On The Acceleration Of Ill-Conditioned Linear Systems
A Pod-Based Deflation Method For The Simulation Of Two-Phase Flow

Diaz Cortes, Gabriela; Jansen, Jan Dirk; Vuik, Kees

DOI
10.3997/2214-4609.201802122

Publication date
2018

Document Version
Final published version

Published in
ECMOR XVI

Mo A2 02

On The Acceleration Of Ill-Conditioned Linear Systems: A Pod-Based Deflation Method For The Simulation Of Two-Phase Flow

G.B. Diaz Cortes* (Delft University of Technology), J.D. Jansen (Delft University of Technology), C. Vuik (Delft University of Technology)

Summary

We explore and develop POD-based deflation methods to accelerate the solution of large-scale linear systems resulting from two-phase flow simulation.

The techniques here presented collect information from the system in a POD basis, which is later used in a deflation scheme.

The snapshots required to obtain the POD basis are captured in two ways: a moving window approach, where the most recently computed solutions are used, and a training phase approach, where a full pre-simulation is run.

We test this methodology in highly heterogeneous porous media: a full SPE 10 model containing O(10^6) cells, and in an academic layered problem presenting a contrast in permeability layers up to 10^6. Among the experiments, we study cases including gravity and capillary pressure terms.

With the POD-based deflated procedure, we accelerate the convergence of a Preconditioned Conjugate Gradient (PCG) method, reducing the required number of iterations to around 10-30 %, i.e., we achieve speed-ups of factors three to ten.

Introduction

For some reservoir simulation problems, difficulties arise when solving the pressure resulting linear systems, which, depending on the rock or fluid properties, may contain high contrasts in the matrix coefficients and may become very large. In this work, we explore the use of a new methodology for the acceleration of the solution of such $ill-conditioned$ systems.

A widely-used approach to speed up the solution of difficult large-scale systems of equations is combining iterative solvers with preconditioning techniques. The latter, speed up the convergence of the iterative methods by changing the system into another one with the same solution but a smaller condition number (Saad et al., 2000). However, in some cases, a small set of extreme eigenvalues is responsible of the large condition number and preconditioning techniques are no longer effective.

Therefore, new techniques have to be developed, that together with the usual preconditioned iterative methods can find approximate solutions in a faster way. Recently, Proper Orthogonal Decomposition (POD) methods (Markovinović and Jansen, 2006; Astrid et al., 2011; Pasetto et al., 2017; Diaz-Cortes et al., 2018) and deflation techniques (Vuik et al., 1999, 2002; Diaz-Cortes et al., 2016) based on system information, have been studied to accelerate iterative methods for large and ill-conditioned problems. An extensive literature exists, with new and innovative ways of approaching deflation and POD methodologies.

Deflation techniques can be used to remove the influence of extreme eigenvalues by creating a subspace where they are no longer present, and an approximate solution can be found in a faster way. For an optimal performance, it is necessary to find good deflation vectors that contain most of the system's variability. If a good selection is made, we can obtain an important decrease in the total simulation time, with only a small increase in the required computing time per iteration but a significant reduction of the number of iterations.

Currently, the selection of the deflation vectors is mainly based on some standard approaches: approximated eigenvectors, recycling solutions (Clemens et al., 2004; Diaz-Cortes et al., 2018), subdomain deflation vectors (Vuik et al., 2002) and multigrid and multilevel-based deflation matrices (Tang et al., 2009; Smith et al., 1996). However, a good selection of deflation vectors is problem-dependent, which implies the need of finding good deflation vectors for each kind of problem.

POD methods are based on the collection of a series of snapshots, i.e., solutions of the system with slightly different characteristics, from which essential system information will be condensed in a basis that can accurately represent the system. Acceleration with POD methods has been approached with diverse ways of collecting and recycling the information.

Some state-of-the-art POD acceleration strategies can be found in, e.g. Astrid et al. (2011), who propose the capture of system information in an offline phase for a later reuse, accelerating problems with slightly modified parameters in a smaller subspace created with this basis. This approach is particularly useful for optimization or history-matching problems where multiple very similar systems need to be solved. Alternatively, Markovinović and Jansen (2006) propose using the solution computed with POD to find a more accurate initial guess, while Pasetto et al. (2017) construct a preconditioner based on the reduced model for the acceleration of a Krylov-subspace iterative method.

Combining the strength of POD and deflation methods could lead to even higher acceleration factors and it could become a general way to select deflation vectors. In this work we introduce this acceleration approach, to which we refer to as a POD-based deflation method, and we study its applicability and properties.

Ill-conditioned linear systems in reservoir simulation

We consider the solution of systems of linear equations resulting from the iterative (Newton-based) solution of spatially and temporally discretized partial differential equations, as used in reservoir simulation. In particular we consider immiscible two-phase (oil/water) flow, including gravity and capillary forces. We use the fractional flow formulation to decouple pressure from saturation and we solve the resulting system with sequential schemes. To obtain the linear pressure system and to solve the transport equation we use the Matlab Reservoir Simulation Toolbox (MRST, Lie (2013)).

In many cases, reservoir simulation involves large and highly heterogeneous problems, i.e., problems with large variations in the permeability coefficients $\mathbf{K}_\alpha$, also known as *ill-conditioned* problems, which lead to large computing times. Furthermore, if we have a time-varying problem, we must compute a large number of simulations, which makes the solution of the problem very expensive.

Sometimes, the solution of large and ill-conditioned linear systems is only possible with iterative methods, and new ways to overcome the difficulties resulting from strong heterogeneities to solve such systems are required. In this work we explore and develop a new methodology for the solution of these problems. The technique here introduced is based on some well-known methodologies commonly used to solve reservoir simulation problems. In the next section we give a brief overview of these methods.

Solution methods for linear systems

Iterative techniques are preferred over direct methods to approximate the solution of ill-conditioned and large sparse linear systems, and the Conjugate Gradient (CG) method preconditioned with the Incomplete Cholesky (IC) factorization is a popular choice to solve Symmetric Positive Definite (SPD) systems, that usually appear in reservoir simulation. In this section we present this method together with some acceleration techniques.

The Conjugate Gradient (CG) method (der Vorst and Dekker, 1988; Kahl and Rittich, 2017), is a Krylov subspace method (der Vorst, 2003; Golub and Loan, 1996) used for SPD matrices, that searches for approximate solutions, $\mathbf{x}_k$, in directions that minimize the error, $e_k = ||\mathbf{x}-\mathbf{x}_k||_\mathbf{A}$, in the $\mathbf{A}-$norm, being $\mathbf{x}$ the true solution. The implementation of this method is given in Algorithm 1, and an upper bound for the error is presented in Equation (1).

Algorithm 1 Conjugate Gradient (CG) method, solving $\mathbf{Ax} = \mathbf{b}$.

Give an initial guess $\mathbf{x}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, and set $\mathbf{p}_0 = \mathbf{r}_0$.
for $k = 0, \ldots,$ until convergence **do**
  $\mathbf{w}_k = \mathbf{Ap}_k$
  $\alpha_k = \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{w}_k, \mathbf{p}_k)}$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
  $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{w}_k$
  $\beta_k = \frac{(\mathbf{r}_{k+1}, \mathbf{r}_{k+1})}{(\mathbf{r}_k, \mathbf{r}_k)}$
  $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
end for

$$||\mathbf{x}-\mathbf{x}_{k+1}||_\mathbf{A} \leq 2||\mathbf{x}-\mathbf{x}_0||_\mathbf{A} \left(\frac{\sqrt{\kappa_2(\mathbf{A})}-1}{\sqrt{\kappa_2(\mathbf{A})}+1} \right)^{k+1}, \qquad \kappa_2(\mathbf{A}) = \frac{\lambda_{max}}{\lambda_{min}}. \tag{1}$$

The convergence of the CG method is related to the condition number, $\kappa_2(\mathbf{A})$, see Equation (1), which depends on the eigenvalues of the system matrix. Therefore, reducing the condition number of the matrix

A results in a better performance. As the condition number of an SPD matrix is related to the largest and smallest eigenvalues of the matrix, a reduction can be obtained by clustering the spectrum, i.e., by putting together the extreme eigenvalues, or by removing them from the spectrum. In the next section, we introduce some acceleration techniques here implemented to attain this reduction. When iterative methods do not achieve convergence in a reasonable amount of time, acceleration of these methods is necessary. In this section, we present some basic information on preconditioners, together with a description of the deflation method, which are the foundations for the acceleration methods implemented in this work.

Preconditioning. Iterative methods can be accelerated by modifying the spectrum of the system, $\sigma(\mathbf{A})$. With preconditioning strategies, the original system is multiplied by a matrix $\mathbf{M}^{-1}$ that clusters the spectrum and reduces κ accordingly (Aubry et al., 2008; Saad et al., 2000).

$$\kappa(\mathbf{M}^{-1}\mathbf{A}) = \frac{\lambda_{max}(\mathbf{M}^{-1}\mathbf{A})}{\lambda_{min}(\mathbf{M}^{-1}\mathbf{A})} \qquad < \qquad \kappa(\mathbf{A}). \tag{2}$$

For this methods to be effective, the matrix $\mathbf{M}$ should approximate $\mathbf{A}$, and $\mathbf{M}^{-1}$ must be cheap to compute, and the resulting preconditioned system,

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}, \tag{3}$$

should have the same solution as the original system.

