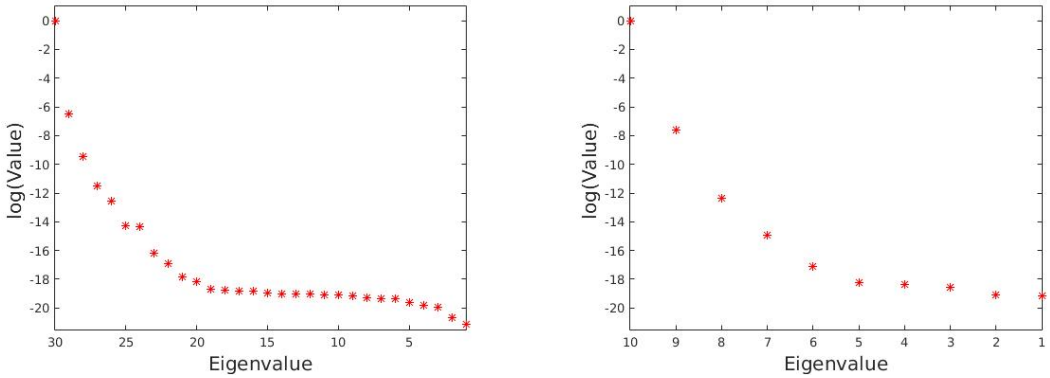

Figure 48: Eigenvalues of the correlation matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$ for 30 snapshots, left, and 10 snapshots, right.

The number of iterations is reduced to around 30% ICCG iterations when we use ten deflation vectors and around 25% when we use 30 (see Table 20). In Figure 48 we observe ten eigenvalues larger than the rest. This indicates that the main information is contained in the eigenvectors corresponding to these eigenvalues. Therefore, the performance of the deflated method does not change dramatically when using 30 or ten snapshots.

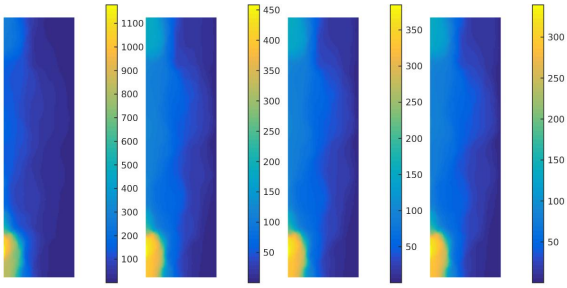

Figure 49: Pressure, SPE 10, 60 x 220 x 1 grid cells.

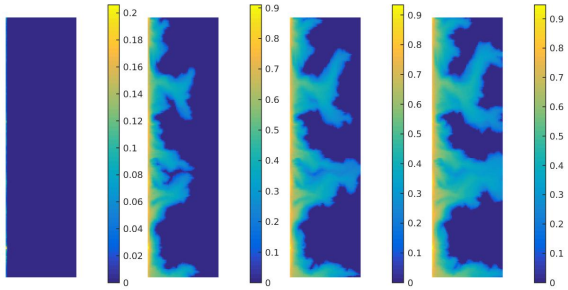

Figure 50: Water Saturation, SPE 10, 60 x 220 x 1 grid cells.

Wells

In this section, we perform a series of experiments injecting water through wells. For these examples, the POD basis and deflation matrix are obtained off-line in a training run with the ICCG method. Once the POD basis and the deflation matrix are obtained, a series of simulations are performed with the DICCG method for diverse values of *bhp* in the producers.

The pressure of the production wells is varied randomly every two time steps during the *training run* phase between 137.5 and 275 bars (see Table 21 and Figure 52). The pressure

in the injection well maintained constant at 1100 bars, and 275 bars for the producers. The pre-simulation is run during 100 time steps with a step of 50 days for the 35 layers case and 150 steps of 5 days for the 85 layers case.

After the *training simulation*, we compute the correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$,, where the columns of the matrix $\mathbf{X}$ are the solutions of the *training phase*. The eigenvectors of the correlation matrix are presented in Figure 53. We observe that some eigenvalues are larger than the rest. We use the eigenvectors of these large eigenvalues as deflation vectors. For the first set of experiments we use 30 vectors and for the second we use ten. We solve three cases with the deflation vectors previously selected. For the first two cases, we select *bhp* of the producers in the range of pressures computed during the *training phase* (200 and 275 [bars]). For the third case, we solve for a *bhp* in the wells outside the *training phase* pressures. The water saturation and the pressure field are presented in Figures 55 and 54. The resulting number of iterations are presented in Table 23 with 35 layers and Table 24 with 85.

Well	Water Sat	Oil Sat	Pressure [bars]
P_1, P_2	0	1	$rand(137.5 - 275)$
P_3	0	1	$275 - P_1$
P_4	0	1	$275 - P_2$
I_1, I_2	1	0	1100
Reservoir			
	Water Sat	Oil Sat	Pressure [bars]
	0	1	500

Table 21: Initial Pressure and Saturations in the reservoir and wells, training run.

Temporal Parameters		
35 layers		
dT	50	days
$Steps$	100	
T_{total}	5000	days
85 layers		
dT	50	days
$Steps$	100	
T_{total}	5000	days

Table 22: Temporal parameters.

