DELFT UNIVERSITY OF TECHNOLOGY

REPORT 17-01

On POD-based Deflation Vectors for DPCG applied to porous media problems.

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

ISSN 1389-6520

Reports of the Delft Institute of Applied Mathematics

Delft 2017

On POD-based Deflation Vectors for DPCG applied to porous media problems.

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

February 2017

Abstract

We study fast and robust iterative solvers for large systems of linear equations resulting from simulation of flow trough strongly heterogeneous porous media. We propose the use of preconditioning and deflation techniques, based on information obtained from the system, to reduce the time spent in the solution of the linear system.
An important question when using deflation techniques is how to find good deflation vectors, which lead to a decrease in the number of iterations and a small increase in the required computing time per iteration. In this paper, we propose the use of deflation vectors based on a POD-reduced set of snapshots. We investigate convergence and the properties of the resulting methods. Finally, we illustrate these theoretical results with numerical experiments. We consider compressible and incompressible single-phase flow in a layered model with variations in the permeability layers up to 10^3 and the SPE 10 benchmark model with a contrast in permeability coefficients of 10^7. Using deflation for the incompressible problem, we reduce the number of iterations to 1 or 2 iterations. With deflation, for the compressible problem, we reduce up to $\sim 80\%$ the number of iterations when compared with the only-preconditioned solver.

1 Introduction.

Often, most computational time in the simulation of multi-phase flow through porous media is taken up by the solution of the pressure equation. This involves, primarily, solving large systems of linear equations as part of the iterative solution of the time and space discretized governing nonlinear partial differential equations. The time spent in solving the linear systems depends on the size of the problem and the heterogeneity, i.e. the spatial variations of rock permeability values within the medium (permeability is an inverse measure of the resistance to flow which is related to the porosity and the pore structure of the rock). Solution of problems with extreme contrasts in the permeability values may lead to very large computing times.

Iterative methods are known to be the best option to solve such extreme problems. However, sometimes iterative methods are not sufficient to solve these problems in a reasonable amount of time. As the systems become larger or ill-conditioned, finding a way to accelerate the convergence of these methods becomes necessary. Preconditioning is a way to accelerate convergence, but new preconditioning techniques still need to be developed to improve the performance of iterative methods [1, 2]. Reduced Order Models (ROM) have also been studied to improve computational efficiency by reducing the model size without losing essential information [3–5]. A potential ROM to reduce the computing time for large-scale problems is Proper Orthogonal Decomposition (POD), a method that has been investigated for flow problems in porous media in [6–15] among others. The use of a POD-based preconditioner for acceleration of the solution is proposed by Astrid et al. [11] to solve the pressure equation resulting from two-phase reservoir simulation, by Jiang et al. [14] for a similar application and by Pasetto et al. [15] for groundwater flow models.

The POD method requires the computation of a series of 'snapshots' which are solutions of the problem with slightly different parameters or well inputs. Astrid et al. [11] use snapshots in the form of solutions of the pressure equation computed in a small number of short pre-simulations, prior to the actual simulation, with diverse well configurations, reporting promising speed ups with factors between three and five. They note that the overhead required to pre-compute the POD solutions implies that the method will be particularly attractive when many solutions of near-similar simulation models are required. A similar approach is followed by Jiang [14], who concludes that POD-based pressure preconditioning does not appear to be an ideal choice because of its dependence on the differences between the right-hand sides (forcing terms) used in the pre-simulations and the actual simulation. The snapshots computed by Pasetto et al. [15] are solutions of the previous time steps in the full-model. Once the snapshots are computed, the POD method is used to obtain a set of basis vectors that capture the most relevant features of the system, which can be used to speed-up the subsequent simulations.

The method of Pasetto at al. [15] is partly based on the work of Markovinovic and Jansen [8] who use a similar, but more restricted, approach in which the acceleration is achieved by only improving the initial guess.

Problems with high contrast between the permeability coefficients are sometimes approached through the use of deflation techniques, see, e.g., [16]. These techniques involve

the search of good deflation vectors, which are usually problem-dependent. In [16], subdomain based deflation vectors are used for layered problems with a large contrast between permeability coefficients. However, these deflation vectors cannot be used if the distribution of the permeability coefficients is not structured, as is usually the case in reservoir simulation models; see, e.g., the well-known SPE 10 benchmark problem [17].
Algebraic Multigrid (AMG)[18], Multi-level and Domain Decomposition [19] preconditioners have been studied in combination deflation techniques to accelerate the convergence of iterative methods. In [8, 11] and [15], after computing a basis from the previously obtained snapshots, the solution is computed in the subspace generated by this basis and then projected back to the original high-dimensional system. Carlberg et al. [20] also use POD to obtain information from the system, in particular, the previous time step solutions. Then, a Krylov-subspace is constructed using the information obtained previously.
Following the ideas of [8, 11, 15, 20], we propose the use of POD of many snapshots to capture the system's behavior and combine this technique with deflation to accelerate the convergence of an iterative Krylov method. In this work, instead of computing the solution in a low dimensional subspace, the basis obtained with POD is proposed as an alternative choice for the deflation vectors to accelerate the convergence of the pressure solution in reservoir simulation.
This work is divided into six sections. Section 2 is devoted to a detailed description of the models used to simulate flow through a porous medium. In Section 3, we present some theory about the linear solvers used in this work and we introduce preconditioning and deflation techniques. In Section 4 we present some theory about POD. We prove two lemmas that will help us in the choice of good deflation vectors for the incompressible case in Section 5.
In Section 6 we present numerical experiments. We describe the problem that is studied, the solver and the preconditioning and deflation techniques used to speed up the solver. The results are also presented in this section. Finally, we end with the conclusions.

2 Flow through porous media

Petroleum reservoirs are layers of sedimentary rock, which vary in terms of their grain size and mineral contents. The volume fraction of the rock in-between the grains, i.e. the void space, is called *rock porosity*, a scalar quantity indicated with ϕ. The ability of the rock to transmit a single fluid when the void space is completely filled with fluid is known as *rock permeability*, a tensor quantity indicated with K.
Reservoir simulation is a way to analyze and predict the fluid behavior in a reservoir. The description of subsurface flow simulation involves two types of models: geological (static) and flow (dynamic) models. The static model is used to describe spatial properties of the reservoir, i.e. the porosities and permeabilities, which are parameters for the dynamic model. The dynamic model is subsequently used to predict fluid pressures and flow taking into account mass conservation and Darcy's law, an empirical, simplified version of the momentum conservation equations. The corresponding equations used to describe single-phase flow through a porous medium are (see, e.g., [21–23]) :

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho v) = q, \qquad v = -\frac{K}{\mu}(\nabla p - \rho g \nabla z), \tag{1}$$

or

$$\frac{\partial(\rho\phi)}{\partial t} - \nabla \cdot \left(\frac{\rho K}{\mu}(\nabla p - \rho g \nabla z) \right) = q, \tag{2}$$

where the pressure p is the primary unknown, g is the constant of gravity, d is the reservoir depth, ρ and μ are the fluid density and viscosity and q is a source term (i.e. an injection or production well). The fluid density $\rho = \rho(p)$ and the rock porosity $\phi = \phi(p)$ can be pressure-dependent. Rock porosity is related to the pressure via the rock compressibility. The relation is given by:

$$c_r = \frac{1}{\phi}\frac{d\phi}{dp} = \frac{d(ln(\phi))}{dp},$$

If the rock compressibility is constant, the previous equation can be integrated as:

$$\phi(p) = \phi_0 e^{c_r(p-p_0)}. \tag{3}$$

The fluid density and the pressure are related via the fluid compressibility c_f, according to:

$$c_f = \frac{1}{\rho}\frac{d\rho}{dp} = \frac{d(ln(\rho))}{dp}.$$

If the fluid compressibility is constant, the previous equation can be integrated as:

$$\rho(p) = \rho_0 e^{c_f(p-p_0)}. \tag{4}$$

To solve Equation (2), it is necessary to supply conditions at the boundary of the domain. While, for parabolic equations, we also need to impose initial conditions. Boundary and initial conditions will be discussed later for each problem.

Incompressible fluid

If the density and the porosity are not pressure-dependent in Equation (2), we have an incompressible model, where density and porosity do not change over time. Therefore, the incompressible model is time-independent. Assuming no gravity terms and a fluid with constant viscosity, Equation (2) then becomes:

$$-\frac{\rho}{\mu}\nabla \cdot (K\nabla p) = q. \tag{5}$$

Discretization

The spatial derivatives are approximated using a finite difference scheme with cell central differences. For a 3D model, taking 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), where $p_{i,j,l} = p(x_i, y_j, z_l)$ is the pressure at this point.
For the x direction, we have (see, e.g., [21–23]):

$$\begin{aligned}\frac{\partial}{\partial x}\left(k\frac{\partial p}{\partial x}\right) &= \frac{\Delta}{\Delta x}\left(k\frac{\Delta p}{\Delta x}\right) + \mathscr{O}(\Delta x^2)\\ &= \frac{k_{i+\frac{1}{2},j,l}(p_{i+1,j,l} - p_{i,j,l}) - k_{i-\frac{1}{2},j,l}(p_{i,j,l} - p_{i-1,j,l})}{(\Delta x)^2} + \mathscr{O}(\Delta x^2),\end{aligned}$$

where $k_{i-\frac{1}{2},j,l}$ is the harmonic average of the permeability for cells $(i-1, j, l)$ and (i, j, l):

$$k_{i-\frac{1}{2},j,l} = \frac{2}{\frac{1}{k_{i-1,j,l}} + \frac{1}{k_{i,j,l}}}. \tag{6}$$

After discretization, Equation (5), together with boundary conditions, can be written as:

$$\mathbf{Tp} = \mathbf{q}, \tag{7}$$

where $\mathbf{T}$ is known as the transmissibility matrix with elements in adjacent grid cells. The *transmissibility* ($T_{i-\frac{1}{2},j,l}$) between grid cells $(i-1, j, l)$ and (i, j, l) is defined as:

$$T_{i-\frac{1}{2},j,l} = \frac{2\Delta y \Delta z}{\mu \Delta x} k_{i-\frac{1}{2},j,l}, \tag{8}$$

System (7) is a linear system that can be solved with iterative or direct methods. For the solution of this system, it is necessary to define boundary conditions in all boundaries of the domain. These conditions can be prescribed pressures (Dirichlet conditions), flow rates (Neumann conditions) or a combination of these (Robin conditions).