Some common choices of preconditioners are based on the LU factorization, $\mathbf{M} = \mathbf{L}\mathbf{U}$ which, for SPD systems, becomes the Incomplete Cholesky (IC) factorization $\mathbf{M} = \mathbf{L}_0\mathbf{L}_0^T$. For the CG method, IC is commonly used, and the convergence bound of the preconditioned system is given by:

$$||\mathbf{x} - \mathbf{x}_{k+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}_0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{M}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{M}^{-1}\mathbf{A})} + 1} \right)^{k+1}. \tag{4}$$

Even when the spectrum of a preconditioned system is more favorable, a few eigenvalues can, nonetheless, spoil the performance of the iterative method. Deflation techniques involve a further reduction in the condition number. This reduction is attained by removing some of the extreme eigenvalues hampering the solver's convergence by making use of available system information.

Deflation. To apply the deflation methodology, we need to find a set of *deflation* or *projection* vectors, that will be used for the construction of the *deflation-subspace matrix* $\mathbf{Z} \in \mathbb{R}^{n\times m}$, such that, the effect of extreme eigenvalues on the convergence of an iterative method is annihilated (Vuik et al., 1999; Löhner et al., 2011). Given an SPD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, the deflation matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$ is defined as follows (Tang, 2008; Tang et al., 2009):

$$\mathbf{P} = \mathbf{I} - \mathbf{A}\mathbf{Q}, \qquad \mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T, \qquad \mathbf{E} = \mathbf{Z}^T\mathbf{A}\mathbf{Z}, \tag{5}$$

where, the invertible matrix $\mathbf{E}$ is known as the *Galerkin* or *coarse* matrix. To obtain the solution of a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with deflation procedures, it is necessary to solve the deflated system: $\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}$ for the deflated solution $\hat{\mathbf{x}}$, which is related to the original system solution $\mathbf{x}$ as (Tang, 2008; Diaz-Cortes et al., 2018):

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

The deflated linear system can also be preconditioned by an SPD matrix $\mathbf{M}$, and the error of this preconditioned deflated system for the k^{th} iteration is bounded by:

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

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

The main challenge of deflation methodologies is to select a set of vectors that effectively capture system information and reduce the condition number efficiently. Our proposal is to base the selection of the vectors on POD basis vectors. To better understand this method, a brief introduction to POD is presented in the next section, followed by some state-of-the-art applications of this method and deflation techniques, together with some common choices of deflation vectors. Finally we give a detailed description of the methodology here introduced.

Proper Orthogonal Decomposition (POD). With the POD method, a high-order model is projected onto a space spanned by a small set of orthonormal basis vectors. These basis vectors, $\{\psi_j\}_{j=1}^p$, are p eigenvectors corresponding to the largest eigenvalues $\{\sigma_j\}_{j=1}^p$ of the data snapshot correlation matrix $\mathbf{R}$.

$$\mathbf{R} := \frac{1}{p}\mathbf{X}\mathbf{X}^T \equiv \frac{1}{p}\sum_{i=1}^{p}\mathbf{x}_i\mathbf{x}_i^T, \qquad \mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_p], \tag{6}$$

where $\mathbf{X} \in \mathbb{R}^{n\times p}$ is the matrix containing a series of m previously obtained snapshots. The p eigenvectors contain almost all the variability of the snapshots. Usually, they are chosen as the eigenvectors of the maximal number, p, of eigenvalues, σ, satisfying:

$$\frac{\sum_{j=1}^{p}\sigma_j}{\sum_{j=1}^{p}\sigma_j} \leq \alpha, \qquad 0 < \alpha \leq 1^1, \tag{7}$$

with α close to 1 (Markovinović and Jansen, 2006).

Sometimes, the snapshots are averaged, $\mathbf{x}_{av} = \frac{1}{p}\sum_{j=1}^{p}\mathbf{x}_j$, and the average is substracted from the original set of snapshots, $\bar{\mathbf{x}}_j = \mathbf{x}_j - \mathbf{x}_{av}$. This results in the covariance matrix

$$\bar{\mathbf{R}} := \frac{1}{p}\bar{\mathbf{X}}\bar{\mathbf{X}}^T \equiv \frac{1}{p}\sum_{i=1}^{p}\bar{\mathbf{x}}_i\bar{\mathbf{x}}_i^T, \qquad \bar{\mathbf{X}} := [\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ...\bar{\mathbf{x}}_p], \tag{8}$$

which is the matrix used throughout this work.

POD-based deflation method. In this work, we develop a methodology that combines deflation techniques with POD to further accelerate the solution of iterative methods, exploiting in this way the main advantages of both methods: with POD we obtain the most relevant system information in a basis used as a subspace deflation matrix, $\mathbf{Z}$, to annihilate the effect of extreme eigenvalues in a deflation procedure.

To obtain the basis, a set of snapshots has to be collected; the acquisition of the snapshots is done using two different schemes:

i. Moving window approach: the snapshots are captured 'on-the-fly', i.e., they are the solutions of

[1] Here σ_j are ordered from large (σ_1) to small (σ_p).

the system obtained during the previous time steps. The first p time steps are computed with the CG method preconditioned with IC, and with these snapshots, the POD basis is obtained and used as deflation-subspace matrix to compute the next solutions with the deflated preconditioned CG method (DICCG). The basis is updated at every time step. The pseudo-code is presented in Algorithm 3.

ii. Training phase approach: a full simulation is run, where the right-hand sides (rhs) are randomly varied by changing the pressure in the production wells, and the solutions of this simulation, obtained with the ICCG method, are used as snapshots to compute the POD basis. This basis is used as deflation-subspace matrix to solve diverse problems with similar characteristics, but fixed well pressures. The pseudocode is given in Algorithm 4.

Algorithm 2 Computing the POD basis from a set of snapshots.

Given a set of snapshots,

$$\mathbf{X}_{1:p} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p\}, \qquad \text{where, } \mathbf{X}_{a:b} := \{\mathbf{x}_a, \mathbf{x}_{a+1}, ..., \mathbf{x}_b\}$$

compute the average value of the set,

$$\mathbf{x}_{av} = \frac{1}{p}\sum_{j=1}^{p} \mathbf{x}_j$$

substract the average from the original set,

$$\bar{\mathbf{x}}_j = \mathbf{x}_j - \mathbf{x}_{av}$$

construct the covariance matrix,

$$\bar{\mathbf{R}} := \frac{1}{p}\bar{\mathbf{X}}\bar{\mathbf{X}}^T, \qquad \bar{\mathbf{X}}_{1:p} = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_p\}$$

obtain the SVD of the covariance matrix,

$$\mathbf{V}\mathbf{D}\mathbf{V}^T = \bar{\mathbf{R}}$$

construct the basis (Ψ) with the eigenvectors ($\mathbf{v}_i \in \mathbf{V}$) corresponding to the p largest eigenvalues ($\lambda_i \in diag(\mathbf{D})$),

$$\Psi_{1:p} = \{\mathbf{v}_1, ..., \mathbf{v}_p\}.$$

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

Compute the solution of the first p time steps with ICCG.
for $t = 1, \quad ..., \quad p$ **do**
  $\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}_t$
end for
Compute the POD basis from collected the snapshots, Algorithm 2.
  $\mathbf{Z} = \Psi_{1:p} = [\mathbf{v}_1, ..., \mathbf{v}_p]$
Compute the solution of the remaining time steps with DICCG.
for $t = p+1, \quad ..., \quad$ steps **do**
  $\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}_t$
  Update the POD basis with the recently computed solution, Algorithm 2.
    $\mathbf{Z} = \Psi_{t-p+1:t} = [\mathbf{v}_{t-p+1}, ..., \mathbf{v}_p]$
end for

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

Run a training phase simulation with ICCG varying randomly the producer's pressure in a range $[p_1, p_2]$, i.e., $\mathbf{b}_t \in [b_1, b_2]$,
for $t = 1, \quad ..., \quad steps$ **do**
$\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}_t, \qquad \mathbf{b}_t = rand, \quad rand \in [b_1, b_2]$
end for
$\mathbf{X}_{1:steps} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{steps}\}$
Compute the POD basis from the snapshots, Algorithm 2,
$\mathbf{Z} = \Psi_{1:p} = \{\mathbf{v}_1, ..., \mathbf{v}_p\}$
Run various simulation with fixed pressure in the wells using DICCG,
for $t = 1, \quad ..., \quad$ steps **do**
$\mathbf{x}_t = \mathbf{A}_t^{-1}\mathbf{b}_t$
end for

We study acceleration of linear systems resulting from simulation of water flooding in highly heterogeneous porous media. To achieve this acceleration, we use the POD-based deflation methodology, with the above-mentioned approaches. We analyze the performance of the method for diverse cases, including 2D and 3D problems, gravity and capillary pressure terms. These examples are presented in the following section.