From Table 23 and 24, we observe that the DICCG with 30 deflation vectors requires around 20% of the number of the ICCG iterations. This value is similar for the three cases, even if the problem is outside of the training range. When we decrease the number of deflation vectors to ten, the percentage of ICCG iterations increases to around 35%, which is still an important reduction in the number of iterations.

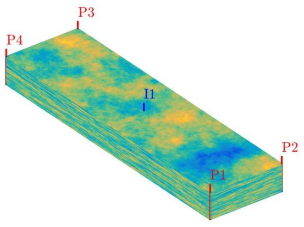

Figure 51: Rock permeability and wells, SPE 10, 220 x 60 x 35 grid cells.

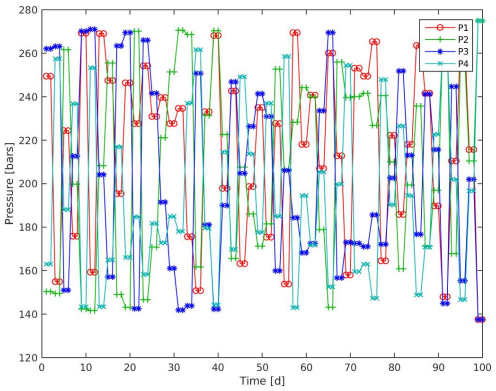

Figure 52: Producers *bhp* training simulation.

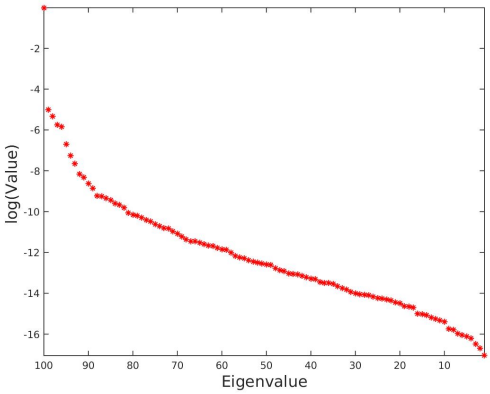

Figure 53: Eigenvalues training simulation.

Total ICCG Iter	DICCG Method	Total DICCG Iter	% ICCG Iter
$P_{bhp} = 275$ [bars]			
44107	$DICCG_{30}$	9636	22
44107	$DICCG_{10}$	15799	36
$P_{bhp} = 200$ [bars]			
44107	$DICCG_{30}$	8371	19
44107	$DICCG_{10}$	15327	35
$P_{bhp} = 400$ [bars]			
44107	$DICCG_{30}$	8371	19
44107	$DICCG_{10}$	15327	35

Table 23: Number of iterations for diverse *bhp* in the production wells, 35 layers.

Total ICCG Iter	DICCG Method	Total DICCG Iter	% ICCG Iter
$P_{bhp} = 275$ [bars]			
96594	$DICCG_{30}$	21526	22
96594	$DICCG_{10}$	37255	39
$P_{bhp} = 200$ [bars]			
96594	$DICCG_{30}$	21103	22
96594	$DICCG_{10}$	36225	38
$P_{bhp} = 400$ [bars]			
96594	$DICCG_{30}$	20855	22
96594	$DICCG_{10}$	35009	36

Table 24: Number of iterations for diverse *bhp* in the production wells, 85 layers.

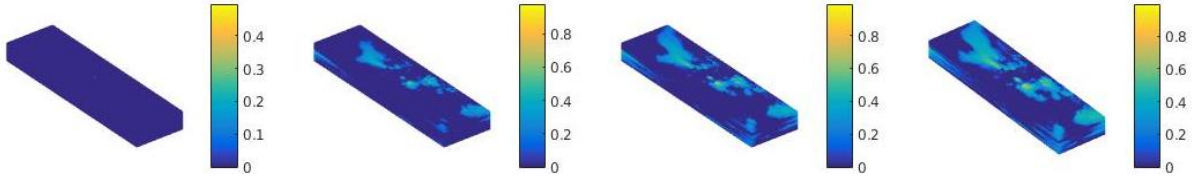

Figure 54: Water Saturation, SPE 10, 60 x 220 x 35 grid cells.

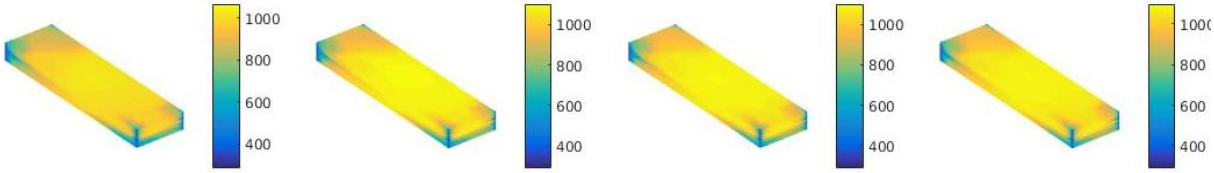

Figure 55: Pressure field, SPE 10, 60 x 220 x 35 grid cells.