Compressible fluid

If the fluid is compressible with a constant compressibility, the density depends on the pressure Equation (4). Therefore, Equations (1) become:

$$\frac{\partial(\rho(p)\phi)}{\partial t} + \nabla \cdot (\rho(p)v) = q, \qquad v = -\frac{K}{\mu}(\nabla p - \rho(p)g\nabla z), \tag{9}$$

Discretization
Using backward Euler time discretization, Equations (9) are approximated by:

$$\frac{(\phi\rho(p))^{n+1} - (\phi\rho(p))^n}{\Delta t^n} + \nabla \cdot (\rho(p)v)^{n+1} = q^{n+1},$$

$$v^{n+1} = -\frac{K}{\mu^{n+1}}(\nabla(p^{n+1}) - g\rho^{n+1}\nabla z). \tag{10}$$

Assuming no gravity terms, constant fluid viscosity and constant rock porosity, Equations (10) become:

$$\phi\frac{\rho(p^{n+1}) - \rho(p^n)}{\Delta t^n} - \frac{1}{\mu}\nabla \cdot (\rho(p^{n+1})K\nabla p^{n+1}) + q^{n+1} = 0. \tag{11}$$

Due to the dependence of ρ on the pressure, the latter is a nonlinear equation for p that can be linearized with, e.g., the Newton-Raphson (NR) method. Equation (11) can be discretized in space, using a finite differences scheme. After spatial discretization, Equation (11) reads:

$$\frac{\mathbf{V}(\mathbf{p}^{n+1}) - \mathbf{V}(\mathbf{p}^n)}{\Delta t^n} + \mathbf{T}\mathbf{p}^{n+1} = \mathbf{q}^{n+1}. \tag{12}$$

We note that in a more general case, where also the porosity is pressure-dependent, a slightly more complex, mass conservative formulation is usually employed; see refs.[21–23]. As in the incompressible case, we need to define boundary condition to solve Equation (12). Dirichlet, Neumann or Robin boundary conditions can be used. For this problem, we also have a derivative with respect to time. Therefore, it is also necessary to specify the initial conditions that are the pressure values of the reservoir at the beginning of the simulation.

Well model

In reservoirs, wells are typically drilled to extract or inject fluids. Fluids are injected into a well or produced from a well at constant rate or constant bottom-hole pressure (bhp). When the bhp is prescribed, the flow rates into or from the wells are usually computed with the aid of a well model, that takes into account the bhp and the average grid pressure in the block containing the well. This model is a linear relationship between the bhp and the flow rate in a well. For a cell (i, j, l) that contains a well, this relationship is given by:

$$q_{(i,j,l)} = I_{(i,j,l)}(p_{(i,j,l)} - p_{bh(i,j,l)}), \tag{13}$$

where $I_{(i,j,l)}$ is the productivity or injectivity index of the well, $p_{(i,j,l)}$ is the reservoir pressure in the cell where the well is located, and $p_{bh(i,j,l)}$ is a prescribed pressure inside the well.
Incompressible fluid
Using the well model for an incompressible fluid, Equation (7) transforms into:

$$\mathbf{T}\mathbf{p} = \mathbf{I}_w(\mathbf{p} - \mathbf{p}_{bh}), \tag{14}$$

where $\mathbf{I}_w$ is a diagonal matrix containing the productivity or injectivity indices of the wells present in the reservoir. The diagonal elements are zero for cells without wells and have

the value of the well index for each cell containing a well.
Compressible fluid
For a compressible fluid, using the well model, Equation (12) reads:

$$\phi \frac{\rho(\mathbf{p}^{n+1}) - \rho(\mathbf{p}^n)}{\Delta t^n} - \frac{1}{\mu} \nabla \cdot (\rho(\mathbf{p}^{n+1}) \mathbf{K} \nabla \mathbf{p}^{n+1}) + \mathbf{I}_w(\mathbf{p}^{n+1} - \mathbf{p}_{bh}^{n+1}) = \mathbf{0}, \tag{15}$$

or

$$\frac{\mathbf{V}(\mathbf{p}^{n+1}) - \mathbf{V}(\mathbf{p}^n)}{\Delta t^n} + (\mathbf{T} + \mathbf{I}_w)\mathbf{p}^{n+1} - \mathbf{I}_w(\mathbf{p}_{bh}^{n+1}) = \mathbf{0}.$$

Solution procedure for compressible flow
As mentioned before, for the compressible problem, we have a nonlinear system that depends on the pressure at the time step n and the pressure at time step $n+1$:

$$\mathbf{g}(\mathbf{p}^{n+1}; \mathbf{p}^n) = 0. \tag{16}$$

This nonlinear system can be solved with the NR method, the system for the $(k+1)$-th NR iteration is:

$$\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1} = -\mathbf{g}(\mathbf{p}^k; \mathbf{p}^n), \qquad \mathbf{p}^{k+1} = \mathbf{p}^k + \delta\mathbf{p}^{k+1},$$

where $\mathbf{J}(\mathbf{p}^k) = \frac{\partial \mathbf{g}(\mathbf{p}^k; \mathbf{p}^n)}{\partial \mathbf{p}^k}$ is the Jacobian matrix, and $\delta\mathbf{p}^{k+1}$ is the NR update at iteration step $k+1$.
Therefore, the linear system to solve is:

$$\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1} = \mathbf{b}(\mathbf{p}^k). \tag{17}$$

with $\mathbf{b}(\mathbf{p}^k)$ being the function evaluated at iteration step k, $\mathbf{b}(\mathbf{p}^k) = -\mathbf{g}(\mathbf{p}^k; \mathbf{p}^n)$.
The procedure to solve a compressible flow problem consists of three stages. During the first stage, we increase the time with one time step and solve Equation (15) for the new time. Because of the nonlinearity of Equation (15) we use an iterative Newton Raphson procedure that involves linearization at each iteration, i.e. we perform a series of iterations to find the zeros of Equation (16). For every NR iteration the linear system in Equation (17) is solved. In this work, the solution of the linear system is performed with iterative methods (see Section 3). A summary of this procedure is presented in Algorithm 1.

Algorithm 1

```
for t = 0, ...,                                   %Time integration
    Select time step
    for NR_iter = 0, ...,                         %NR iteration
        Find zeros of g(p^{n+1}; p^n) = 0
        for lin_iter = 0, ...,                    %Linear iteration
            Solve J(p^k) δp^{k+1} = b(p^k) for each NR iteration
        end
    end
end
```

3 Iterative solution methods

When simulating single-phase flow through a porous medium, we obtain a linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{18}$$

for both compressible and incompressible models. Since $\mathbf{A}$ is SPD, we choose Conjugate Gradient (CG) as iterative method accelerated with the Incomplete Cholesky preconditioner. In this work, we also study the acceleration with deflation techniques. In this section, we give a brief overview of the methods.

Conjugate Gradient Method
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 $\mathcal{K}_k(\mathbf{A}, \mathbf{r}^0) = span\{\mathbf{r}^0, \mathbf{A}\mathbf{r}^0, \ldots, \mathbf{A}^{k-1}\mathbf{r}^0\}$ and $\mathbf{x}^k \in \mathbf{x}^0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}^0)$ 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:

$$||\mathbf{x} - \mathbf{x}^{k+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}^0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} \right)^{k+1} {}^{1}. \tag{19}$$

The pseudo code for CG is given in Algorithm 2.

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

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

[1]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, $\kappa_2(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})}$.

Preconditioning

To accelerate the convergence of a Krylov method, one can transform the system into another one containing an iteration matrix with a better spectrum , i.e, a smaller condition number. This can be done by multiplying the system (18) by a matrix $\mathbf{M}^{-1}$.

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}. \tag{20}$$

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:

$$||\mathbf{x} - \mathbf{x}^k||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}^0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} + 1} \right)^k. \tag{21}$$

$\mathbf{M}$ is chosen as an SPD matrix such that $\kappa(\mathbf{M}^{-1}\mathbf{A}) \leq \kappa(\mathbf{A})$, and $\mathbf{M}^{-1}\mathbf{b}$ is cheap to compute.

Deflation

Deflation is used to annihilate the effect of extreme eigenvalues on the convergence of an iterative method ([16]). Given an SPD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, for a given matrix $\mathbf{Z} \in \mathbb{R}^{n\times m}$ the deflation matrix $\mathbf{P}$ is defined as follows ([19, 24]):

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

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

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

The matrix $\mathbf{E}$ is known as the *Galerkin* or *coarse* matrix that has to be invertible. If $\mathbf{A}$ is SPD and $\mathbf{Z}$ is full rank then $\mathbf{E}$ is invertible. The full rank matrix $\mathbf{Z}$ is called the $deflation - subspace$ matrix, and it's columns are the $deflation$ vectors or *projection* vectors.

Deflated PCG Method

To obtain the solution of linear system (18), we have to solve the deflated system (see Appendix D):

$$\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}, \tag{22}$$

with the CG method, 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 D):

$$\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^T\hat{\mathbf{x}}. \tag{23}$$

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 [19]:

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

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 $\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{P}\mathbf{b}$ is computed and the error is bounded by:

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

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 $M^{-1}PA$.

3.1 Choices of Deflation Vectors

The deflation method is used to remove the effect of the most unfavorable eigenvalues of $\mathbf{A}$. If the matrix $\mathbf{Z}$ contains eigenvectors corresponding to the unfavorable eigenvalues, the convergence of the iterative method is achieved faster. However, to obtain and to apply the eigenvectors is costly in view of memory and CPU time. Therefore, a good choice of the matrix $\mathbf{Z}$ that efficiently approximate the eigenvectors is essential for the applicability of the method.
A good choice of the deflation vectors is usually problem-dependent. Available information on the system is, in general, used to obtain these vectors. Most of the techniques used to choose deflation vectors are based on approximating eigenvectors, recycling [25], subdomain deflation vectors [1] or multigrid and multilevel based deflation techniques [19, 26]. A summary of these techniques is given below.

Recycling Deflation. A set of search vectors previously used is reused to build the deflation-subspace matrix [25]. The vectors could be, for example, $q-1$ solution vectors of the linear system with different right-hand sides or of different time steps. The matrix $\mathbf{Z}$ containing these solutions is:

$$\mathbf{Z} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(q-1)}].$$

Subdomain Deflation. The domain is divided into several subdomains, using domain decomposition techniques or taking into account the properties of the problem. For each subdomain, there is a deflation vector that contains ones for cells in the subdomain and zeros for cells outside [1].

Multi Grid and Multilevel Deflation. For the multigrid and multilevel methods, the prolongation and restriction matrices are used to pass from one level or grid to another. These matrices can be used as the deflation-subspace matrices $\mathbf{Z}$ [19].

4 Proper Orthogonal Decomposition (POD)

As mentioned before, in this work we want to combine deflation techniques and Proper Orthogonal Decomposition method (POD) to reduce the number of iterations necessary to solve the linear system obtained from reservoir simulation in a cheap and automatic way. In this section, we give a brief overview of the POD method.
The POD method is a Model Order Reduction (MOR) method, where a high-order model is projected onto a space spanned by a small set of orthonormal basis vectors. The high dimensional variable $\mathbf{x} \in \mathbb{R}^n$ is approximated by a linear combination of l orthonormal basis vectors [11]:

$$\mathbf{x} \approx \sum_{i=1}^{l} c_i \psi_i, \tag{24}$$

where $\psi_i \in \mathbb{R}^n$ are the basis vectors and c_i are their corresponding coefficients. In matrix notation, equation (24) is rewritten as :

$$\mathbf{x} \approx \Psi \mathbf{c},$$

where $\Psi = [\psi_1 \ \psi_2 \ .. \ \psi_l]$, $\Psi \in \mathbb{R}^{n \times l}$ is the matrix containing the basis vectors, and $\mathbf{c} \in \mathbb{R}^l$ is the vector containing the coefficients of the basis vectors.
The basis vectors ψ_i are computed from a set of 'snapshots' $\{\mathbf{x_i}\}_{i=1,..,m}$, obtained by simulation or experiments [8]. In POD, the basis vectors $\{\psi_j\}_{j=1}^{l}$, are l eigenvectors corresponding to the largest eigenvalues $\{\sigma_j\}_{j=1}^{l}$ of the data snapshot correlation matrix $\mathbf{R}$.