Experiments

In this section, we present a series of experiments were we test the deflation method with a POD basis as *subspace-deflation matrix*. We study water flooding problems in a highly heterogeneous 2D and 3D reservoirs for immiscible fluids (oil and water). For the 3D case, we include gravity terms.

Model problem. We model water flooding in a reservoir initially filled with oil. The test cases are an academic layered problem and the SPE 10 benchmark (Christie and Blunt, 2001), presenting a contrast in permeability coefficients up to $\mathcal{O}(10^7)$.

We focus on the solution of systems of linear equations for the pressure, resulting from the discretization of the governing partial differential equations. We use the fractional flow formulation to decouple pressure from saturation and we solve the resulting system with sequential schemes. To obtain the linear pressure system and to solve the transport equation we use the Matlab Reservoir Simulation Toolbox (MRST, Lie (2013)) .

Transport solver. We use an implicit transport solver combined with an aggregation-based algebraic multigrid (AGMG) method (Notay, 2010; Napov and Notay, 2012; Notay, 2012) implemented in MRST to solve the transport equation.

Pressure solver. For the solution of the linear pressure equation, we implement the Deflated Preconditioned Conjugate Gradient method, with IC as preconditioner (DICCG). A POD basis is used as deflation-subspace matrix. We compare the results with the non-deflated method, ICCG. For each iteration, the computational cost to solve one time step using the ICCG method is $31n$ for a 2D case, and $39n$ for a 3D problem of size n. The required flops for the DICCG method using p deflation vectors is $(31+4p)n$ for 2D and $(39+4p)n$ for 3D cases. This implies that the DICCG method requires $\sim 1+\frac{4p}{30}$ of ICCG operations for the 2D case, and $\sim 1+\frac{p}{10}$ for the 3D case (see Diaz-Cortes et al. (2017)). In Table 1 we present the extra work per iteration for the DICCG method when compared with the ICCG for various number of deflation vectors p.

Table 1 *Extra work per iteration.*

Dimension	Extra work	$p=5$	$p=10$	$p=20$
2D	$\frac{4d}{30}$	0.7	1.3	2.6
3D	$\frac{d}{10}$	0.5	1	2

Deflation procedures. The deflation-subspace matrix $\mathbf{Z}$ consists of a POD basis obtained using two different approaches: a moving window or a training phase.

Moving window: In this approach, we start with computing a set of p snapshots to obtain a POD basis, used later as deflation vectors to 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 (see Algorithm 3).

Training simulation: For this case, we use a training phase, where we run the simulation for all the time steps with the ICCG method, randomly varying 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 the 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 (see Algorithm 4). For more details of the POD-based deflation method we refer to Diaz-Cortes et al. (2016, 2017, 2018).

As 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 rhs of the preconditioned system,

$$\frac{||\mathbf{M}^{-1}\mathbf{r}^k||_2}{||\mathbf{M}^{-1}\mathbf{b}||_2} \leq \varepsilon.$$

The tolerance of the solvers is presented for each problem.

Results and discussion

Heterogeneous permeability layers We study water injection through the left boundary of an academic system consisting of equal-sized layers with a constant porosity field of 0.2 and different permeability values (see Figure 1). A set of layers with permeability $\kappa_1 = 1\ mD$ is followed by layers with permeability $\kappa_2 = 10^1$ or $10^6\ mD$. The domain consists of a Cartesian grid of 35 x 35 cells, and the layers are placed in the x and y direction. For the relative permeability, we use the Corey model, with exponents $n_w = n_{nw} = 2$ (see Table 2). The first set of experiments does not consider capillary pressure terms. We study two cases with different accuracy for the solution methods: $\varepsilon = 5 \cdot 10^{-4}$, and $\varepsilon = 5 \cdot 10^{-7}$.

Water is injected through the left boundary at a rate of 4 m^3/day, into a reservoir with an initial pressure of 100 bars inside the reservoir and zero at the right boundary (See Table 3). The simulation is run during 300 time steps with a time step of 5 days (see Table 3). The moving window approach is used to construct the deflation subspace matrix, consisting of five or ten deflation vectors.

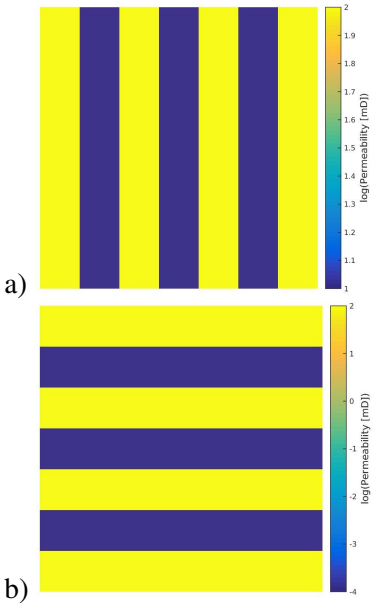


Figure 1 *Rock permeability, layers a) y direction, b) x direction.*

The pressure field and the water saturation are presented in Figure 2 and Figure 3 for both directions, x and y, and for different

Table 2 *Fluids properties.*

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

Table 3 *Boundary conditions and temporal parameters.*

Temporal parameters			Boundary conditions		
T_{steps}	300		$P_{0,x\neq(0,L_x)}$	100	$bars$
dT	5	days	$P_{x=L_x}$	0	$bars$
T_{total}	1500	days	$Q_{x=0}$	4	m^3/day

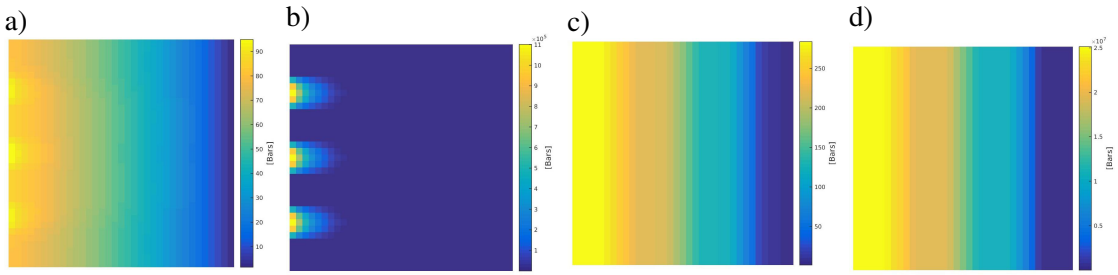


Figure 2 *Pressure field for the last time step, contrast between permeability values of a)* 10^1*, b)* 10^6*, layers in the x−direction, c)* 10^1*, d)* 10^6*, layers in the y−direction.*

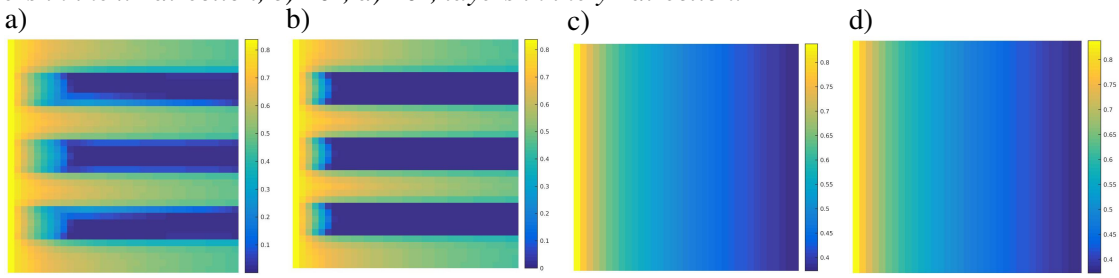


Figure 3 *Water saturation during the last time step, contrast between permeability values of a)* 10^1*, b)* 10^6*, layers in the x−direction, c)* 10^1*, d)* 10^6*, layers y−direction.*

contrast between permeability layers during the last time step. We observe that the pressure is higher at the boundary, where water is injected and it decreases towards the right boundary.

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

We observe an important reduction in the number of iterations when using the DICCG method, and this reduction appears to be larger for smaller tolerances, except for the case with layers in the x direction and a contrast of 10^6; however, in this case, the number of ICCG iterations is already small, and therefore not much gain can be achieved. We note that the results are slightly better for layers aligned in the y direction and a tolerance of $\varepsilon = 5\cdot 10^{-4}$, for which the largest gain is achieved when the contrast between the permeability layers is 10^6. For this case, the work required with the ICCG method is the largest of all the cases, requiring 24255 iterations; meanwhile, the DICCG requires around 2000 iterations, i.e., fewer than 10%.

Table 4 *Number of iterations for the layered problem with injection through the left boundary and a tolerance of* $\varepsilon = 5 \cdot 10^{-7}$.

$\frac{\kappa_2}{\kappa_1}$	p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
Layers *x* direction						
10^1	10	16408	438	3428	3866	24
10^1	5	16408	438	3573	4011	24
10^6	10	7228	147	1476	1623	22
10^6	5	7228	147	1810	1957	27
Layers *y* direction						
10^1	10	15965	475	2371	2846	18
10^1	5	15965	475	2914	3389	21
10^6	10	28621	233	4174	4407	15
10^6	5	28621	233	4119	4352	15

Table 5 *Number of iterations for the layered problem with injection through the left boundary and a tolerance of* $\varepsilon = 5 \cdot 10^{-4}$.