Conclusions

Approximating a solution to large or ill-conditioned systems of linear equations with iterative methods leads to a computationally expensive process. This is usually the case of the linear pressure system when simulating two-phase flow through highly heterogeneous porous media. We propose the reuse of system information by combining POD and deflation techniques to accelerate this process.

In our proposal, the POD method is used to collect system information for later use in a deflation procedure (Deflated Conjugate Gradient preconditioned with Incomplete Cholesky, *DICCG*). The POD basis is obtained in two different ways: in the first one (*moving window* approach), the basis is constructed from previously computed solutions and updated at each time step; for the second one (*training phase* approach), the solutions of a pre-simulation are used to construct the basis. We studied injection of water through boundaries and through wells for an academic layered problem and the SPE 10 benchmark.

For the layered problems, we studied the *moving window* approach. The DICCG method reduced the number of ICCG iterations to around 14%. The performance of the method was independent of the contrast between permeability layers or the inclusion of gravity terms. However, the capillary terms increased the number of iterations by a factor of two. Reducing the time step for this problem resulted in a decrease of the number of iterations; whereas, increasing the number of deflation vectors did not show a remarkable change.

For the SPE 10 benchmark, we studied the *moving window* and *training phase* approaches. For both cases, the DICCG method reduced the ICCG iterations to around 20% of the total using 30 deflation vectors, and to around 35% with ten. These results where similar for all cases, including the full benchmark containing $\sim 12 \times 10^6$ cells and gravity terms included.

For a 3D case using five deflation vectors, the DICCG method costs 1.5 times more operations per iteration than the ICCG method (see Appendix F), where an acceleration up to a factor 5 is observed. As we increase the number of deflation vectors the cost increases. Therefore, the implementation of the DICCG method with fewer deflation vectors leads to an optimal performance.

We present the POD-based deflation method for reservoir simulation examples together with the CG method. Nevertheless, it can be applied to various transient problems, and multiple iterative linear solvers, e.g., Krylov subspace and Multigrid methods.

Acknowledgements

We like to thank the 'Consejo Nacional de Ciencia y Tecnología (CONACYT)', the 'Secretaría de Energía (SENER)' and the Mexican Insitute of Petroleum (IMP) which, through the programs: 'Formación de Recursos Humanos Especializados para el Sector Hidrocarburos (CONACYT-SENER Hidrocarburos)' and 'Programa de Captación de Talento, Reclutamiento, Evaluación y Selección de Recursos Humanos (PCTRES)', have sponsored this work.

References

[1] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics Philadelphia, PA, USA. 2nd edition, 2003.

[2] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics*, 182(2):418–477, 2002.

[3] 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.

[4] 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(5):525–541, 2006.

[5] 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(8):1159–1179, 2017.

[6] Y. Saad, M. Yeung, J. Erhel, Jocelyne and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926, 2000.

[7] 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*, 152:385, 1999.

[8] 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(Supplement C):193 – 213, 2018.

[9] G.B. Diaz Cortes, C. Vuik and J.D. Jansen. Physics-based Pre-conditioners for Large-scale Subsurface Flow Simulation. In *Proceedings of the 15th European Conference on the Mathematics of Oil Recovery, ECMOR XV*, 2016.

[10] 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(1):57–69, 2004.

[11] W. Schilders, H. Van der Vorst and J. Rommes. *Model order reduction: theory, research aspects and applications*, volume 13. Springer, 2008.

[12] A. Quarteroni and G. Rozza. *Reduced order methods for modeling and computational reduction*, volume 9. Springer, 2014.

[13] 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(3):1304–1336, 2016.

[14] 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(9):1322–1350, 2009.

[15] T. Heijn, R. Markovinovic, J.D. Jansen et al. Generation of low-order reservoir models using system-theoretical concepts. *SPE Journal*, 9(02):202–218, 2004.

[16] J. van Doren, R. Markovinović and J.D. Jansen. Reduced-order optimal control of water flooding using proper orthogonal decomposition. *Computational Geosciences*, 10(1):137–158, 2006.

[17] H.A. Van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13. Cambridge University Press, 2003.

[18] H.A. Van der Vorst, C. Dekker. Conjugate gradient type methods and preconditioning. *Journal of Computational and Applied Mathematics*, 24:73–87, 11 1988.

[19] 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(1):219–233, 2002.

[20] 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(3):340–370, 2009.

[21] J. Tang and C. Vuik. On deflation and singular symmetric positive semi-definite matrices. *Journal of computational and applied mathematics*, 206(2):603–614, 2007.

[22] J. Tang. *Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems*. PhD thesis, Delft University of Technology, 2008.

[23] G.B. Diaz Cortes, C. Vuik and J.D. Jansen. On POD-based Deflation Vectors for DPCG applied to porous media problems. Report 17-1, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2017.

[24] K.A. Lie. *An Introduction to Reservoir Simulation Using MATLAB: User guide for the Matlab Reservoir Simulation Toolbox (MRST).* SINTEF ICT, 2013.

[25] K. Aziz and A. Settari. *Petroleum reservoir simulation*. Chapman & Hall, 1979.