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

where $\mathbf{X} \in \mathbb{R}^{n \times m}$ is an SPSD matrix containing the previously obtained snapshots. The l eigenvectors should contain almost all the variability of the snapshots. Usually, they are chosen as the eigenvectors of the maximal number (l) of eigenvalues satisfying [8]:

$$\frac{\sum_{j=1}^{l} \sigma_j}{\sum_{j=1}^{m} \sigma_j} \leq \alpha, \qquad 0 < \alpha \leq 1, \tag{26}$$

with α close to 1. The eigenvalues σ_j are ordered from large to small with σ_1 the largest eigenvalue of $\mathbf{R}$. It is not necessary to compute the eigenvalues from $\mathbf{X}\mathbf{X}^T$, instead, it is possible to compute the eigenvalues of the much smaller matrix $\mathbf{X}^T\mathbf{X}$ (see Appendix C).
In this study, we normalize the snapshots, so $||\mathbf{x}_i||_2 = 1$.

5 Deflation vector analysis.

As mentioned in Section 3, it is important to choose 'good' deflation vectors if we want to speed up an iterative method.
We can use solutions of systems slightly different from the original (snapshots) as deflation vectors. For this, we need to choose a way of selecting these snapshots. The idea behind this selection is to obtain a small number of snapshots and, at the same time, obtain the largest amount of information from the system.
In this section, two lemmas are proved. The lemmas are helpful to select the systems used to obtain the snapshots.

Lemma 1. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a non-singular matrix, and $\mathbf{x}$ is the solution of:

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{27}$$

Let $\mathbf{x}_i, \mathbf{b}_i \in \mathbb{R}^n$, $i = 1, ..., m$, be vectors linearly independent $(l.i.)$ and

$$\mathbf{A}\mathbf{x}_i = \mathbf{b}_i. \tag{28}$$

The following equivalence holds

$$\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{x}_i \qquad \Leftrightarrow \qquad \mathbf{b} = \sum_{i=1}^{m} c_i\mathbf{b}_i. \tag{29}$$

Proof $\Rightarrow$

$$\mathbf{x} = \sum_{i=1}^{m} c_i\mathbf{x}_i \Rightarrow \mathbf{b} = \sum_{i=1}^{m} c_i\mathbf{b}_i. \tag{30}$$

Substituting $\mathbf{x}$ from (30) into $\mathbf{A}\mathbf{x} = \mathbf{b}$ leads to:

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^{m} \mathbf{A}c_i\mathbf{x}_i = \mathbf{A}(c_1\mathbf{x}_1 + ... + c_m\mathbf{x}_m).$$

Using the linearity of $\mathbf{A}$ the equation above can be rewritten as:

$$\mathbf{A}c_1\mathbf{x}_1 + ... + \mathbf{A}c_m\mathbf{x}_m = c_1\mathbf{b}_1 + ... + c_m\mathbf{b}_m = \mathbf{B}\mathbf{c}. \tag{31}$$

where $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{c} \in \mathbb{R}^m$, and the columns of $\mathbf{B}$ are the vectors $\mathbf{b}_i$.
From (27) and (31) we get:

$$\mathbf{A}\mathbf{x} = \mathbf{b} = c_1\mathbf{b}_1 + ... + c_m\mathbf{b}_m = \sum_{i=1}^{m} c_i\mathbf{b}_i.$$

Proof $\Leftarrow$

$$\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i \Leftarrow \mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{b}_i. \tag{32}$$

Substituting $\mathbf{b}$ from (32) into $\mathbf{A}\mathbf{x} = \mathbf{b}$ leads to:

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{b}_i. \tag{33}$$

Since $\mathbf{A}$ is non-singular, multiplying (28) and (32) by $\mathbf{A}^{-1}$ we obtain:

$$\mathbf{x}_i = \mathbf{A}^{-1}\mathbf{b}_i,$$

$$\mathbf{x} = \mathbf{A}^{-1} \sum_{i=1}^{m} c_i \mathbf{b}_i = \sum_{i=1}^{m} c_i \mathbf{A}^{-1}\mathbf{b}_i,$$

then

$$\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i. \tag{34}$$

$\boxtimes$

Lemma 2. If the the deflation matrix $\mathbf{Z}$ is constructed with a set of m vectors

$$\mathbf{Z} = \begin{bmatrix} \mathbf{x}_1 & ... & ... & \mathbf{x}_m \end{bmatrix}, \tag{35}$$

such that $\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i$, with $\mathbf{x}_i$ *l.i.*, then the solution of system (27) is obtained with one iteration of DCG.

Proof.

The relation between $\hat{\mathbf{x}}$ and $\mathbf{x}$ is given in Equation (23):

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

For the first term $\mathbf{Q}\mathbf{b}$, taking $\mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{b}_i$ we have:

$$\begin{aligned}
\mathbf{Q}\mathbf{b} &= \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T \left(\sum_{i=1}^{m} c_i \mathbf{b}_i \right) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T \left(\sum_{i=1}^{m} c_i \mathbf{A}\mathbf{x}_i \right) \qquad \text{using Lemma 1} \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T \left(\mathbf{A}\mathbf{x}_1 c_1 + ... + \mathbf{A}\mathbf{x}_m c_m \right) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T (\mathbf{A}\mathbf{Z}\mathbf{c}) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})\mathbf{c} \\
&= \mathbf{Z}\mathbf{c} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + c_3\mathbf{x}_3 + c_4\mathbf{x}_4 + c_5\mathbf{x}_5 \\
&= \sum_{i=1}^{m} c_i \mathbf{x}_i = \mathbf{x}.
\end{aligned}$$

Therefore,

$$\mathbf{x} = \mathbf{Q}\mathbf{b}, \tag{36}$$

is the solution to the original system.
For the second term of Equation (23), $\mathbf{P}^T\hat{\mathbf{x}}$, we compute $\hat{\mathbf{x}}$ from Equation (22):

$$\begin{aligned}
\mathbf{P}\mathbf{A}\hat{\mathbf{x}} &= \mathbf{P}\mathbf{b} \\
\mathbf{A}\mathbf{P}^T\hat{\mathbf{x}} &= (\mathbf{I} - \mathbf{A}\mathbf{Q})\mathbf{b} \qquad \text{using D f) and definition of } \mathbf{P}, \\
\mathbf{A}\mathbf{P}^T\hat{\mathbf{x}} &= \mathbf{b} - \mathbf{A}\mathbf{Q}\mathbf{b} \\
\mathbf{A}\mathbf{P}^T\hat{\mathbf{x}} &= \mathbf{b} - \mathbf{A}\mathbf{x} = 0 \qquad \text{taking } \mathbf{Q}\mathbf{b} = \mathbf{x} \text{ from above}, \\
\mathbf{P}^T\hat{\mathbf{x}} &= 0 \qquad \text{as } \mathbf{A} \text{ is invertible.}
\end{aligned}$$

Then we have obtain the solution

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

in one step of DCG.

⊠

5.1 Accuracy of the snapshots.

If we use an iterative method to obtain an approximate solution $\mathbf{x}^k$ for the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, we cannot compute the relative error e_r (Equation (37)) of the approximation with respect to the true solution because the true solution is unknown,

$$e_r = \frac{||\mathbf{x} - \mathbf{x}^k||_2}{||\mathbf{x}||_2}. \tag{37}$$

Instead, we compute the relative residual r_r (Equation (38)),

$$r_r = \frac{||\mathbf{r}^k||_2}{||\mathbf{b}||_2} \leq \epsilon, \tag{38}$$

and we set a stopping criterium ϵ or tolerance, that is related to the relative error as follows [27] (see Appendix B),

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

Various tolerance values can be used in the experiments for the snapshots as well as for the solution of the original system.
If the maximum relative residual for the snapshots $(\mathbf{x}_i)$ is $\epsilon = 10^{-\eta}$, then, the error in the snapshots is given by

$$\frac{||\mathbf{x}_i - \mathbf{x}_i^k||_2}{||\mathbf{x}_i||_2} \leq \kappa_2(\mathbf{A}) \times 10^{-\eta} = r_r.$$

From Equation (34), if we compute m snapshots with an iterative method such that the solution of $\mathbf{x}$ is a linear combination of these vectors, after one iteration of DCG we obtain

$$\mathbf{x}^1 = \sum_{i=1}^{m} c_i \mathbf{x}_i^{1(i)},$$

where $\mathbf{x}_i^{1(i)}$ is the approximated solution of the snapshot i after one DCG iteration.
The error of this solution is given by:

$$\frac{||\mathbf{x} - \mathbf{x}^1||_2}{||\mathbf{x}||_2} = \frac{||\sum_{i=1}^{m} c_i(\mathbf{x}_i - \mathbf{x}_i^1)||_2}{||\sum_{i=1}^{m} c_i \mathbf{x}_i||_2} \leq \frac{\sum_{i=1}^{m} |c_i| \times \kappa_2(\mathbf{A}) \times 10^{-\eta}}{||\sum_{i=1}^{m} c_i \mathbf{x}_i||_2}.$$

Which means that the approximation has an error of the order $\kappa_2(\mathbf{A}) \times 10^{-\eta}$.
From Lemma 2 we know that if we use the snapshots $\mathbf{x}_i$ as deflation vectors, for the deflation method the solution is given by (Equation (36)):

$$\mathbf{x} = \mathbf{Q}\mathbf{b}.$$

If the approximation $\mathbf{x}^1$ has an error of the order $\kappa_2(\mathbf{A}) \times 10^{-\eta}$, then, the solution achieved with the deflation method will have the same error,

$$\mathbf{Q}\mathbf{b} - \mathbf{x}^1 = \kappa_2(\mathbf{A}) \times 10^{-\eta}.$$

Therefore, it is important to take into account the condition number of the matrix to estimate the accuracy of the deflation vectors.

5.2 Boundary conditions.

From Lemma 2, we know that if we use as deflation vectors a set of m snapshots

$$\mathbf{Z} = [\mathbf{x}_1 \quad ... \quad \mathbf{x}_m],$$

such that $\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i$, where $\mathbf{x}$ is the solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, the solution of the latter system is achieved with one DCG iteration.
In our application, only a small number (m) of elements of the right-hand side vector $\mathbf{b}$ can be changed. This implies that every $\mathbf{b}$ can be written as $\mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{b}_i$. Using Lemma 1, this implies that $\mathbf{x}$ is such that $\mathbf{x} \in span\{\mathbf{x}_1, ..., \mathbf{x}_m\}$, which is called the solution span. Therefore, it is necessary to find the solution span of the system, such that the sum of the elements in the solution span and the sum of right-hand sides give as result the original system. In this section, we explore the subsystems that should be chosen, depending on the boundary conditions of the original system.

Neumann Boundary conditions

When we have Neumann boundary conditions everywhere, the resulting matrix $\mathbf{A}$ is singular, and $\mathbf{A}[1 \quad 1 \quad ... \quad 1 \quad 1]^T = \mathbf{0}$, $Ker(\mathbf{A}) = span([1 \quad 1 \quad ... \quad 1 \quad 1]^T)$. Note that $\mathbf{Ax} = \mathbf{b}$ has only a solution if $\mathbf{b} \in span\{\mathbf{a}_1, ..., \mathbf{a}_n\}$ (with $\mathbf{a}_i$ the $i-th$ column of $\mathbf{A}$), which is equivalent to $\mathbf{b} \perp Ker(\mathbf{A})$ [28]. This implies that if we have m sources with value s_i for the vector $\mathbf{b}_i$, we need that

$$\sum_{j=1}^{m} s_i^j = 0.$$

Then, for each nonzero right-hand side we need to have at least two sources. Therefore, we can have at most $m-1$ linearly independent right-hand sides $\mathbf{b}_i$ containing two sources. This means that the solution space has dimension $m-1$ and it can be spanned by $span\{\mathbf{x}_1, ..., \mathbf{x}_{m-1}\}$. Each of these subsystems will have the same no-flux conditions (Neumann) in all the boundaries. As the original system is a linear combination of the subsystems (Lemma 1), the deflation vectors can be chosen as the solutions corresponding to the subsystems. Therefore, the deflation matrix will be given by:

$$\mathbf{Z} = [\mathbf{x}_1 \qquad ... \qquad \mathbf{x}_{m-1}],$$

and if the accuracy of the snapshots used as deflation vectors is high enough (see Section 5.1), the solution is expected to be achieved in one DCG iteration.