$\frac{\kappa_2}{\kappa_1}$	p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
Layers *x* direction						
10^1	10	6042	119	810	929	15
10^1	5	6042	119	965	1084	18
10^6	10	996	64	356	420	42
10^6	5	996	64	379	443	44
Layers *y* direction						
10^1	10	9004	87	937	1024	11
10^1	5	9004	87	1093	1180	13
10^6	10	24255	116	1631	1747	7
10^6	5	24255	116	1891	2007	8

The eigenvalues of the covariance matrix are presented in Figure 4 for all the cases during the 100-th time step. We note that they are similar; however, for the layers in the direction *x*, they are slightly larger, resulting in a small increase in the number of DICCG iterations with respect to the other cases. The results are comparable when using 5 and 10 deflation vectors, which implies that most of the system's variability is contained in the first 5 vectors. In Figure 4, we note that the first five eigenvalues are significantly larger than the rest.

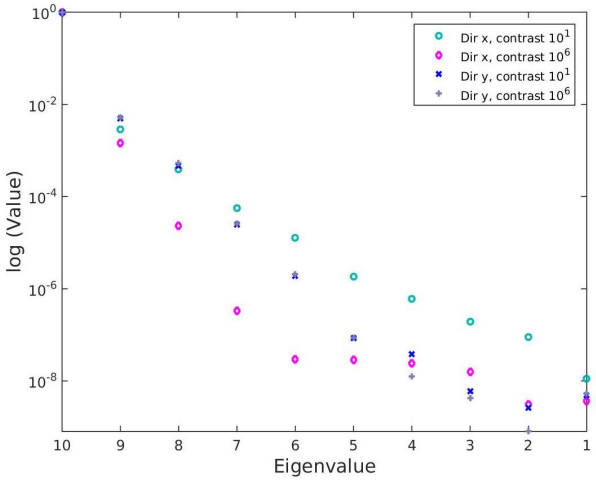


Figure 4 *Normalized eigenvalues of the covariance matrix* $\mathbf{R} = \frac{1}{p}\mathbf{X}\mathbf{X}^T$ *for the different layered problems.*

Homogeneous reservoir with gravity driven flow. For this set of experiments, we simulate the flow in a reservoir containing water in the top part and oil in the bottom part. The water saturation is presented in Figure 5 for the initial, an intermediate and the last time steps. We observe that the water is at the top of the reservoir at the beginning of the simulation, whereas at the end, it has completely gone to the bottom.

We model a reservoir with a porosity field of 1, and a permeability of 0.1 [D]; this example is taken from MRST (Lie, 2013). The reservoir contains 20 x 20 x 40 cells, 2 m long in the x- and y-directions and 1, 2 and 4 meters long in the z-direction. The simulation is run for 800 steps, with time steps of 75 days. The solution is obtained with the DICCG method, using ten and five POD basis vectors as deflation vectors,

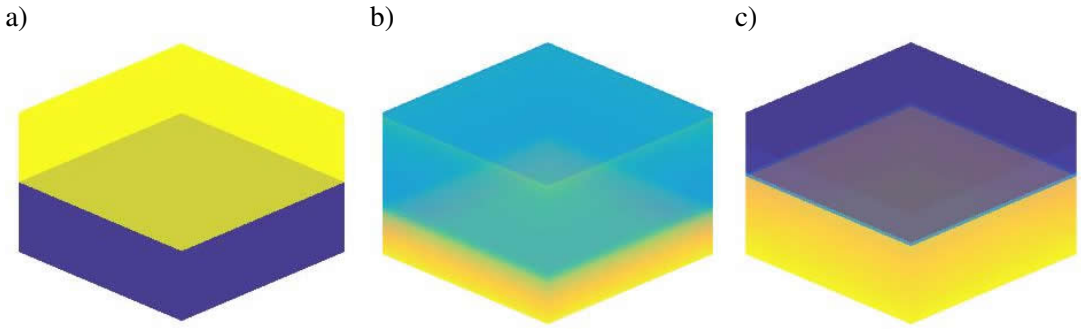


Figure 5 *Water saturation during the last time step for gravity driven experiment, s) 1st time step, b) 400-th time step, and c) 800-th time step.*

and a moving window approach. We study examples with two tolerances $5 \cdot 10^{-7}$, and $5 \cdot 10^{-4}$.

The eigenvalues of the covariance matrix are presented in Figure 6 for all the cases during the 200-th time step. From this plot, we can observe that they are similar; however, as the reservoir height increases, the eigenvalues become slightly smaller.

The number of iterations required to achieve convergence is presented in Table 6 for a stopping criterion of $\varepsilon = 5 \cdot 10^{-7}$, and Table 7 for a stopping criterion of $\varepsilon = 5 \cdot 10^{-4}$.

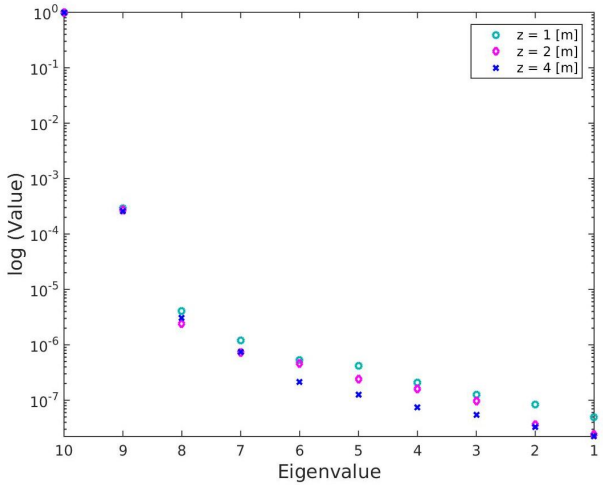


Figure 6 *Normalized eigenvalues of the covariance matrix* $\mathbf{R} = \frac{1}{p}\mathbf{X}\mathbf{X}^T$*, gravity problem, various reservoir heights.*

We observe an important reduction in the number of iterations with the DICCG method; for the cells with the smallest reservoir height, 1 meter, the number of iterations increases, requiring 20% and 26% of the ICCG iterations for ten and five deflation vectors. If the high of the cells is 2 meters, the reduction is 14% for ten deflation vectors, and 20% for five. For cells 4 of meters high, we achieve a reduction of 10% of the number of ICCG iterations if we use ten deflation vectors, and 13% if we use five. Therefore, according to the results, the performance improves for higher reservoirs.

If we compare the eigenvalues of the covariance matrix with the performance of the DICCG method, we note a better performance when the eigenvalues are smaller, which is the case for the reservoir with highest cells (4 [m]). We note that using a larger tolerance, $\varepsilon = 5 \cdot 10^{-4}$, the performance is apparently worse; however, from Figure 7 we note that even if the residual is smaller than 10^{-4} for the ICCG

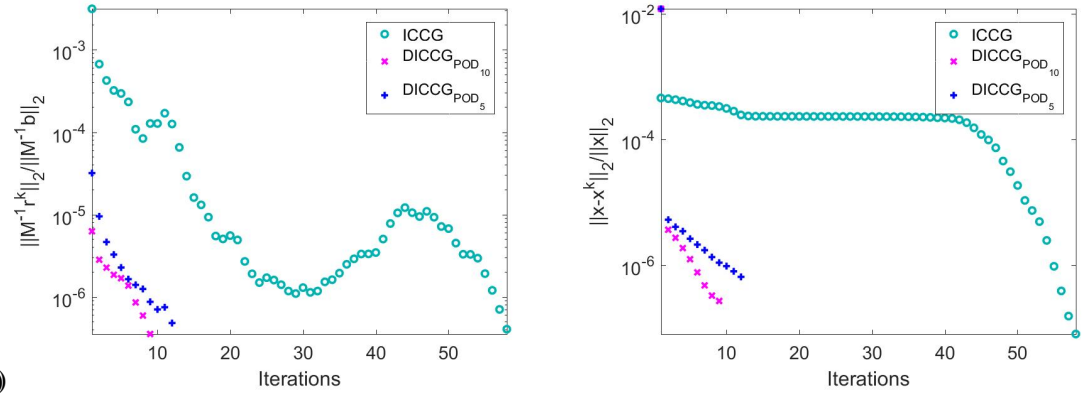
Table 6 *Number of iterations for the gravity column example, tolerance of* $\varepsilon = 5 \cdot 10^{-7}$.

p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
Size of the z cells: 1 [m]					
10	25659	529	4506	5035	20
5	25659	529	6127	6656	26
Size of the z cells: 2 [m]					
10	36835	530	4665	5195	14
5	36835	530	6843	7373	20
Size of the z cells: 4 [m]					
10	45043	664	3974	4638	10
5	45043	664	5257	5921	13

Table 7 *Number of iterations for the gravity column example, tolerance of* $\varepsilon = 5 \cdot 10^{-4}$.

p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
Size of the z cells: 1 [m]					
10	2642	146	1394	1540	58
5	2642	146	1494	1640	62
Size of the z cells: 2 [m]					
10	4013	177	1436	1613	40
5	4013	177	1720	1897	47
Size of the z cells: 4 [m]					
10	3808	146	1582	1728	45
5	3808	146	1701	1847	49

method after the first iterations, the true error is still large. The first iteration obtained with the DICCG method has a true error smaller than 10^{-4}, but the ICCG method achieves this accuracy only after around 40 iterations, this implies that the results presented in Table 7 do not represent the selected accuracy for ICCG, and the comparison is not proper.



a)

Figure 7 *Relative residual and true relative error for various methods, gravity driven flow.*

These results show that the POD-based deflation method is able to capture information from a gravity-driven flow and accelerate the solution with this information; however, we also note that the size of the reservoir influences the performance of the method, where the optimal performance is achieved for larger cells. Furthermore, the DICCG method give accurate results for the given tolerance; whereas, the ICCG method does not.