[26] Z. Chen, G. Huan and Y. Ma. *Computational methods for multiphase flows in porous media*. SIAM, 2006.

[27] J.D. Jansen. *A systems description of flow through porous media.* Springer, 2013.

[28] 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, 03 2016.

[29] M. Clemens, M. Wilke, R. Schuhmann and T. Weiland. Subspace projection extrapolation scheme for transient field simulations. *IEEE Transactions on Magnetics*, 40(2):934–937, 2004.

[30] B. Smith, P. Bjorstad and W. Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations.* Cambridge University Press New York, 1996.

[31] M.A. Christie and M.J. Blunt. Tenth SPE Comparative Solution Project: a Comparison of Upscaling Techniques. *SPE Reservoir Engineering and Evaluation*, 4(4):308–317, 2001.

[32] J.J. Leader. *Numerical analysis and scientific computation.* Pearson Addison Wesley, 2004.

A List of notation

Symbol	Quantity	Unit
α	Fluid phase	
ϕ	Rock porosity	
$\mathbf{K}$	Rock permeability	$Darcy\ (D)$
$\mathbf{K}_\alpha$	Effective permeability	$Darcy\ (D)$
$\mathbf{k}_{r\alpha}$	Relative permeability	
c_r	Rock compressibility	Pa^{-1}
$\mathbf{v}_\alpha$	Darcy's velocity	m/d
ρ_α	Fluid density	kg/m^3
μ_α	Fluid viscosity	$Pa \cdot s$
λ_α	Fluid mobilities	$D/(Pa \cdot s)$
p	Pressure	Pa
p_c	Capillary pressure	Pa
S_α	Saturation	
g	Gravity	m/s^2
c_f	Fluid compressibility	Pa^{-1}
q_α	Sources	
d	Reservoir depth	

Table 25: Notation

B Krylov iterative methods: CG

In this appendix, we explain how to obtain the solution of the linear system:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{28}$$

using diverse Krylov-subspace techniques.

Conjugate Gradient (CG) method is a Krylov-subspace iterative method based in the minimization of the residual in the A-norm. Given a starting solution $\mathbf{x}^0$ and the residual defined by $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$, we define the Krylov subspace as:

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}^0) = span\{\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \ldots, \mathbf{A}^{k-1}\mathbf{r}^0\}.$$

For this space, the approximate solution

$$\mathbf{x}^k \in \mathbf{x}^0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}^0)$$

has a minimal error measured in the $\mathbf{A}$-norm for all approximations contained in $\mathbf{x}^0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}^0)$.
The error of this approximation is bounded by:

$$||\mathbf{x} - \mathbf{x}^{k+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}^0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} \right)^{k+1} {}^{2}. \tag{29}$$

Hence, the convergence of the method depends on the condition number (κ_2) of the matrix.

B.1 Preconditioning

The linear system (28) can be further accelerated by transforming the system into a one with better spectrum, this is done by multiplying the system by a matrix, cheap to compute $\mathbf{M}^{-1}$,

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}. \tag{30}$$

The new system has the same solution but can provide a substantial reduction of the condition number. For this preconditioned system, the error is bounded by:

$$||\mathbf{x} - \mathbf{x}^k||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}^0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} + 1} \right)^{k}. \tag{31}$$

[2]The condition number $\kappa_2(\mathbf{A})$ is defined as $\kappa_2(\mathbf{A}) = \frac{\sqrt{\lambda_{max}(\mathbf{A}^T\mathbf{A})}}{\sqrt{\lambda_{min}(\mathbf{A}^T\mathbf{A})}}$. If $\mathbf{A}$ is SPD, $\kappa_2(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})}$.

B.2 Deflated PCG Method

Another way to accelerate the solution of a linear system is the deflation method. Where, instead of solving the original system, we solve the deflated system (see Appendix C):

$$\mathbf{PA}\hat{\mathbf{x}} = \mathbf{Pb}, \tag{32}$$

for the deflated solution $\hat{\mathbf{x}}$. This deflated the solution is related to the solution $\mathbf{x}$ of the original system as (see Appendix C):

$$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T\hat{\mathbf{x}}. \tag{33}$$

The deflated linear system can also be preconditioned by an SPD matrix $\mathbf{M}$. After preconditioning, the deflated preconditioned system to solve with CG is [20]:

$$\tilde{\mathbf{P}}\tilde{\mathbf{A}}\hat{\tilde{\mathbf{x}}} = \tilde{\mathbf{P}}\tilde{\mathbf{b}},$$

where:

$$\tilde{\mathbf{A}} = \mathbf{M}^{-\frac{1}{2}}\mathbf{A}\mathbf{M}^{-\frac{1}{2}}, \qquad \hat{\tilde{\mathbf{x}}} = \mathbf{M}^{\frac{1}{2}}\hat{\mathbf{x}}, \qquad \tilde{\mathbf{b}} = \mathbf{M}^{-\frac{1}{2}}\mathbf{b}.$$

This method is called the Deflated Preconditioned Conjugate Gradient $DPCG$ method. In practice $\mathbf{M}^{-1}\mathbf{PAx} = \mathbf{M}^{-1}\mathbf{Pb}$ is computed and the error is bounded by:

$$||\mathbf{x} - \mathbf{x}^{i+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}^0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})} - 1}{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})} + 1} \right)^{i+1},$$

were $\kappa_{eff} = \frac{\lambda_{max}(M^{-1}PA)}{\lambda_{min}(M^{-1}PA)}$ is the effective condition number and $\lambda_{min}(M^{-1}PA)$ is the smallest non-zero eigenvalue of $M^{-1}PA$.