Dirichlet Boundary conditions

In this case, the right-hand side of the system can contain the values of the boundary $\mathbf{b}_b$ and the sources of the system $\mathbf{s}_i$. If we have m sources, as in the previous case, the right-hand side will be given by:

$$\mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{s}_i + \mathbf{b}_b.$$

The subsystems will be $m+1$, where one of them corresponds to the boundary conditions $\mathbf{Ax}_b = \mathbf{b}_b$, and the other m will correspond to the sources $\mathbf{Ax}_i = \mathbf{s}_i$. Therefore, snapshot $m+1$ will be the solution $\mathbf{x}_b$ of the system with no sources and the Dirichlet boundary conditions of the original system. The other m snapshots will correspond to the m sources with homogeneous Dirichlet boundary conditions. Then, the solution space will be given by $span\{\mathbf{x}_1, ..., \mathbf{x}_m, \mathbf{x}_b\}$. If we use the solution of the $m+1$ snapshots as deflation vectors, with the correct accuracy, we will obtain the solution within one DCG iteration.

6 Numerical experiments

6.1 Model problems.

We study the solution of systems of linear equations resulting from the discretization of elliptic and parabolic partial differential equations for the description of single-phase flow through a porous medium. The solution of the system is performed with the Deflated Conjugate Gradient method preconditioned with Incomplete Cholesky (DICCG). We propose the use of snapshots and the snapshots-based POD vectors as deflation vectors for the above-mentioned method.
In the present section, we give a general overview of the experiments that we perform, but the specifications are presented below for each problem separately. In the first part, we solve the elliptic problem (incompressible flow) and the second is devoted to the parabolic problem (compressible flow). For the elliptic problem, a good choice of deflation vectors depends on the boundary conditions of the problem. Hence, we study two cases with different boundary conditions. For the first set of elliptic problems, Dirichlet boundary conditions are used for an academic layered model with various contrasts in permeability between the layers. In the second set of elliptic problems, we used Neumann boundary conditions (no-flux) for the previous academic layered problem and for the SPE 10 benchmark problem. We investigate the behavior of the ICCG and DICCG methods with various contrasts between the permeability layers for both cases.
We study the influence of the size of the problem in the performance of the ICCG and DICCG methods. We vary the grid size of the SPE 10 benchmark, we study diverse grid sizes of the 2nd layer, and the complete benchmark (85 layers).
For the compressible problem, we impose Neumann boundary conditions in all the boundaries. We study a layered permeability problem and the SPE 10 benchmark.

The model
The experiments simulate flow through a porous medium with a constant porosity field of 0.2. We model incompressible and compressible single-phase flow. For the incompressible single-phase model the following properties of the fluid are used:

- $\mu = 1cp$,
- $\rho = 1014kg/m^3$,

In the compressible case, the compressibility of the fluid is:

- $c = 1 \times 10^{-3}$.

In these experiments, a Cartesian grid with different grid sizes is used. Each grid cell has a length of 1 meter. The length of the reservoir (Lx,Ly) is then the number of grid cells in meters. Wells or sources are added to the system. The matrices corresponding to the linear systems $\mathbf{A}$ and right-hand sides $\mathbf{b}$ are obtained with the Matlab Reservoir Simulation Toolbox (MRST) [29].

Snapshots
As mentioned above, for the DICCG method we need a set of deflation vectors. In the first series of experiments (incompressible model), the deflation vectors are solutions of the system with various wells configurations and boundary conditions. These solutions, called snapshots, are obtained with ICCG, the tolerance of the linear solvers is given for each problem. The configuration used to obtain each snapshot depends on the problem that we are solving (see section 5). For the compressible problem, the snapshots are the solutions at the first time steps, first with the same well configuration, and then with different wells configurations. Solutions of the same problem with zero compressibility are also used as snapshots. For each case, the configuration of the snapshots, as well as the configuration of the solved system are presented.

The solver
The solution of the system is approximated with ICCG and DICCG.
For the DICCG method, we need a set of deflation vectors. In the first set of experiments, we use a linearly independent set of solutions as deflation vectors. Then, we use as deflation vectors a linearly dependent set of solutions, and finally, the deflation vectors are a linearly independent basis of the latter dependent set obtained with POD. As tolerance or stopping criterium 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:

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

The stopping criterium is varied for each problem.

6.2 Incompressible Problem

Case 1: Dirichlet and Neumann boundary conditions.

In the configuration of *Case 1*, four wells are positioned in a square at distances equal to one-third of the reservoir length and width. Two wells have a bottom hole pressure (bhp) of 5 bars and two have a bhp of -5 bar. No-flux conditions are imposed at the right and left boundaries and a pressure drop is prescribed in the vertical direction. The pressure on the lower boundary ($y = 0$) is 0 bars, and on the upper boundary ($y = Ly$) is 3 bars. The first four snapshots ($z_1 - z_4$) are obtained setting only one well pressure different from zero, taking no-flux conditions at the right and left boundaries and homogeneous Dirichlet conditions at the other boundaries. A fifth snapshot is obtained setting all the wells pressures to zero and setting the pressure drop in the vertical direction of the original system. A summary is presented in Table 1.

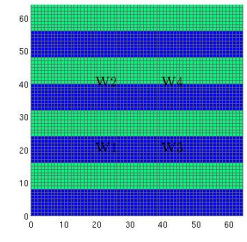

Figure 1: Heterogeneous permeability, 4 wells.

System configuration								
Well pressures (bars)					Boundary conditions (bars)			
	$W1$	$W2$	$W3$	$W4$	$P(y=0)$	$P(y=Ly)$	$\frac{\partial P(x=0)}{\partial n}$	$\frac{\partial P(x=Lx)}{\partial n}$
	-5	-5	+5	+5	0	3	0	0
Snapshots								
	$W1$	$W2$	$W3$	$W4$	$P(y=0)$	$P(y=Ly)$	$\frac{\partial P(x=0)}{\partial n}$	$\frac{\partial P(x=Lx)}{\partial n}$
$\mathbf{z}_1$	-5	0	0	0	0	0	0	0
$\mathbf{z}_2$	0	-5	0	0	0	0	0	0
$\mathbf{z}_3$	0	0	-5	0	0	0	0	0
$\mathbf{z}_4$	0	0	0	-5	0	0	0	0
$\mathbf{z}_5$	0	0	0	0	0	3	0	0

Table 1: Table with the well configuration and boundary conditions of the system and the snapshots used for the Case 1.

As mentioned above, we studied flow through a porous medium with *heterogeneous permeability* layers. A grid of $nx = ny = 64$ elements is studied. We use 8 layers of the same size, 4 layers with one value of permeability σ_1, followed by a layer with a different permeability value σ_2. Figure 1 shows these layers. The permeability of one set of layers is set to $\sigma_1 = 1mD$, the permeability of the other set σ_2 is changed. Therefore, the contrast in permeability between the layers ($\frac{\sigma_2}{\sigma_1} = \sigma_2$), depends on the value of σ_2.
We investigate the dependence on the contrast between permeability layers for the ICCG and DICCG methods. The permeability σ_2 varies from $\sigma_2 = 10^{-1}mD$ to $\sigma_2 = 10^{-3}mD$. The tolerance is set as 10^{-11} for the snapshots as well as for the original problem.

κ_2 (mD)	10^{-1}	10^{-2}	10^{-3}
ICCG	75	103	110
DICCG	1	1	1

Table 2: Table with the number of iterations for different contrasts between the permeability of the layers for the ICCG and DICCG methods.

Table 2 shows the number of iterations required to achieve convergence for ICCG and DICCG for various permeability contrasts between the layers.
The plot of the residual and the solution to the problem are presented in Figures 2 and 3 for a value of permeability $\sigma_2 = 10^{-2}$.
In Table 2 we observe that the number of iterations increases when the contrast between the permeability layers increases for ICCG. For DICCG, we observe that we only need one

iteration despite the change in permeability contrast between the layers.

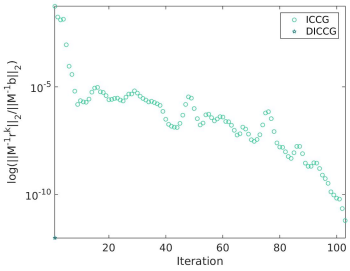

Figure 2: Convergence for the heterogeneous problem, 64 x 64 grid cells, $\sigma_2 = 10^{-2}mD$.

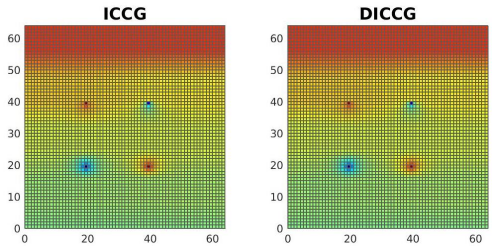

Figure 3: Solution of the heterogeneous problem, 64 x 64 grid cells, $\sigma_2 = 10^{-2}mD$.

Case 2: Neumann boundary conditions everywhere.

In this case, four wells are positioned in the corners and have a bhp of -1 bar. One well is positioned in the center of the domain and has a bhp of +4 bars (see Figure 4). Homogeneous Neumann boundary conditions are imposed on all boundaries. For this case, we use a set of four linearly independent snapshots as deflation vectors. We also use a linearly dependent set of 15 snapshots and the basis of POD (linearly independent set) obtained from the 15 snapshots. We set the same boundary conditions as in the original problem for all the snapshots. The four linearly independent snapshots $(z_1 - z_4)$ are obtained giving a value of zero to one well and non-zero values to the other wells, such that the sum of the well pressures is equal to zero. The set of 15 snapshots are all the possible combinations of wells that satisfy that the flow in equals the flow out of the reservoir. A summary of the configurations is presented below.

Heterogeneous permeability layers

As in the previous case, single-phase flow through a porous medium with heterogeneous permeability layers is studied. A grid of $nx = ny = 64$ elements is investigated. The deflation vectors used in this case are the 4 snapshots ($\mathbf{z}_1$-$\mathbf{z}_4$), a set of 15 linearly dependent vectors and 4 basis vectors obtained for the POD method from the latter set.

The snapshots and the solutions are obtained with a tolerance of 10^{-11}.

Table 4 shows the number of iterations required to reach convergence for the ICCG method and the deflation method with four linearly independent snapshots as deflation vectors DICCG$_4$, 15 linearly dependent snapshots DICCG$_{15}$ and the basis vectors of POD, DICCG$_{POD}$[2].