SPE 10

In this section, we perform a series of experiments using the SPE 10 model, injecting and producing fluids through wells. The permeability field of this model and the position of the wells is presented in Figure 8. We study a series of cases with the upper layer (60 x 220 cells), and we test the full model (85 layers). For these examples, the POD basis and deflation matrix are obtained offline in a training phase run with the ICCG method. Once the basis is 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 2 time steps during the training phase between $P = 137.5$ and $P = 275$ bars, (see Figure 9). The solutions of this simulation are use to construct the covariance matrix and to compute the POD basis. Five and ten basis vectors are used as deflation vectors to compute the solution of slightly different problems using the DICCG method.

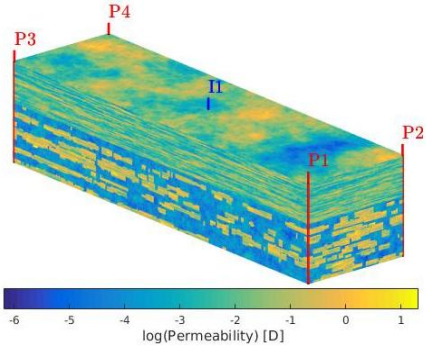


Figure 8 *Permeability field SPE 10.*

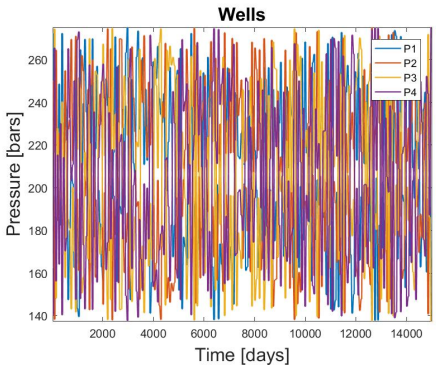


Figure 9 *Well pressures in the production wells during the training run.*

The pressure in the injection well is maintained constant at $I = 1100$ bars for all cases, and the initial reservoir pressure is 500 bars. For the examples solved with DICCG, we use different pressures in the producers. The first set of experiments has the same fixed pressure in all the producers, for the second set, the pressure in one production well is different. The experiments performed are the following:
Equal bottom hole pressure in the producers. For the first experiment, the bhp is $P = 275$ bars in all the producers, an extreme value of the training phase run. For the second experiment, it is $P = 200$ bars, a value inside the pressure range of the training phase. The final experiment has a pressure of $P = 400$ bars, outside the training phase pressure range.
Different bottom hole pressure in the producers. One well has a bhp of $P_i = 20$ bars, and the rest have the same pressure as the reservoir $P_{j\neq i} = 500$ bars.

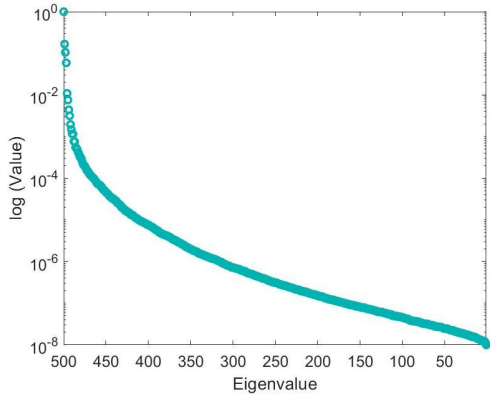


Figure 10 *Eigenvalues of the covariance matrix.*

The simulation is run during 500 time steps, with a step of 100 days until water reaches all the wells. The oil production and water breakthrough for each well are presented in Figure 11 for the case when pressure is the same for all producers $P = 200$ bars. We note that the fourth well P_4 is the one that produces more oil, followed by P_2 and P_3, finalizing with P_1; we also observe that water reaches the wells in the same order.

The eigenvalues of the covariance matrix obtained with the training run are presented in Figure 10. We observe that only a few of them are larger than 10^{-4}, which indicates that large part of the system information is contained in these eigenvalues.

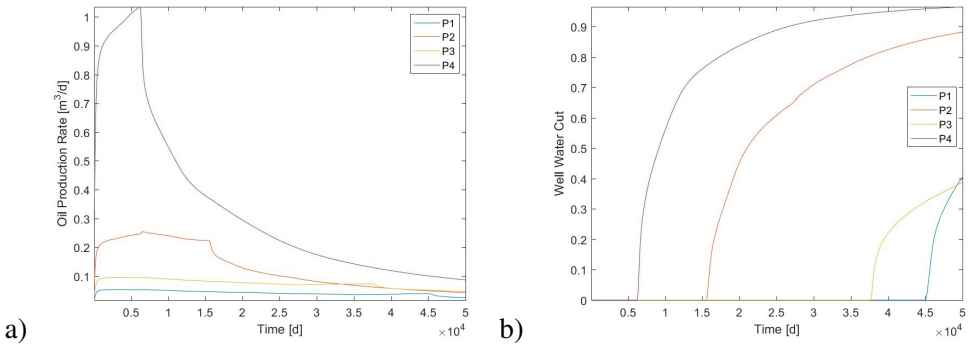


Figure 11 *Oil production and water breakthrough, waterflooding in the the first layer of the SPE 10 benchmark.*

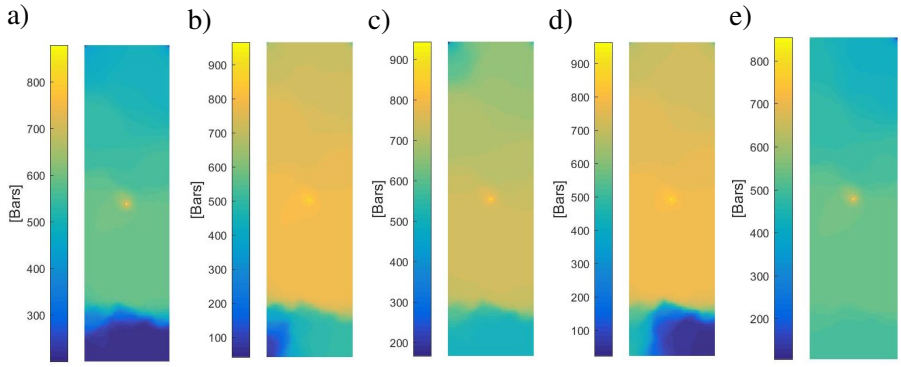


Figure 12 *Pressure field for the last time step for the first layer of the SPE 10 benchmark, various bhp pressures in the wells a)* $P_{2,3,4} = 500$ *bars,* $P_1 = 20$ *bars, b)* $P_{1,3,4} = 500$ *bars,* $P_2 = 20$ *bars, c)* $P_{1,2,4} = 500$ *bars,* $P_3 = 20$ *bars, d)*$P_{1,2,3} = 500$ *bars,* $P_4 = 20$ *bars.*

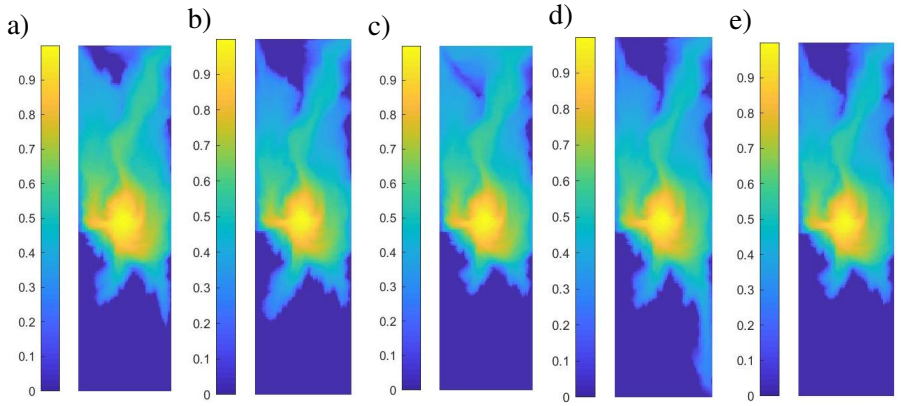


Figure 13 *Water saturation during the 200-th time step for the first layer of the SPE 10 benchmark, various bhp pressures in the wells a)* $P_{2,3,4} = 500$ *bars,* $P_1 = 20$ *bars, b)* $P_{1,3,4} = 500$ *bars,* $P_2 = 20$ *bars, c)* $P_{1,2,4} = 500$ *bars,* $P_3 = 20$ *bars, d)*$P_{1,2,3} = 500$ *bars,* $P_4 = 20$ *bars.*