C Deflation method properties

Some properties of the matrices used for deflation that will help us to find the solution of system (28) are [22]:

a) $\mathbf{P}^2 = \mathbf{P}$.

b) $\mathbf{A}\mathbf{P}^T = \mathbf{P}\mathbf{A}$.

c) $(\mathbf{I} - \mathbf{P}^T)\mathbf{x} = \mathbf{Q}\mathbf{b}$.

d) $\mathbf{P}\mathbf{A}\mathbf{Q} = \mathbf{0}^{n\times n}$.

e) $\mathbf{P}\mathbf{A}\mathbf{Z} = \mathbf{0}^{n\times l}$.

To obtain the solution of the linear system (28), we start with the splitting:

$$\mathbf{x} = \mathbf{x} - \mathbf{P}^T\mathbf{x} + \mathbf{P}^T\mathbf{x} = (\mathbf{I} - \mathbf{P}^T)\mathbf{x} + \mathbf{P}^T\mathbf{x}. \tag{34}$$

Multiplying expression (34) by $\mathbf{A}$, using the properties of the deflation matrices, we have:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{A}(\mathbf{I} - \mathbf{P}^T)\mathbf{x} + \mathbf{A}\mathbf{P}^T\mathbf{x}, && \textit{Property}: \\ \mathbf{A}\mathbf{x} &= \mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{A}\mathbf{P}^T\mathbf{x}, && c) \\ \mathbf{b} &= \mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{P}\mathbf{A}\mathbf{x}, && b), \end{aligned}$$

multiplying by $\mathbf{P}$ and using the properties $\mathbf{P}\mathbf{A}\mathbf{Q} = \mathbf{0}^{n\times n}$ and $\mathbf{P}^2 = \mathbf{P}$, properties d) and a), we have:

$$\begin{aligned} \mathbf{P}\mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{P}^2\mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{b}, \\ \mathbf{P}\mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{b}, \end{aligned}$$

where $\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{P}\mathbf{b}$ is the deflated system. Since $\mathbf{P}\mathbf{A}$ is singular, the solution of Equation (35) can contain components of the null space of $\mathbf{P}\mathbf{A}$, ($\mathcal{N}(\mathbf{P}\mathbf{A})$). A solution of this system, called the deflated solution, is denoted by $\hat{\mathbf{x}}$. Then, the linear system to solve is:

$$\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}. \tag{35}$$

As the solution of Equation (35) can contain components of $\mathcal{N}(\mathbf{P}\mathbf{A})$, $\hat{\mathbf{x}}$ can be decomposed as:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{y}, \tag{36}$$

with $\mathbf{y} \in \mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{P}\mathbf{A})$, and $\mathbf{x}$ the solution of Equation (28).
Note: If $\mathbf{y} \in \mathcal{R}(\mathbf{Z})$, then

$$\mathbf{y} = \sum_{i=1}^{m} \alpha_i \mathbf{z}_i,$$

$$\mathbf{PAy} = \mathbf{PA}(\mathbf{z}_1\alpha_1 + ... + \mathbf{z}_m\alpha_m) = \mathbf{PAZ}\alpha,$$

from property e) we have:

$$\mathbf{PAy} = \mathbf{0}.$$

Therefore $\mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{PA})$, and using property b) we have:

$$\mathbf{PAy} = \mathbf{AP}^T\mathbf{y} = \mathbf{0}.$$

As $\mathbf{A}$ is invertible, we have:

$$\mathbf{P}^T\mathbf{y} = \mathbf{0}. \tag{37}$$

Multiplying Equation (36) by $\mathbf{P}^T$ we obtain:

$$\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x} + \mathbf{P}^T\mathbf{y}.$$

substituting Equation (37) we arrive to:

$$\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x}. \tag{38}$$

Substitution of Equation (38) and property c) in Equation (34) leads to:

$$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T\hat{\mathbf{x}}, \tag{39}$$

which gives us the relation between $\hat{\mathbf{x}}$ and $\mathbf{x}$.

D Stopping criteria

When we use an iterative method, we always want that our approximation is close enough to the exact solution. In other words, we want that the error [1, pag. 42]:

$$||\mathbf{e}^k||_2 = ||\mathbf{x} - \mathbf{x}^k||_2,$$

or the relative error:

$$\frac{||\mathbf{x} - \mathbf{x}^k||_2}{||\mathbf{x}||_2},$$

is small.