For the deflation vectors of DICCG$_{POD}$ we plot the eigenvalues of the snapshot correlation matrix $\mathbf{R} = \mathbf{X}^T\mathbf{X}$ (see section 4) in Figure 5. We observe that there are 4 eigenvalues much larger than the rest of the eigenvalues which are responsible for the divergence of the method. In DICCG$_{POD}$ we use the eigenvectors corresponding to the larger eigenvalues as

[2]The * means that the solution is not reached.

System configuration					
Well pressures (bars)					
	$W1$	$W2$	$W3$	$W4$	$W5$
	-1	-1	-1	-1	-1
Snapshots (4 linearly independent)					
	$W1$	$W2$	$W3$	$W4$	$W5$
$\mathbf{z}_1$	0	-1	-1	-1	3
$\mathbf{z}_2$	-1	0	-1	-1	3
$\mathbf{z}_3$	-1	-1	0	-1	3
$\mathbf{z}_4$	-1	-1	-1	0	3

Snapshots (linearly dependent)					
	W1	W2	W3	W4	W5
$\mathbf{z}_5$	-1	-1	-1	-1	4
$\mathbf{z}_6$	-1	0	0	-1	2
$\mathbf{z}_7$	-1	-1	0	0	2
$\mathbf{z}_8$	-1	0	-1	0	2
$\mathbf{z}_9$	0	-1	-1	0	2
$\mathbf{z}_{10}$	0	-1	0	-1	2
$\mathbf{z}_{11}$	0	0	-1	-1	2
$\mathbf{z}_{12}$	-1	0	0	0	1
$\mathbf{z}_{13}$	0	-1	0	0	1
$\mathbf{z}_{14}$	0	0	-1	0	1
$\mathbf{z}_{15}$	0	0	0	-1	1

Table 3: Table with the well configuration of the system and the snapshots used for the Case 2, we use homogeneous Neumann boundary conditions.

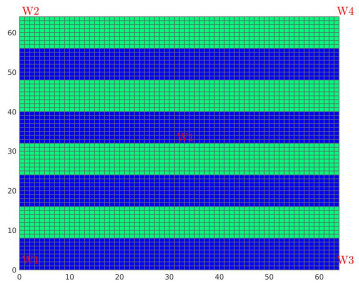

Figure 4: Heterogeneous permeability, 5 wells.

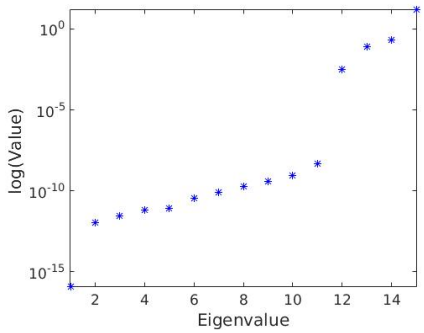

Figure 5: Eigenvalues of the snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, if 15 snapshots are used.

deflation vectors.
The plot of the residual and the solution of the problem are presented in Figure 6 and 7 for the ICCG and DICCG methods for the case of $\sigma_2 = 10^{-2}$.

σ_2 (mD)	10^{-1}	10^{-2}	10^{-3}
ICCG	90	115	131
DICCG_4	1	1	1
DICCG_{15}	200*	200*	200*
DICCG_{POD}	1	1	1

Table 4: Table with the number of iterations for different contrast in the permeability of the layers for the ICCG, DICCG_4, DICCG_{15}, and DICCG_{POD} methods, tolerance of solvers and snapshots 10^{-11}.

In Table 4, for the ICCG method, we observe that the number of iterations increases if the contrast in the permeability increases. For the DICCG method with 4 linearly independent deflation vectors and 4 basis vectors of POD, convergence is reached within one iteration. However, for the case of 15 linearly dependent vectors, the solution is not reached within the 200 iterations allowed for this problem.

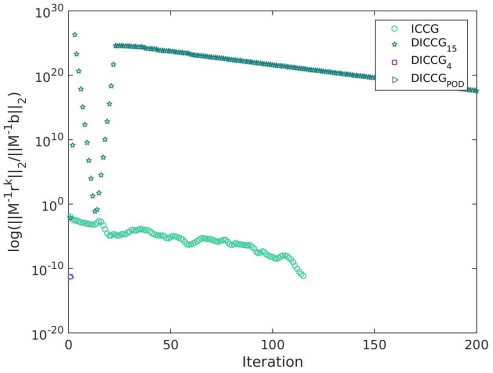

Figure 6: Convergence for the heterogeneous problem, 64 x 64 grid cells, $\sigma_2 = 10^{-2}$.

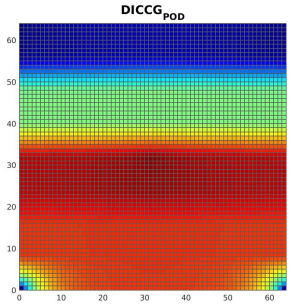

Figure 7: Solution of the heterogeneous problem, 64 x 64 grid cells, $\sigma_2 = 10^{-2}$.

SPE 10 model

This model has large variations in the permeability coefficients, the contrast between coefficients is of the order of 10^7. It has 5 sources or wells, four producers in the corners (negative) and one injector in the center (positive). The model contains 60 x 220 x 85 cells. We study the dependence of the ICCG and the DICCG method on the size of the problem. One layer is studied with various grid sizes 16 x 56, 30 x 110, 46 x 166 and 60 x 220, and the complete model containing 85 layers. Permeability is upscaled averaging the permeability in each grid using the harmonic-arithmetic average algorithm from MRST. The permeability of the coarser grid (16 x 56 cells) is shown in Figure 8 and the complete model in Figure 9. The permeability contrast for the diverse grid size problems is shown in Table 5. From this table, we observe that the contrast in the permeability for different grid sizes varies slightly, but that the order of magnitude remains the same for all the cases.
Snapshots are obtained solving the system with different well configurations (*Configuration 2*). As before, we simulate single-phase incompressible flow.

The system and snapshots are solved with an accuracy of 10^{-11}. In the first experiment with the deflation method, the four linearly independent snapshots are used as deflation vectors (DICCG). Then, 15 linearly dependent vectors and finally 4 vectors of the POD basis are used as deflation vectors (DICCG$_{POD}$).

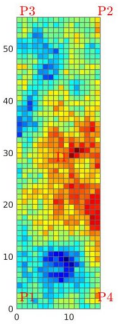

Figure 8: SPE 10 benchmark, 2nd layer 16 x 56 grid cells, permeability field.

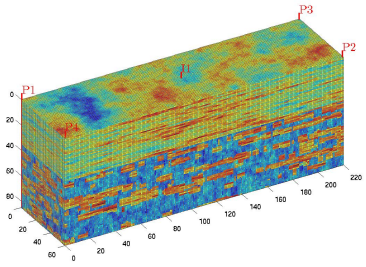

Figure 9: SPE 10 benchmark, permeability field.

Grid size	16x56x1	30x110x1	46x166x1	60x220x1	60x220x85
Contrast ($\times 10^7$)	1.04	2.52	2.6	2.8	3

Table 5: Table with the number of iterations for different grid sizes for the ICCG, DICCG$_4$, DICCG$_{15}$, and DICCG$_{POD}$ methods, tolerance of solvers and snapshots 10^{-11}.

The number of iterations required to achieve convergence with the ICCG and DICCG methods for various grid sizes is presented in Table 6.
The convergence and the solution obtained with the ICCG and DICCG methods are presented in Figure 10 and Figure 11 for the complete problem. In Table 6 we observe that for the ICCG method the required iterations to reach convergence increases as the size of the grid increases. Meanwhile, for the deflated methods only a few iterations are required and it does not depend on the size of the grid. The large contrast in the permeability field may require higher accuracy in the snapshots to find the solution with a deflated method within one iteration (see [30]) within the imposed tolerance. However, we observe in Figure 10 that the first iteration has a relative residual smaller than 10^{-10} for the DICCG$_4$ and DICCG$_{POD}$ methods. We also observe that for the deflated method with 15 linearly dependent snapshots as deflation vectors (DICCG$_{15}$), the relative residual is close to 10^{-7} for the first time steps, and then it increases, which shows that this choice leads to an unstable method (note that the matrix E is a nearly singular matrix).

Method	16x56x1	30x110x1	46x166x1	60x220x1	60x220x85
ICCG	45	101	178	219	1011
DICCG_{15}	500*	500*	500*	500*	2000*
DICCG_{4}	1	2	3	2	2
DICCG_{POD}	1	2	3	2	2

Table 6: Table with the number of iterations for ICCG and DICCG methods, various grid sizes.

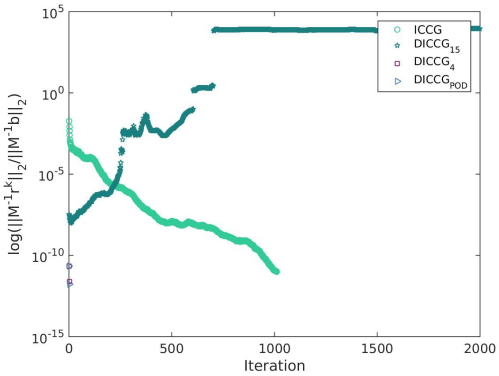

Figure 10: Convergence for the SPE 10 benchmark, 60 x 220 x 85 grid cells, accuracy of the snapshots and solvers 10^{-11}.

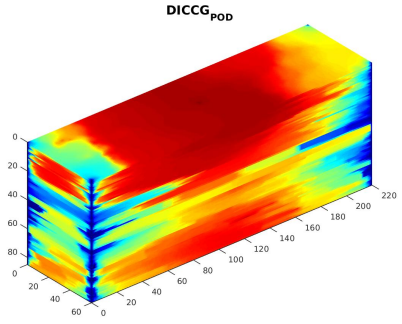

Figure 11: Solution of the SPE 10 benchmark, 60 x 220 x 85 grid cells, accuracy of the snapshots and solvers 10^{-11}.

6.3 Compressible Problem

6.3.1 Model parameters

In this section we model single-phase flow through a porous medium for a case when the density depends on the pressure according to Equation (4). We solve Equation (12) for a fluid with the a compressibility of $c = 1 \times 10^{-3}$ and the same viscosity and density as in the compressible case. Equation (12) is non-linear due to the dependence of the density on the pressure. Therefore, we need to linearize this equation via the Newton-Raphson (NR) method and to solve the resulting linear system. After linearization, we obtain the linear system (17) and we solve it with an iterative method, a summary of the procedure is presented in Algorithm 1. The simulation, with exception of the linear solvers, is performed with MRST. Automatic Differentiation (AD) is used for the NR loop [29]. The resulting linear system is solved with ICCG and DICCG methods. We compute the solution of the system for the first 10 time steps with the ICCG method. The rest of the time steps is solved with DICCG, using as deflation vectors the solution of the previous ten time steps and POD basis vectors computed from these solutions. The number of POD deflation vectors is specified for each problem.

We study an academic layered problem that consists of layers with two different permeability values (see Figure 12). The first layer has a permeability of $\sigma_1 = 30mD$, and the permeability of the second layer is varied $\sigma_2 = [3\text{mD}, 0.3\text{mD}, 0.03\text{mD}]$. Therefore, the contrast between the layers is 10^{-1}, 10^{-2} and 10^{-3}. The domain is a square with five wells, Four of which are positioned in the corners of the domain and one well is placed in the center. The length of the domain is 70 m and three different grid sizes are studied: 35, 70 and 105 grid cells in each dimension. We use homogeneous Neumann boundary conditions on all boundaries.
The initial pressure of the reservoir is set as 200 bars. The pressure in the corner wells is 100 bars and in the central well is 600 bars.