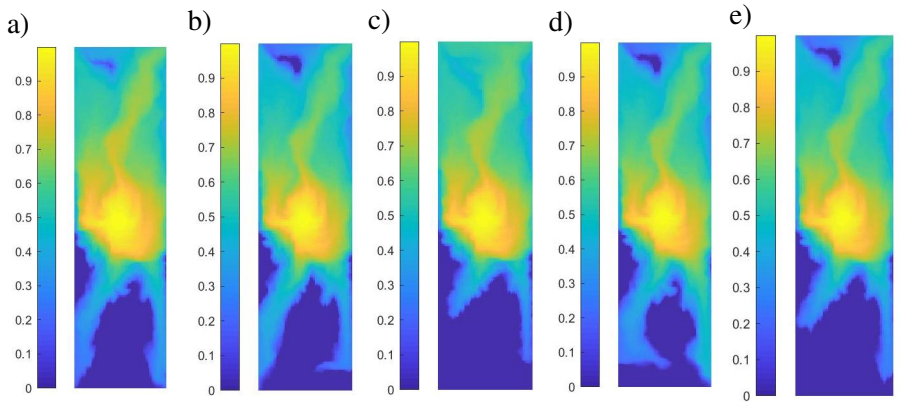


Figure 14 *Water saturation during the last time step for the first layer of the SPE 10 benchmark, various bhp pressures in the wells a)* $P_{2,3,4} = 500$ *bars,* $P_1 = 20$ *bars, b)* $P_{1,3,4} = 500$ *bars,* $P_2 = 20$ *bars, c)* $P_{1,2,4} = 500$ *bars,* $P_3 = 20$ *bars, d)*$P_{1,2,3} = 500$ *bars,* $P_4 = 20$ *bars.*

The pressure and saturation fields for the last time step are shown in Figure 12 and Figure 14, while Figure 13 displays the results for the 200-th time step for various cases. The first plot, a), corresponds to the experiment where the pressure in the production wells is the same $P = 200$ bars, b)-e) all the wells have a bhp of $P_{j\neq i} = 500$ bars, except for one with bhp $P_i = 20$ bars. We observe that the water flow patterns are different for each case, as are the pressure fields.

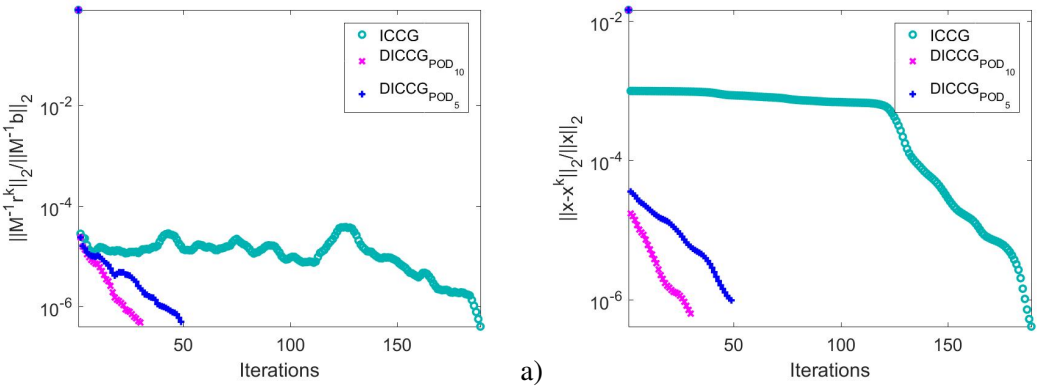
The resulting number of iterations for the ICCG and DICCG methods is presented in Table 8 for diverse wells configurations. We observe that the number of iterations does not change considerably for the different cases. For all the cases, an important reduction in the number of ICCG iterations is obtained, being around 15% for the case with 10 deflation vectors, and around 22% for the case with 5 deflation vectors.

1 layer			
Total ICCG	DICCG Method	Iter	% of ICCG Iter
P_{bhp} = 275 [bars]			
10	90130	12975	14
5	90130	20514	23
P_{bhp} = 200 [bars]			
10	90130	13720	15
5	90130	21522	24
P_{bhp} = 400 [bars]			
10	90130	11374	13
5	90130	18552	21

1 layer, various pressures in producers			
Total ICCG	DICCG Method	Iter	% of ICCG Iter
$P_{2,3,4}$ = 200 [bars], P_1 = 20 [bars]			
10	90130	11740	13
5	90130	17855	20
$P_{1,3,4}$ = 200 [bars], P_2 = 20 [bars]			
10	90130	10518	12
5	90130	20926	23
$P_{1,2,4}$ = 200 [bars], P_3 = 20 [bars]			
10	90130	10518	12
5	90130	17325	19
$P_{1,2,3}$ = 200 [bars], P_4 = 20 [bars]			
10	90130	11636	13
5	90130	18693	21

Table 8 *Number of ICCC and DICCG iterations for diverse bhp in the production wells, training phase approach.*

In Figure 15, we present the relative residual and the true relative error of the studied methods for the 100-th time step. We note that after the first iteration, the residual and the true solution of the deflated method are both smaller than 10^{-4}. However, for the ICCG method, the residual is around 10^{-4} for the first iterations, but true error is around 10^{-3}, i.e., the approximation is not as accurate as with the DICCG method. Therefore, the DICCG method gives a better approximation than the ICCG method.



a) a)

Figure 15 *Relative residual and true relative error for the waterflooding with a bhp of $P = 275$ bars in the wells and $I = 1100$ bars in the injector.*

85 layers, SPE 10

We model water flooding in the full SPE 10 model, consisting of 85 layers. We run the simulation during 200 time steps with a time step of 1.5 days. We use a training phase scheme randomly varying the bhp of the production wells to compute the POD basis in a range $P = 137.5 - 275$, see Figure 9. Later, we solve for three cases with different bhp in the production wells, $P = [200, 275, 400]$. The water saturation at an intermediate and final time steps are presented in Figure 16.

The number of iterations required to find an approximate solution with an accuracy of 10^{-7} for the full SPE 10 model, containing 85 layers, is presented in Table 9. We can observe a reduction to around 27% the number of ICCG iterations when using 20 deflation vectors and 31% when using 15. These results are similar for all the cases presenting diverse pressures in the producers.

d	Total ICCG	Total DICCG	% of ICCG
3D case, 60 x 220 x 85 cells			
No capillary pressure included			
P_{bhp} = 275 [bars]			
20	96468	25376	26
15	96468	29658	31
P_{bhp} = 200 [bars]			
20	96468	26730	28
15	96468	31146	32
P_{bhp} = 400 [bars]			
20	96468	26730	28
15	96468	27429	28

Table 9 *Number of iterations for the SPE 10 benchmark with injection through wells and a tolerance of $\varepsilon = 5 \cdot 10^{-7}$, training phase approach.*

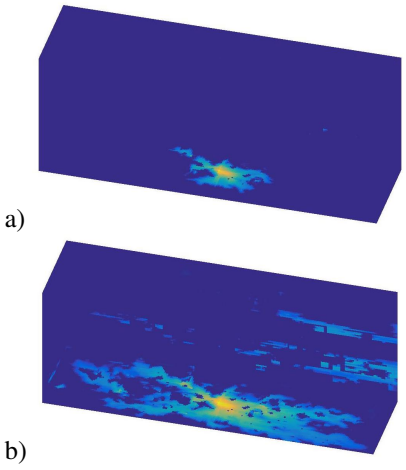


Figure 16 *Water saturation of the full SPE 10 benchmark, various time steps, a) 50, b) 200.*

SPE 10, gravity driven flow. We simulate waterflooding in a reservoir containing water in the top part and oil in the bottom part. The water saturation is presented in Figure 18 for various time steps. We model the first 10 layers of the SPE 10 benchmark. We use 20 x 20 cells for which we average the porosity and the permeability using the harmonic average function of MRST. The length of the cells in the x- and y-direction is 6 and 2 [m] and for the layers we vary the length, using 8, 10 and 12 [m]. The simulation is run during 800 steps, with a time step size of 40 days.