When we want to chose a stopping criteria, we could think that the relative error is a good candidate, but is has the disadvantage that we need to know the exact solution to compute it. What we have instead is the residual

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k,$$

that is actually computed in each iteration of the CG method. There is a relationship between the error and the residual that can help us with the choice of the stopping criteria.

$$\frac{||\mathbf{x} - \mathbf{x}^k||_2}{||\mathbf{x}||_2} \leq \kappa_2(A)\frac{||\mathbf{r}^k||_2}{||\mathbf{b}||_2}.$$

With this relationship in mind, we can choose the stopping criteria as an ϵ for which

$$\frac{||\mathbf{r}^k||_2}{||\mathbf{b}||_2} \leq \epsilon.$$

But we should keep to have in mind the condition number of the matrix $\mathbf{A}$, because the relative error will be bounded by:

$$\frac{||\mathbf{x} - \mathbf{x}^k||_2}{||\mathbf{x}||_2} \leq \kappa_2(A)\epsilon.$$

E Singular Value Decomposition for POD

If we perform SVD in a set of vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m]$, we obtain [32]:
$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T \in \mathbb{R}^{n\times m}, \quad \mathbf{U} \in \mathbb{R}^{n\times n}, \quad \mathbf{\Sigma} \in \mathbb{R}^{n\times m}, \quad \mathbf{V} \in \mathbb{R}^{m\times m}.$
We can decompose the data correlation matrix $\mathbf{R}$ and the transpose of this matrix as:

$$\begin{aligned}\mathbf{R} &= \mathbf{X}\mathbf{X}^T \\ &= \mathbf{U}\Sigma\mathbf{V}^T(\mathbf{U}\Sigma\mathbf{V}^T)^T \\ &= \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma^T\mathbf{U}^T,\ \mathbf{V}^T\mathbf{V} = \mathbf{I} \\ &= \mathbf{U}\Lambda\mathbf{U}^T,\ \Lambda = \Sigma\Sigma^T \in \mathbb{R}^{n\times n}\end{aligned} \qquad \begin{aligned}\mathbf{R}^T &= \mathbf{X}^T\mathbf{X} \\ &= (\mathbf{U}\Sigma\mathbf{V}^T)^T\mathbf{U}\Sigma\mathbf{V}^T \\ &= \mathbf{V}\Sigma^T\mathbf{U}^T\mathbf{U}\Sigma\mathbf{V}^T,\ \mathbf{U}^T\mathbf{U} = \mathbf{I} \\ &= \mathbf{V}\Lambda^T\mathbf{V}^T,\ \Lambda^T = \Sigma^T\Sigma \in \mathbb{R}^{m\times m}.\end{aligned}$$

The eigenvectors of $\mathbf{X}$ can be obtained as follows:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$$
$$\mathbf{U} = \mathbf{X}\mathbf{V}\Sigma^{-1}$$
$$\mathbf{U} = \mathbf{X}\mathbf{V}\Lambda^{-\frac{1}{2}}$$

Where we need the eigenvalues of $\mathbf{R}$, if we compute the eigenvectors of $\mathbf{R}^T$ instead, Λ^T, we can compute $\mathbf{U}$ as follows:

$$\mathbf{U} = \mathbf{X}\mathbf{V}(\Lambda^T)^{-\frac{T}{2}} = \mathbf{X}\mathbf{V}(\Lambda^T)^{\frac{1}{2}}$$

F Operation counts

The number of operations necessary to perform the deflation procedure is computed in this section for full matrices and sparse matrices.

First, we compute the number of operations between vectors and matrices necessary for ICCG method (see Table 26) and DICCG method (see Table 27).

With the numbers previously computed, we compute the number of operations necessary to perform the ICCG (see Table 28) and DICCG methods (see Table 29). In Table 31, we compute the number of operations necessary to perform the ICCG and DICCG methods for different sparsity of the matrix (m) and a diverse number of deflation vectors (p).

$\mathbf{A}$, $\mathbf{B} \in \mathbb{R}^{n\times n}$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$		
Operation	Number of Operations	
	Full matrix	Sparse matrix (m non zero entries)
$\mathbf{x}^T\mathbf{y}$	$n(*) + (n-1)(+) = 2n-1$	$2n-1$
$\mathbf{x}(+/-)\mathbf{y}$	n	n
$\alpha\mathbf{x}$	n	n
$\mathbf{Ax}$	$(n(*) + (n-1)(+))n$ (r) $= 2n^2 - n$	$(m(*) + m - 1(+))n$ (r) $= 2mn - n$
$\mathbf{AB}$	$[(n(*) + (n-1)(+))n$ (r)$]n$ (c) $=$ $= 2n^3 - n^2$	$[(m(*) + m - 1(+))n$ (r) $]m$ (c) $=$ $= 2m^2n - nm$
$\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$, $\mathbf{z} \in \mathbb{R}^p$		
$\mathbf{AB}$	$([n(*) + n - 1(+)]p + (p-1)(+))p$ $= (2np-1)p$	$([m(*) + m - 1(+)]p + (p-1)(+))p$ $= (2mp-1)p$
$\mathbf{Ax}$	$(n(*) + (n-1)(+))p = (2n-1)p$	$(m(*) + (m-1)(+))p = (2m-1)p$
$\mathbf{Bz}$	$(p(*) + (p-1)(+))n = (2p-1)n$	$(m(*) + (m-1)(+))n = (2m-1)n$
$\mathbf{A} = \mathbf{L}\mathbf{L}^T$	$1/3n^3$	
$\mathbf{Lx} = \mathbf{y}$	n^2	nm
$\mathbf{L}^T\mathbf{x} = \mathbf{y}$	n^2	nm
$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T \in \mathbb{R}^{n\times m}$	nm^2	nm^2

Table 26: Number of operations between matrices and vectors.