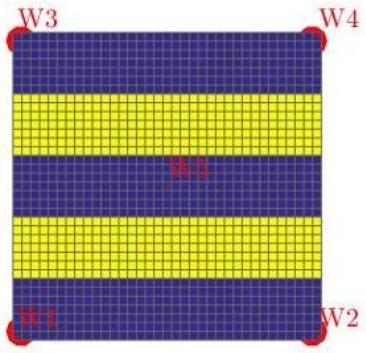

Figure 12: Heterogeneous permeability, 5 wells, compressible problem.

The simulation was performed during 152 days with 52 time steps and a time step of 3 days. The tolerance of the NR method and the linear solvers is 10^{-5}.

Different contrast in permeability.
As mentioned previously we change the contrast between the permeability layers, in this section we present the results obtained for three different contrast for a 2D Cartesian grid of 35×35 grid cells covering an area of 70×70 m^2. As a first set of experiments, we compute 10 snapshots, solutions to the first 10 time steps, with ICCG and we use these snapshots as deflation vectors to solve the rest of the time steps with deflation DICCG$_{10}$. After the first these snapshots are computed, we update the snapshots with the most

recently computed solution, such that the ten snapshots correspond to the ten solutions of the ten previous time steps. We compute SVD of the matrix constructed with these snapshots as columns and we study the eigenvalues obtained to select as deflation vectors the eigenvectors corresponding to the largest eigenvalues.

Contrast between permeability layers of 10^{-1}.
In Figure 13, the solution obtained with the ICCG method is presented, the solution is the same for all methods. The upper left figure represents the pressure field at the final time step. The upper right figure represents the pressure across the diagonal joining the (0,0) and (35,35) grid cells for all the time steps. We observe the initial pressure (200 bars) across this diagonal and the evolution of the pressure field through time. In the lower figure, we observe the surface volume rate for the five wells during the simulation.

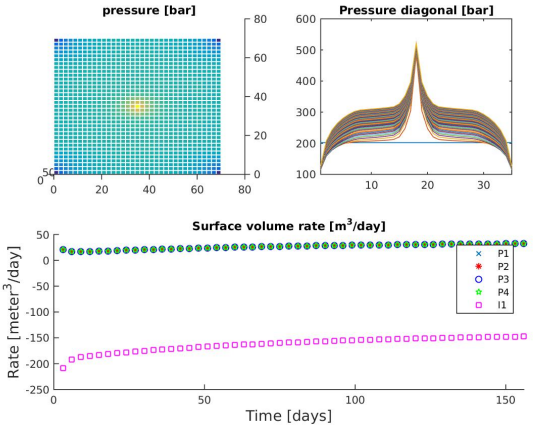

Figure 13: Solution of the compressible problem solved with the ICCG method for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

As a second set of experiments, we use basis vectors of POD as deflation vectors of the DICCG method, these basis vectors are the eigenvectors corresponding of the largest eigenvalues of the snapshot correlation matrix $\mathbf{X}$ (see Section 4). The snapshot correlation matrix is constructed with the previously computed solutions, i.e., the solutions of the previous time steps. As mentioned before, for each time step, the previous 10 solutions are used as snapshots to compute the POD basis. The eigenvalues of the snapshot correlation matrix $\mathbf{R} = \frac{1}{m}\mathbf{X}\mathbf{X}^T$ constructed with the previous ten time steps are presented in Figure 15 for the 20^{th} time step. In this figure, we observe that six eigenvalues are larger than the rest. Therefore, we use the eigenvectors corresponding to these six eigenvalues as deflation vectors (DICCG$_6$).

For this problem, only the first time step requires more than two NR iterations. Therefore, we solely study the behavior of the linear solvers during the first two NR iterations. The number of iterations necessary to reach convergence with the linear solvers is presented for the first two NR iterations in Figure 16 for the ICCG method, Figure 18 for the deflated method DICCG$_{10}$ using 10 snapshots as deflation vectors and Figure 20 using 6

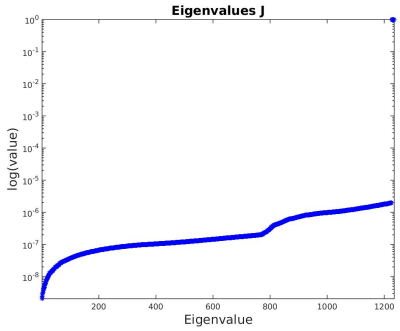

Figure 14: Eigenvalues of the original matrix $\mathbf{J}$, time step 1 for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

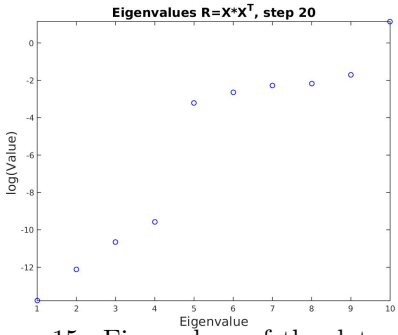

Figure 15: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20 for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

POD basis vectors as deflation vectors. The eigenvalues of the matrices are presented in Figure 14 for the original system matrix $\mathbf{J}$ for the first time step, Figure 17 for the preconditioned system, Figure 19 for DICCG$_{10}$ and Figure 21 the deflated system DICCG$_6$. The preconditioned system is studied for the first time step, and the deflated systems are studied for the 11^{th} time step. For the preconditioned and the deflated system, the same scale is used for the comparison.

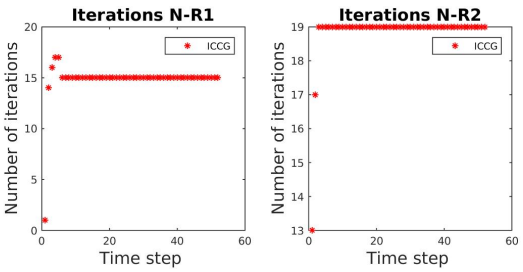

Figure 16: Number of iterations of the ICCG method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

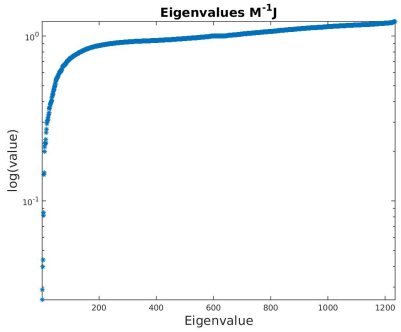

Figure 17: Eigenvalues of the preconditioned matrix, time step for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

From Figure 17, Figure 19 and Figure 21 we observe that the smallest eigenvalues are not longer visible in the system. This is because we use the same scale for the plots in all the figures and with the deflation methods we remove the smallest eigenvalues, they are sent to zero, in this case to a very small value. As they are very small, they are not longer important for the convergence of the system. After removing these eigenvalues, the condition number is reduced and therefore we obtain an acceleration in the convergence of the method. From the spectrum of these systems, we observe that after the deflation pro-

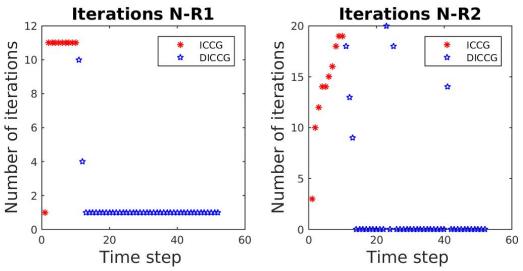

Figure 18: Number of iterations of the DICCG_{10} method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

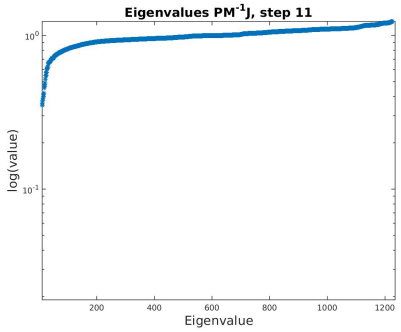

Figure 19: Eigenvalues of the deflated system DICCG_{10} for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

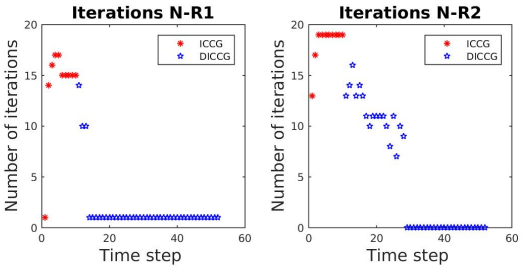

Figure 20: Number of iterations of the DICCG_{6} method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

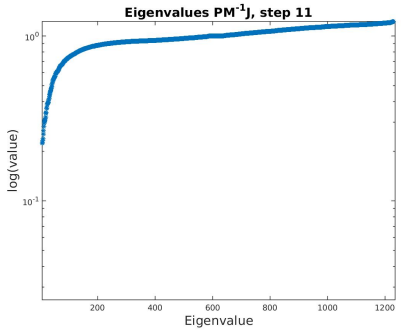

Figure 21: Eigenvalues of the deflated system DICCG_{6} for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

cedure we reduce in one order of magnitude the condition number (see Table 7), the order the magnitude of the eigenvalues is the same for both deflation cases (10 and 6 deflation vectors).

From Figure 16, Figure 18 and Figure 20, we observe that the reduction of the condition

Matrix	λ_{max}	λ_{min}	$\kappa_2 = \frac{\lambda_{max}}{\lambda_{min}}$
$\mathbf{M}^{-1}\mathbf{J}$	1	$\approx 10^{-2}$	$\approx 10^{2}$
$\mathbf{PM}^{-1}\mathbf{J}$	1	$\approx 10^{-1}$	$\approx 10^{1}$

Table 7: Condition number of the preconditioned and deflated systems for a layered problem with a contrast between permeability layers of 10^{-1} in a domain of 70×70 m^2.

number results in a reduction of the number of iterations for the first and second NR iterations of the deflated methods (DICCG$_{10}$, DICCG$_6$) compared with the ICCG method. For the ICCG method, we need on average 15 and 19 linear iterations in the first two NR iterations to reach the desired tolerance. In contrast, for the deflated methods, we need on average 1 or 2 linear iterations for the first NR iteration. For the second NR iteration, after the computation of the snapshots, instead of computing the solution for all the remaining 42 time steps computed for the ICCG method, we only need to compute 18 times steps with an average of 3 and 11 linear iterations. Which implies that the convergence is already achieved after the first NR iteration for the rest of the time steps. A summary of the average number of iterations is presented in Table 8 and Table 9.

Contrast between permeability layers of 10^{-2}.

We repeat the experiments of previous sections. In this case, the contrast between permeability layers is 10^{-2}. The solution obtained with the ICCG method is presented in Figure 22, the solution is the same for the DICCG method. The eigenvalues of the snapshot correlation matrix $\mathbf{R}$ for the 20^{th} time step are presented in Figure 24. From this figure, we observe that there are 7 eigenvalues of the correlation matrix larger than the rest. Therefore, the largest amount of information might be contained in these vectors. For this problem, we studied the deflation method using 6 (DICCG$_6$) and 7 (DICCG$_7$) POD basis vectors as deflation vectors as well as the DICCG$_{10}$ method with 10 snapshots as deflation vectors.