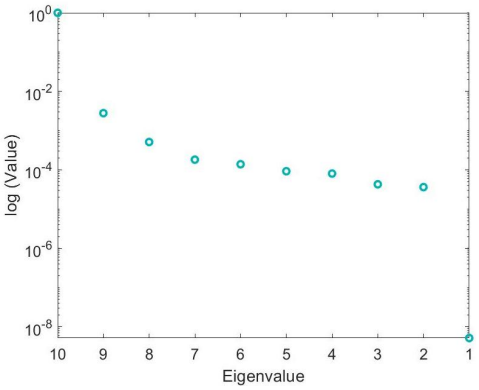


Figure 17 *Eigenvalues of the covariance matrix.*

The number of iterations required to achieve convergence is presented in Table 10 for a stopping criterion of $\varepsilon = 5 \cdot 10^{-7}$. The eigenvalues of the covariance matrix are presented in Figure 17, where we observe that the first nine eigenvalues are larger than the last one, and among them, the first four are the largest, which implies that most of the information is contained the corresponding eigenvectors.

p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
Size of the z cells: 8 [m]					
10	17603	315	2550	2865	16
5	17603	315	2823	3138	18
Size of the z cells: 10 [m]					
10	19303	304	2473	2777	14
5	19303	304	2802	3106	16
Size of the z cells: 12 [m]					
10	20570	309	2456	2765	13
5	20570	309	2647	2956	14

Table 10 *Number of iterations for the gravity column example for the SPE 10 model, tolerance of* $\varepsilon = 5 \cdot 10^{-7}$.

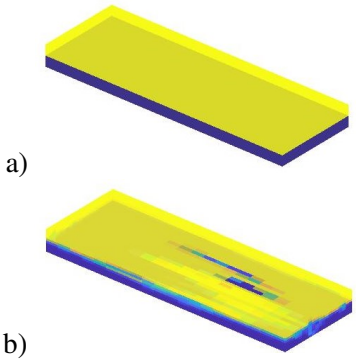
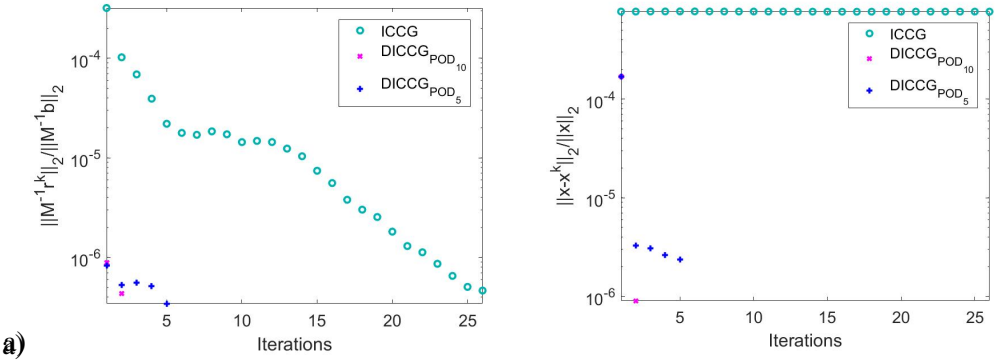


Figure 18 *Water saturation at the beginning and the end of the simulation.*



a)

Figure 19 *Relative residual and true relative error for gravity driven flow.*

In Figure 19, we present the relative residual and the true relative error for the 400-th time step. We note that after the first iteration, the residual and the true relative error of the deflated method are both smaller than 10^{-4}. However, for the ICCG method, the residual is around 10^{-7}, the required accuracy, but the true error is still large; therefore, the approximation is not accurate.

SPE 10 benchmark including capillary pressure terms. In this section we perform a set of experiments including capillary forces. We model waterflooding for the first layer of the SPE 10, with the same well configuration and reservoir properties as the previous examples of this section. We compare a case without capillary pressure with three cases presenting diverse Corey coefficients for the wetting phase, n_w =[2,3,4], and $n_{nw} = 2$. The curves are presented in Figure 22. We use a linear capillary relationship, $P_c = C(1-S)$, see Figure 20. The fluid properties are presented in Table 11. We use the training phase scheme, for which we run a simulation varying the pressure in the production wells. We run the simulation during 600 time steps, with a step size of 30 days. The pressure field and water saturation are presented in Figure 21 for all the cases at the last time step.

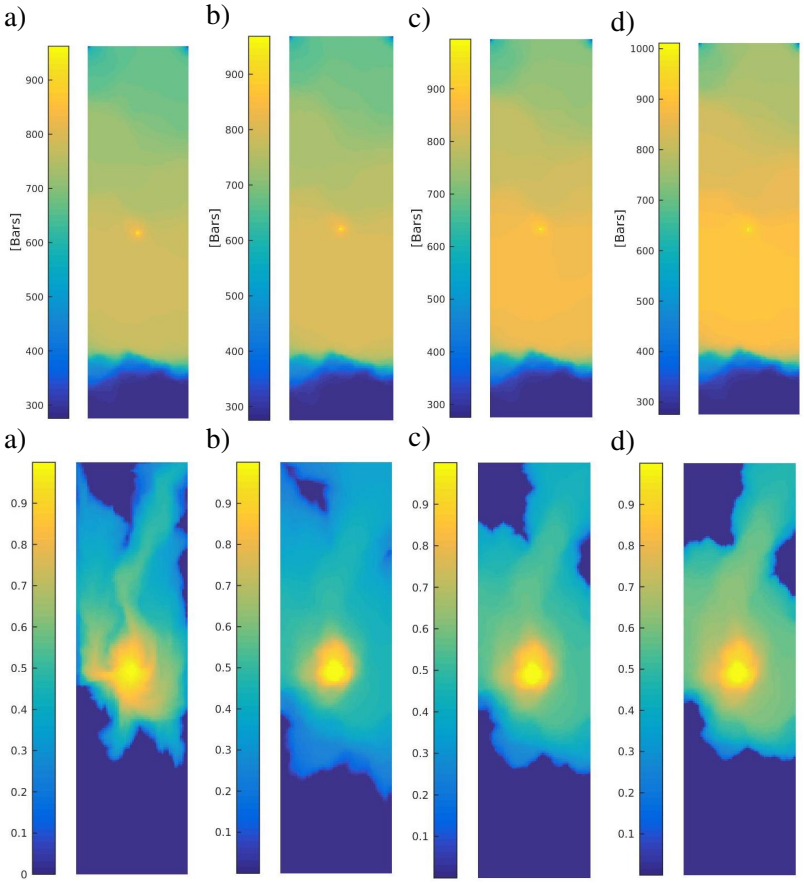


Figure 21 *Pressure field and water saturation for the last time step, first layer of the SPE 10 benchmark, a) No capillary pressure, b)*$n_w = n_{nw} = 2$*, c)* $n_w = 3$, $n_{nw} = 2$*, d)* $n_w = 4$, $n_{nw} = 2$.

	Water	Oil	Units
μ	1	10	cp
ρ	1000	700	kg/m^3
k_r	$(S_w)^2$	$(1-S_w)^2$	
k_r	$(S_w)^3$	$(1-S_w)^2$	
k_r	$(S_w)^4$	$(1-S_w)^2$	
C_p	$10*(1-S)$		bars

Table 11 *Fluids properties.*

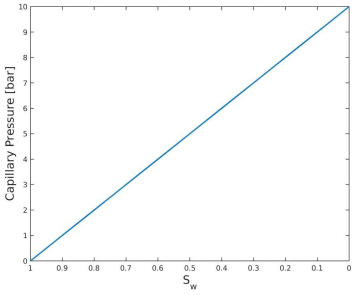


Figure 20 *Capillary pressure function* $C_p = 10 * (1 - S)$.

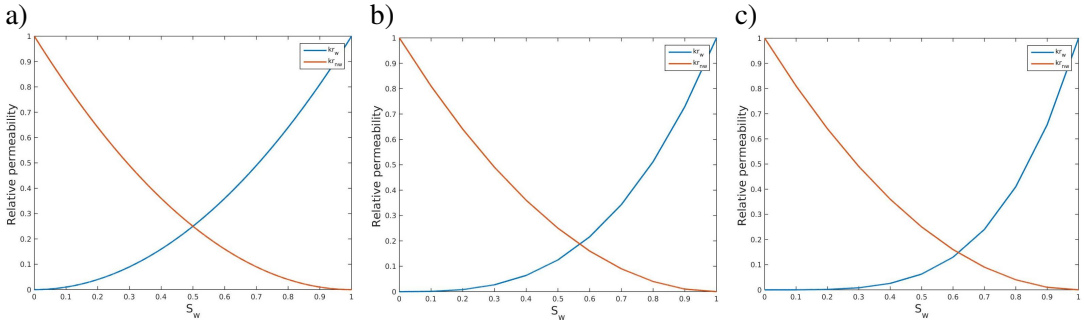


Figure 22 *Relative permeability curves, a)* $n_w = n_{nw} = 2$*, b)* $n_w = 3$, $n_{nw} = 2$*, c)* $n_w = 4$, $n_{nw} = 2$.