$\mathbf{Z} \in \mathbb{R}^{n\times p}$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$		
Operation	Number of Operations	
	Full matrix	Sparse matrix (m non zero entries)
$\mathbf{V} = \mathbf{AZ} \quad \in \mathbb{R}^{n\times p}$	$np(2n-1)$	$np(2m-1)$
$\mathbf{Z}^T\mathbf{V} \quad \in \mathbb{R}^{n\times p}$	$p^2(2n-1)$	$p^2(2m-1)$
$\mathbf{E} = \mathbf{Z}^T\mathbf{AZ} \quad \in \mathbb{R}^{p\times p}$	$p[n(2n+2p-1)-p]$	$p(2m-1)(p+n)$
$\mathbf{E}^{-1}$	p^3	p^3
$\mathbf{B} = \mathbf{AZE}^{-1} = \mathbf{VE}^{-1} \quad \in \mathbb{R}^{n\times p}$	$np(2p-1)$	$np(2p-1)$
$\mathbf{y} = \mathbf{Z}^T\mathbf{x}$	$2np-p$	$2np-p$
$\mathbf{z} = \mathbf{By} \quad \in \mathbb{R}^n$	$n(2p-1)$	$n(2p-1)$
$\mathbf{w} = \mathbf{E}^{-1}\mathbf{y}$	$2p^2-p$	$2p^2-p$
$\mathbf{Q} = \mathbf{Zw}$	$2pn-n$	$2pn-n$
$\mathbf{Qx} = \mathbf{ZE}^{-1}\mathbf{Z}^T\mathbf{x}$	$(4p-1)n+p^2-2p$	$(4p-1)n+p^2-2p$
$\mathbf{Q}_E\mathbf{x} = \mathbf{ZE}^{-1}\mathbf{Z}^T\mathbf{x}$	$(4np+2p-1)n-2p+p^3$	$(4mp+2p-1)n-2p+p^3$
(computing $\mathbf{E}$ and $\mathbf{E}^{-1}$)		
$\mathbf{Vw}$	$(2p-1)n$	$(2p-1)n$
$\mathbf{AQx} = \mathbf{AZE}^{-1}\mathbf{Z}^T\mathbf{x} =$		
$= [\mathbf{AZE}^{-1}][\mathbf{Z}^T\mathbf{x}] = [\mathbf{B}][\mathbf{Z}^T\mathbf{x}]$	$[2np-p]+[n(2p-1)]$	$[2np-p]+[n(2p-1)]$
(without computing $\mathbf{B}$)	$= n(4p-1)-p$	$= n(4p-1)-p$
$\mathbf{Px} = (\mathbf{I}-\mathbf{AQ})\mathbf{x} = \mathbf{x}-\mathbf{B}[\mathbf{Z}^T\mathbf{x}]$	$4np-p$	$4np-p$
(without computing $\mathbf{B}$)		
$\mathbf{P}_E\mathbf{x} = (\mathbf{I}-\mathbf{AQ})\mathbf{x}$	$(2n+4np+2p-1)n-$	$(2m+4mp+2p-1)n-$
	$-2p+p^3$	$-2p+p^3$

Table 27: Number of operations required to compute matrices and vectors for the deflation method.

Algorithm 1 ICCG method, solving $\mathbf{Ax} = \mathbf{b}$.	Operations	
Split preconditioner	Full matrix	Sparse matrix
Give an initial guess $\mathbf{x}^0$.		
Compute $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$.	$2n^2$	$2mn$
Compute $\hat{\mathbf{r}}^0 = \mathbf{L}^{-1}\mathbf{r}^0$.	n^2	nm
Compute $\hat{\mathbf{p}}^0 = \mathbf{L}^{-T}\hat{\mathbf{r}}^0$.	n^2	nm
for $k = 0, ...,$ until convergence		
$\mathbf{w}^k = \mathbf{A}\mathbf{p}^k$	$2n^2 - n$	$2nm - n$
$\mathbf{ry}^k = (\hat{\mathbf{r}}^k, \mathbf{y}^k)$	$2n - 1$	$2n - 1$
$\alpha^k = \frac{\mathbf{ry}^k}{(\mathbf{p}^k, \hat{\mathbf{w}}^k)}$	$2n$	$2n$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{p}^k$	$2n$	$2n$
$\hat{\mathbf{r}}^{k+1} = \hat{\mathbf{r}}^k - \alpha^k\mathbf{L}^{-1}\hat{\mathbf{w}}^k$	$n^2 + 2n$	$nm + 2n$
$\beta^k = \frac{(\hat{\mathbf{r}}^{k+1}, \hat{\mathbf{y}}^{k+1})}{\mathbf{ry}^k}$	$2n$	$2n$
$\mathbf{p}^{k+1} = \mathbf{L}^{-T}\mathbf{r}^{k+1} + \beta^k\mathbf{p}^k$	$n^2 + 2n$	$nm + 2n$
end		
Total each k	$4n^2 + 11n - 1$	$4nm + 11n - 1 \sim$ $\sim (4m + 11)n$

Table 28: Number of operations required to perform the ICCG method.