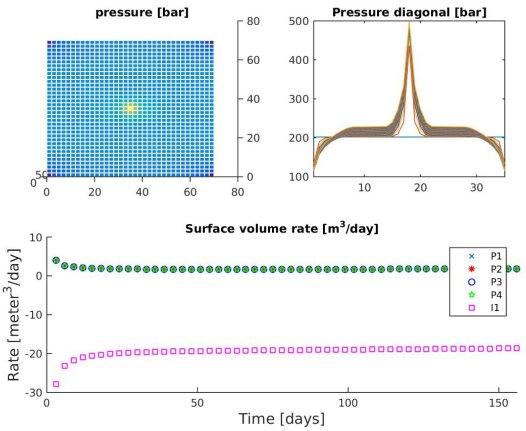

Figure 22: Solution of the compressible problem solved with the ICCG method for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

As in the previous case, we study the behavior of the linear solvers only for the first two NR iterations. The number of iterations necessary to reach convergence with the linear solvers is presented for the first two NR iterations in Figure 25 for the ICCG method, Figure 27 for the deflated method DICCG$_{10}$, Figure 29 for DICCG$_6$ and Figure 31 for DICCG$_7$. The eigenvalues of the matrices are presented in Figure 23 for the original system matrix $\mathbf{J}$ for the first time step, Figure 26 for the preconditioned system, Figure 28

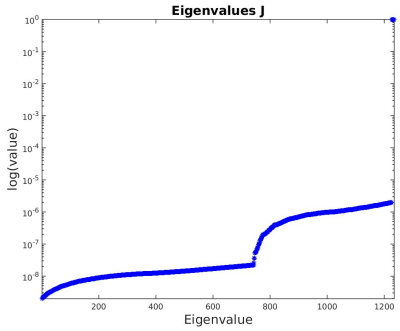

Figure 23: Eigenvalues of the original matrix $\mathbf{J}$, time step 1 for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

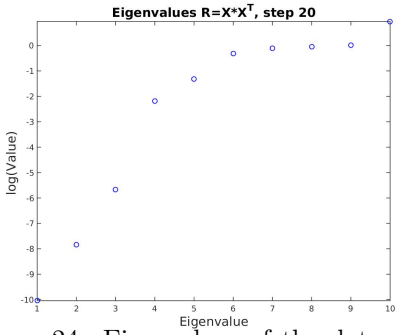

Figure 24: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20 for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

the deflated system DICCG$_{10}$, Figure 30 the deflated system DICCG$_6$ and Figure 32 for DICCG$_7$. From the previously mentioned figures, we observe that some eigenvalues from the preconditioned system are not longer in the plot for deflated systems, this is because they are very small and they are not longer visible in the system. When we use 10 snapshots as deflation vectors, we remove more eigenvalues than when use 6 or 7, but the order of magnitude of the smallest eigenvalue is the the same for all cases, which means that the behavior should be similar. Comparing the cases when we have 6 and 7 deflation vectors, we observe that both spectra are almost the same except for one eigenvalue that is smaller than the case when we use 6 deflation vectors (Figure 30). Hence, we expect a slightly better behavior when using 7 deflation vectors.

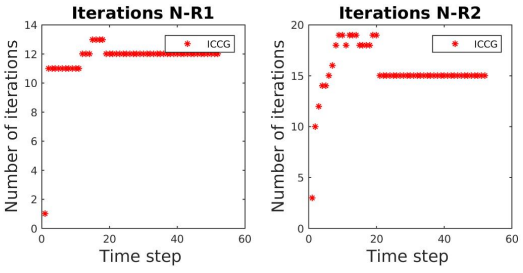

Figure 25: Number of iterations of the ICCG method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

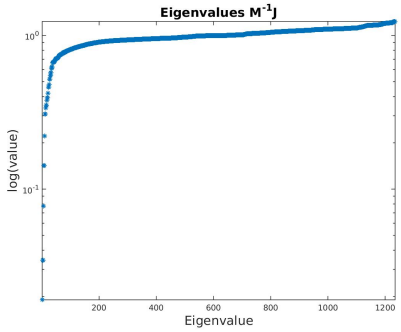

Figure 26: Eigenvalues of the preconditioned matrix, time step 1 for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

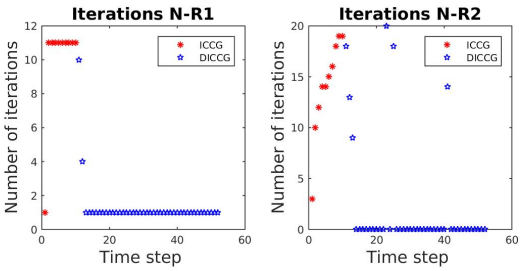

Figure 27: Number of iterations of the DICCG_{10} method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

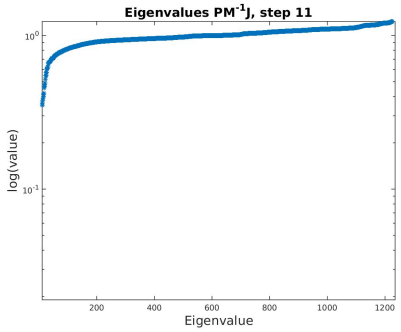

Figure 28: Eigenvalues of the deflated system DICCG_{10} for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

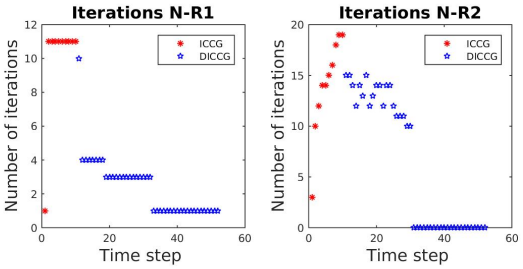

Figure 29: Number of iterations of the DICCG_{6} method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

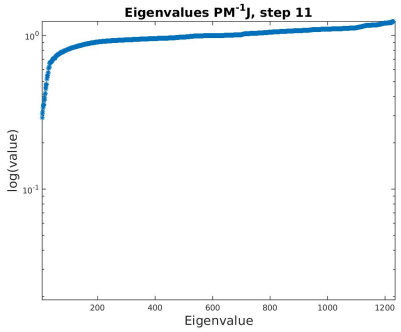

Figure 30: Eigenvalues of the deflated system DICCG_{6} for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

From Figure 25, Figure 27, Figure 29 and Figure 31, we observe that the number of iterations of the first and second NR iterations is lower for the deflated methods compared with the ICCG method. For the ICCG method, we need on average 12 and 16 linear iterations in the first two NR iterations to reach the desired accuracy. In contrast, for the deflated method DICCG_{10} we need in average 1 iteration for the first NR iteration, 2 for DICCG_{6} and 1 for the DICCG_{7} method. For the second NR iteration, besides the 10 snapshots, we only need to compute the solution for 6, 20 and 20 extra time steps with an average of 15, 13 and 14 linear iterations for the DICCG_{10}, DICCG_{6} and DICCG_{7} methods (see Tables 8 and 9).

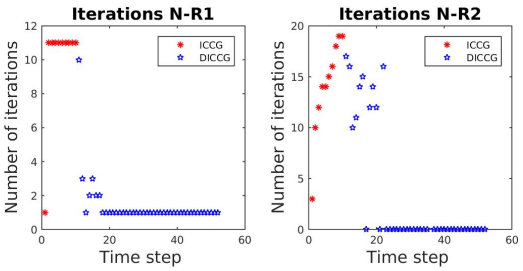

Figure 31: Number of iterations of the DICCG$_7$ method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

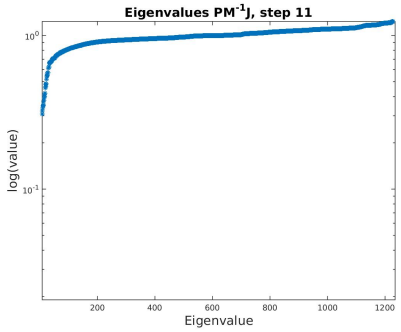

Figure 32: Eigenvalues of the deflated system DICCG$_7$ for a layered problem with a contrast between permeability layers of 10^{-2} in a domain of 70×70 m^2.

Contrast between permeability layers of 10^{-3}.
The solution obtained with the ICCG method is presented in Figure 33, the solution is the same for the DICCG method. The eigenvalues of the snapshot correlation matrix $\mathbf{R}$ for the 20^{th} time step are presented in Figure 35. From this figure, we observe that there are 6 eigenvalues larger than the rest, but the 7^{th} eigenvalue is also large compared with the rest of the eigenvalues. Therefore, we study the deflation methods using the 10 previous time steps as deflation vectors DICCG$_{10}$ and 6 and 7 POD basis vectors DICCG$_6$, DICCG$_7$.

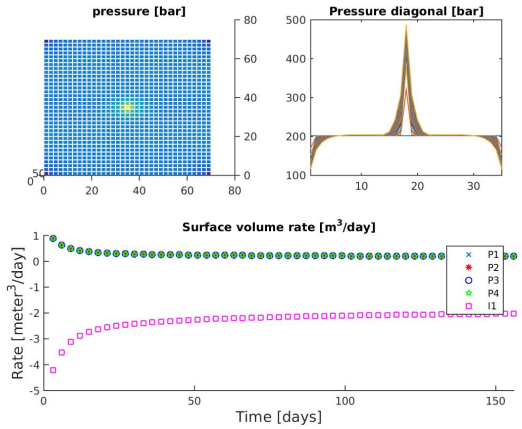

Figure 33: Solution of the compressible problem solved with the ICCG method for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

The number of iterations necessary to reach convergence with the linear solvers is presented for the first two NR iterations in Figure 36 for the ICCG method, Figure 38 for the deflated method DICCG$_{10}$ using 10 snapshots as deflation vectors, Figure 40 for the

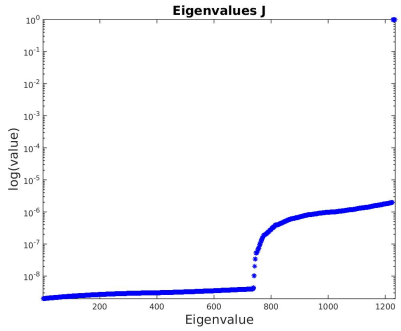

Figure 34: Eigenvalues of the original matrix $\mathbf{J}$, time step 1 for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

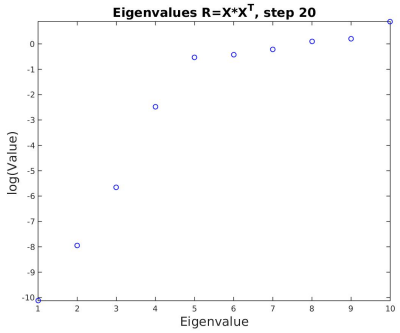

Figure 35: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20 for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

deflated method DICCG$_6$ using 6 POD basis vectors as deflation vectors and Figure 42 using 7 POD basis vectors as deflation vectors DICCG$_7$.
The eigenvalues of the matrices are presented in Figure 34 for the original system matrix $\mathbf{J}$ for the first time step, Figure 37 for the preconditioned system, Figure 39 the deflated system DICCG$_{10}$, Figure 41 for DICCG$_6$ and Figure 43 for DICCG$_7$. As in the previous cases, we observe that the smallest eigenvalues of the preconditioned system (Figure 37) are removed with the deflation methods. We also observe that the spectra of the three deflatad systems is similar, except for the case when we use 6 defaltion vectors (DICCG$_6$), where we have an eigenvalue smaller than the rest of the spectrum. Hence, we expect a slightly worst behavior with this selection of deflation vectors.