Table 12 shows the number of iterations required to achieve convergence for a stopping criterion of $\varepsilon = 5 \cdot 10^{-7}$. We note that we require 16% of the ICCG iterations with ten deflation vectors. For the cases with capillary pressure a reduction of the ICCG iterations; however, we also note a small increment in the number of iterations when we increase the Corey coefficient. If we observe the eigenvalues of the covariance matrix, Figure 23, we note that they are similar; however, they are smaller for the case without capillary pressure and for the smaller Corey coefficient. We also observe a few eigenvalues larger than 10^{-4}, which suggests that most of the information is contained in the corresponding eigenvectors; thus, if we use ten eigenvectors as deflation vectors, they contain enough information to achieve an important acceleration.

p	Total ICCG	Total DICCG	% of ICCG
No capillary pressure			
10	102831	16072	16
$k_{rw} = (S_w)^2$, $k_{rnw}(S_{nw})^2$			
10	103849	15087	15
$k_{rw} = (S_w)^3$, $k_{rnw}(S_{nw})^2$			
10	96614	16628	17
$k_{rw} = (S_w)^4$, $k_{rnw}(S_{nw})^2$			
10	94609	18478	20

Table 12 *Number of iterations for the gravity column example for the SPE 10 model, tolerance of* $\varepsilon = 5 \cdot 10^{-7}$.

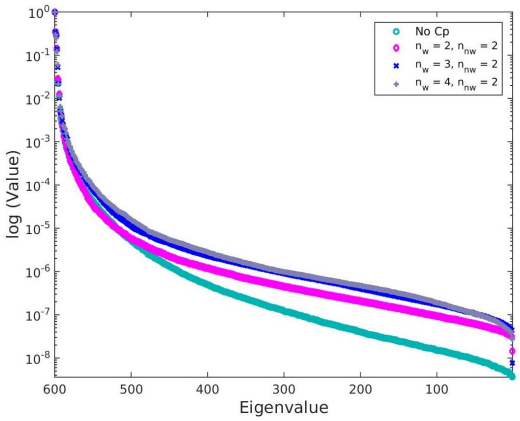
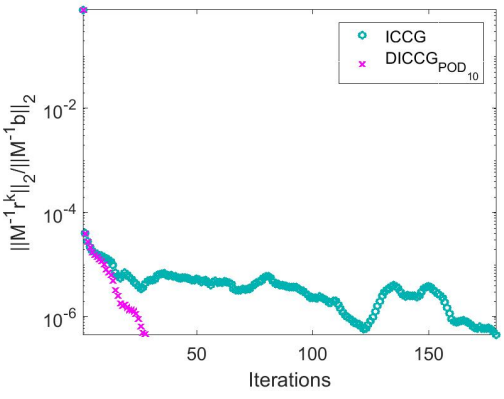
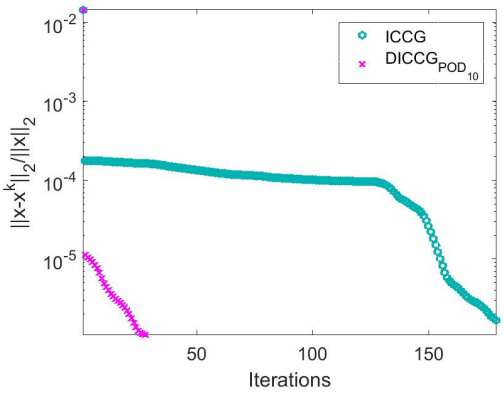


Figure 23 *Eigenvalues of the covariance matrix, diverse Corey coefficients*



a)



d)

Figure 24 *Relative residual and true relative error for ICCG and DICCG methods for a Corey coefficient of* $n_w = 3$.

In Figure 24, we plot the residual and the true error for the case with capillary pressure and Corey coefficients $n_w = 3$, and $n_{nw} = 2$. For the first iteration, we note that even if the residual is smaller than 10^{-4}, the true error is still large. Hence, if we would like to compute an approximation with this accuracy, the ICCG method will give an incorrect solution. Furthermore, the method reaches the superlinear convergence region after around 120 iterations, but it does not reach the required accuracy. By contrast, the approximation obtained with the DICCG method has a true error smaller than 10^{-5} after the first approximation; furthermore, the residual presents a similar behavior as the true error. Hence, the DICCG method for problems involving capillary pressure is also more accurate than the ICCG method.

Conclusions

The large number of cells and the high contrast in permeability coefficients make the simulation of flow through porous media an expensive process, in particular for the resulting linear pressure system. We present new possibilities to accelerate this process using POD basis vectors in a deflation procedure for the Conjugate Gradient method preconditioned with Incomplete Cholesky (DICCG).

We studied water flooding with injection of water through boundaries and through wells in an academic layered heterogeneous porous medium, and for the SPE 10 benchmark problem presenting a contrast in permeability coefficients up to 10^7. Among the test cases, we included 2D and 3D problems, the latter presenting gravity forces; furthermore, we studied cases including capillary pressure. For all the cases, we achieved important reductions in the number of iterations when compared to a standard method, Conjugate Gradient method preconditioned with incomplete Cholesky (ICCG). For the POD-based case, we collected system information with a *moving window* approach, where the POD basis is computed at each time step; and with a *training phase* approach, for which the basis is computed in a pre-simulation.

We noted a relation between the performance of the method and the spectrum of the covariance matrix obtained during the POD procedure. A large difference between a few large eigenvalues and the rest resulted in a better performance.

We observed a good performance for all the studied cases using both, moving window and training phase approaches. For the layered problem, we reduced the number of iterations to less than 10% of the number of ICCG iterations with five deflation vectors. For the full SPE 10 case, we reduced the number of iterations to $\sim 27\%$ of the ICCG iterations using 20 deflation vectors.

When using a training phase approach, the flow pattern and the pressure fluctuations on the wells do not affect significantly the performance of the method. We illustrated this behavior using as deflation vectors a POD basis obtained running a training phase with randomly varying bottom hole pressures in the production wells. We used this basis to perform various experiments with diverse well configurations resulting in various flow patterns; the performance of the method was similar in all cases, also in a case with bhp slightly outside the training phase range.

We also tested the method for more complex problems; the experiments of gravity-driven flow showed an improvement in the performance for taller reservoirs. Including capillary pressure terms did not change dramatically the performance of the method for the studied cases; however, increasing the Corey coefficients (i.e. increasing the nonlinearity of the underlying flow problem) resulted in a small increment in the number of iterations for the DICCG method.

Finally, we observed that the residual computed with the DICCG method was very close to the true error of the approximation. By contrast, while the residual of the ICCG method was small, for some cases, the true error was still large; hence, the POD-based DICCG solution resulted in a more accurate approximation. Furthermore, the first DICCG iteration resulted in a solution with a true error smaller than 10^{-4} for all the cases. The latter implies that, if the required accuracy is in this range, as usual in industry, only one DICCG iteration is necessary.

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)', who have sponsored this work.

References

Astrid, P., Papaioannou, G., Vink, J. and Jansen, J. [2011] Pressure Preconditioning Using Proper Orthogonal Decomposition. In: *2011 SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA*. 21–23.

Aubry, R., Mut, F., Löhner, R. and Cebral, J. [2008] Deflated preconditioned conjugate gradient solvers for the Pressure–Poisson equation. *Journal of Computational Physics*, **227**(24), 10196–10208.

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

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

Diaz-Cortes, G., Vuik, C. and Jansen, J. [2016] 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*. 1031–1051.

Diaz-Cortes, G., Vuik, C. and Jansen, J. [2017] 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. `http://ta.twi.tudelft.nl/nw/users/vuik/papers/Dia17VJ.pdf`.

Diaz-Cortes, G., Vuik, C. and Jansen, J. [2018] On POD-based Deflation Vectors for DPCG applied to porous media problems. *Journal of Computational and Applied Mathematics*, **330**(Supplement C), 193 – 213.

Golub, G. and Loan, C.V. [1996] *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, USA, 3rd edn.

Kahl, K. and Rittich, H. [2017] The deflated conjugate gradient method: Convergence, perturbation and accuracy. *Linear Algebra and its Applications*, **515**, 111–129.

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

Löhner, R., F., J.R. Cebral, J.R., Aubry, R. and Houzeaux, G. [2011] Deflated preconditioned conjugate gradient solvers for the pressure-Poisson equation: Extensions and improvements. *International Journal for Numerical Methods in Engineering*, **87**(1-5), 2–14.

Markovinović, R. and Jansen, J. [2006] Accelerating iterative solution methods using reduced-order models as solution predictors. *International journal for numerical methods in engineering*, **68**(5), 525–541.

Napov, A. and Notay, Y. [2012] An algebraic multigrid method with guaranteed convergence rate. *SIAM journal on scientific computing*, **34**(2), A1079–A1109.

Notay, Y. [2010] An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, **37**(6), 123–146.

Notay, Y. [2012] Aggregation-based algebraic multigrid for convection-diffusion equations. *SIAM journal on scientific computing*, **34**(4), A2288–A2316.

Pasetto, D., Ferronato, M. and Putti, M. [2017] 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.

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

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

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

Tang, J., Nabben, R., Vuik, C. and Erlangga, Y. [2009] Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *Journal of scientific computing*, **39**(3), 340–370.

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

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

Vuik, C., Segal, A. and Meijerink, J. [1999] 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.

Vuik, C., Segal, A., Yaakoubi, L. and Dufour, E. [2002] A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients. *Applied Numerical Mathematics*, **41**(1), 219–233.