Algorithm 3 DICCG method, solving $\mathbf{Ax} = \mathbf{b}$.	Operations	
Split preconditioner	Full matrix	Sparse matrix
Give an initial guess $\mathbf{x}^0$.		
Compute:		
$\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0$	$2n^2$	$2mn$
$\mathbf{V} = \mathbf{AZ}$	$np(2n-1)$	$np(2m-1)$
$\hat{\mathbf{r}}^0 = \mathbf{Pr}^0$	$4np - p$	$4np - p$
$\mathbf{y}^0 = \mathbf{L}^{-1}\hat{\mathbf{r}}^0$	n^2	nm
$\mathbf{p}^0 = \mathbf{L}^{-T}\mathbf{y}^0$	n^2	nm
for $k = 0, ...,$ until convergence		
$\mathbf{w}^k = \mathbf{Ap}^k$	$2n^2 - n$	$2nm - n$
$\hat{\mathbf{w}}^k = \mathbf{Pw}^k$	$4np - p$	$4np - p$
$\mathbf{ry}^k = (\hat{\mathbf{r}}^k, \mathbf{y}^k)$	$2n - 1$	$2n - 1$
$\alpha^k = \frac{\mathbf{ry}^k}{(\mathbf{p}^k, \hat{\mathbf{w}}^k)}$	$2n$	$2n$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$	$2n$	$2n$
$\hat{\mathbf{r}}^{k+1} = \hat{\mathbf{r}}^k - \alpha^k \mathbf{L}^{-1}\hat{\mathbf{w}}^k$	$n^2 + 2n$	$nm + 2n$
$\beta^k = \frac{(\hat{\mathbf{r}}^{k+1}, \hat{\mathbf{y}}^{k+1})}{\mathbf{ry}^k}$	$2n$	$2n$
$\mathbf{p}^{k+1} = \mathbf{L}^{-T}\mathbf{r}^{k+1} + \beta^k \mathbf{p}^k$	$n^2 + 2n$	$nm + 2n$
end		
Total each k	$4n^2 + 4pn + 11n - p - 1$	$4nm + 4pn + 11n - p - 1$ $\sim (4m + 4p + 11)n$

Table 29: Number of operations required to perform the DICCG method.

F.1 Operation counts for the POD-based deflation method.

For the POD-based deflation method, first, we need to compute the snapshots with the ICCG method. Setting s as the number of snapshots, the work necessary to compute the snapshots is $31Ns$ for a 2D problem with N grid cells (see Table 30). The number of operations required to compute the ICCG method and the DICCG method with different deflation vectors is presented in Table 30. In this table, we also present a comparison between the two methods. We note that as we increase the number of deflation vectors, the DICCG method becomes more expensive. However, if the matrix becomes less sparse, the becomes less expensive, compared with the ICCG method.

Description	Sparsity	Operations	Count		
			$s = 5$	$s = 10$	$s = 30$
Compute	$m = 3$ (1D)	$23Ns$	$115N$	$230N$	$690N$
Snapshots	$m = 5$ (2D)	$31Ns$	$155N$	$310N$	$930N$
for various dimensions	$m = 7$ (3D)	$39Ns$	$195N$	$390N$	$1080N$
Compute SVD of $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T \in \mathbb{R}^{N\times s}$		Ns^2	25N	100N	900N
Total	$m = 3$	$23Ns + Ns^2$	$140N$	$330N$	$1590N$
number of	$m = 5$	$31Ns + Ns^2$	$180N$	$410N$	$1830N$
operations	$m = 7$	$39Ns + Ns^2$	$220N$	$490N$	$1980N$

Table 30: Number of operations required to compute the POD basis. If the snapshots are computed using the ICCG method, for which we need $s \times ICCG = s \times (4m + 11)N$ operations.

			m	p=5	p=10	p=30
	ICCG	(4m+11)N	23N	23N	23N	23N
m=3	DICCG	(4m+11+4p)N	(23+4p)N	43N	63N	143N
	DICCG/ICCG			43/23=1.8	63/23=2.7	143/23=6.2
	ICCG	(4m+11)N	31N	31N	31N	31N
m=5	DICCG	(4m+11+4p)N	(31+4p)N	51N	71N	151N
	DICCG/ICCG			51/31=1.6	71/31=2.3	151/31=4.8
	ICCG	(4m+11)N	39N	39N	39N	39N
m=7	DICCG	(4m+11+4p)N	(39+4p)N	59N	79N	159N
	DICCG/ICCG			59/39=1.5	79/39=2	5159/39=4.1

Table 31: Number of operations required per iteration to solve a linear system with the ICCG and DICCG methods for different sparsity of the matrices and different number deflation vectors.