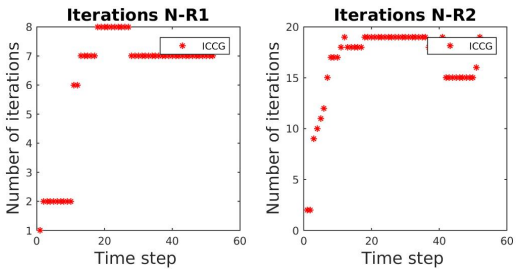

Figure 36: Number of iterations of the ICCG method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

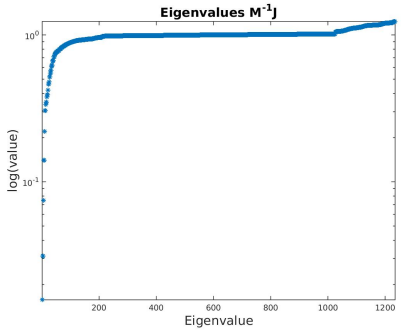

Figure 37: Eigenvalues of the preconditioned matrix, time step 11 for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

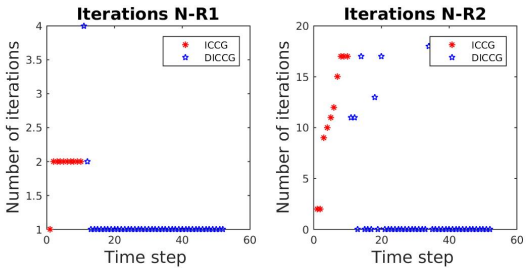

Figure 38: Number of iterations of the DICCG_{10} method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

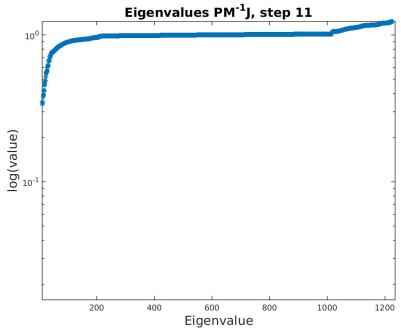

Figure 39: Eigenvalues of the deflated system DICCG_{10} for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

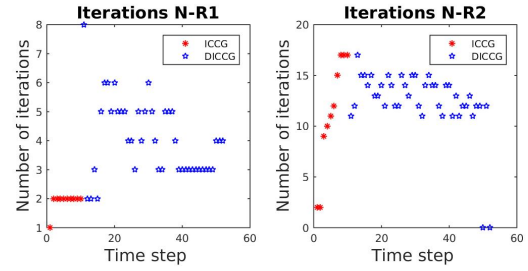

Figure 40: Number of iterations of the DICCG_{6} method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

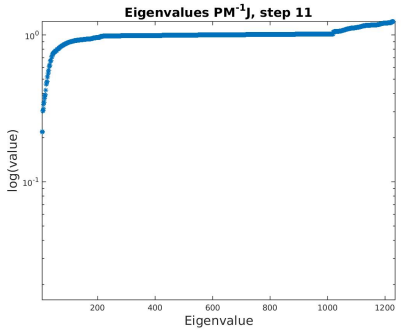

Figure 41: Eigenvalues of the deflated system DICCG_{6} for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

From Figure 36, Figure 38, Figure 40 and Figure 42, we observe that the number of iterations for the first and second NR iterations is lower for the deflated methods compared with the ICCG method. For the ICCG method, we need on average 7 and 17 linear iterations for the first two NR iterations to reach the desired tolerance. For the deflated method DICCG_{10}, we need on average 1 iteration, 4 for DICCG_{6} and 1 for the DICCG_{7} for the first NR iteration. After the computation of the snapshots, we need to compute the solution for 6 (DICCG_{10}), 40 (DICCG_{6}) and 10 (DICCG_{7}) time steps during the second NR iteration for the deflated methods, this means that the solution is already achieved for the rest of the time steps during the first NR iteration. On average, 15, 13 and 15 linear solver iterations are required for the DICCG_{10}, DICCG_{6} and DICCG_{7} for these 10 time steps (see Tables 8 and 9).

In Table 8 and Table 9 the average number of linear iterations (Average L-iter) is presented

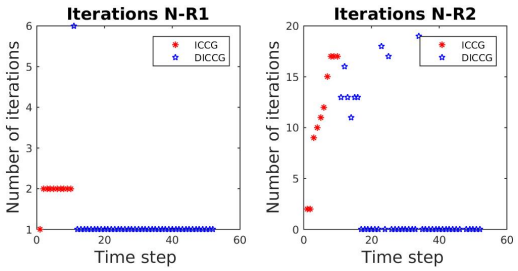

Figure 42: Number of iterations of the DICCG_7 method for the first two NR iterations for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

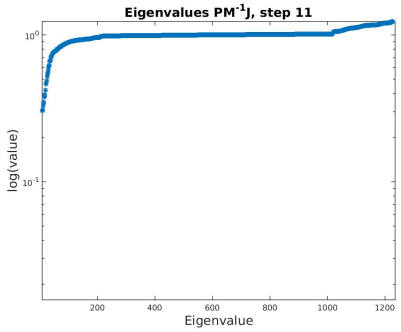

Figure 43: Eigenvalues of the deflated system DICCG_7 for a layered problem with a contrast between permeability layers of 10^{-3} in a domain of 70×70 m^2.

for the ICCG, DICCG_{10}, DICCG_6 and DICCG_7 methods. The number of time steps computed with each method (Time steps) is also presented in the tables. However, for the deflated methods we first compute 10 snapshots with ICCG, these iterations are separated from the iterations performed with the deflated methods in the table. The total number of iterations is also computed (Tot L-iter = Average L-iter * Time steps).

For the first NR iteration, we observe a significant reduction in the total number of linear iterations. For the case when we have a contrast between permeability layers of 10^{-1} we observe that with ICCG we need 780 linear iterations to compute the solution for the 52 time steps. By contrast, when we use the deflated method, we need 140 linear iterations to compute the snapshots during the first ten time steps and 42 and 84 for the 42 remaining time steps computed with DICCG_{10} and DICCG_6. Then, we need in total 182 and 224 linear iterations to compute the solution for the 52 time steps, which is 23% and 29% of the linear iterations required with (see Table 11).

When we have a contrast in permeability of 10^{-2}, the required average of linear iterations to solve the 52 time steps is 624. With the deflated methods, taking into account the computation of the snapshots, we require 142 for the DICCG_{10} method, 184 for the DICCG_6 method and 142 for the DICCG_7 method. That is the 23%, 30% and 23% of the ICCG iterations. Then, for the DICCG_7 and DICCG_{10} methods we have a larger acceleration during the solution of the linear system. However, we note that with 6 deflation vectors we also have a good improvement.

For a contrast between permeability layers of 10^{-3} we require 364 linear iterations for the 52 time steps with the ICCG method. We require 62, 188 and 62 iterations for the DICCG_{10},DICCG_6 and DICCG_7 methods. That is 17%, 51% and 17% of the ICCG iterations. For this case, the larger reduction is achieved when we use 10 or 7 deflation vectors, where we observe a similar behavior (see Table 10).

For some time steps, it is not necessary to compute a second NR iteration, because the solution is already achieved during the first NR iteration when we use deflation methods.

1^{st} NR Iteration							
Method	$\frac{\sigma_2}{\sigma_1}$	Time steps		Average L-iter		Tot L-iter	
ICCG	10^{-1}	52		15		780	
	10^{-2}	52		12		624	
	10^{-3}	52		7		364	
		ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG
$DICCG_{10}$	10^{-1}	10	42	14	1	140	42
$DICCG_6$		10	42	14	2	140	84
$DICCG_{10}$		10	42	10	1	100	42
$DICCG_6$	10^{-2}	10	42	10	2	100	84
$DICCG_7$		10	42	10	1	100	42
$DICCG_{10}$		10	42	2	1	20	42
$DICCG_6$	10^{-3}	10	42	2	4	20	168
$DICCG_7$		10	42	2	1	20	42

Table 8: Average number of linear iterations for the first NR iteration for various contrast between permeability layers.

Therefore, not only the number of linear iterations but also the NR iterations are reduced with deflation. For the second NR iteration, we also observe a significant reduction in the total number of linear iterations. For the case when we have a contrast between permeability layers of 10^{-1} we observe that with ICCG we need 988 linear iterations to compute the solution for the 52 time steps. By contrast, when we use the deflated method, we need 180 linear iterations to compute the snapshots during the first ten time steps and 78 and 198 for the 42 remaining time steps with the $DICCG_{10}$ and $DICCG_6$ methods. Therefore we need in total 258 and 378 linear iterations to compute the solution for the 52 time steps, which is 26% and 38% of the linear iterations required with ICCG (see Table 11). When we have a contrast in permeability of 10^{-2}, the required linear iterations to solve the 52 time steps are 832. With the deflated methods, taking into account the computation of the snapshots, we require 230 iterations for $DICCG_{10}$, 400 for $DICCG_6$ and 294 for the $DICCG_7$ method. That means the 28%, 48% and 33% of the ICCG iterations. We can observe that there is a significant difference when we use 6 deflation vectors compared with 10 or 7. For a contrast between permeability layers of 10^{-3}, we require 884 linear iterations for the 52 time steps. For the $DICCG_{10}$, $DICCG_6$ and $DICCG_7$ methods, we require 200, 630 and 260 iterations . That is 23%, 71% and 29% of the ICCG iterations. As in the previous cases, a larger reduction in the number of iterations is achieved when we use 10 snapshots as deflation vectors or 7 POD basis vectors as deflation vectors.

2^{nd} NR Iteration							
Method	$\frac{\sigma_2}{\sigma_1}$	Time steps		Average L-iter		Tot L-iter	
ICCG	10^{-1}	52		19		988	
	10^{-2}	52		16		832	
	10^{-3}	52		17		884	
		ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG
$DICCG_{10}$	10^{-1}	10	26	18	3	180	78
$DICCG_6$		10	18	18	11	180	198
$DICCG_{10}$		10	6	14	15	140	90
$DICCG_6$	10^{-2}	10	20	14	13	140	260
$DICCG_7$		10	11	14	14	140	154
$DICCG_{10}$		10	6	11	15	110	90
$DICCG_6$	10^{-3}	10	40	11	13	110	520
$DICCG_7$		10	10	11	15	110	150

Table 9: Average number of linear iterations for the second NR iteration for various contrast between permeability layers.

1^{st} NR Iteration						
$\frac{\sigma_2}{\sigma_1}$	Method	Total ICCG (only)	ICCG Snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
10^{-1}	$DICCG_{10}$	780	140	42	182	23
	$DICCG_6$		140	84	224	29
	$DICCG_{10}$		100	42	142	23
10^{-2}	$DICCG_6$	624	100	84	184	30
	$DICCG_7$		100	42	142	23
	$DICCG_{10}$		20	42	62	17
10^{-3}	$DICCG_6$	364	20	168	188	51
	$DICCG_7$		20	42	62	17

Table 10: Comparison between the ICCC and DICCG methods of the average number of linear iterations for the first NR iteration for various contrast between permeability layers.

2^{nd} NR Iteration						
$\frac{\sigma_2}{\sigma_1}$	Method	Total ICCG (only)	ICCG Snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
10^{-1}	$DICCG_{10}$	988	180	78	258	26
	$DICCG_6$		180	198	378	38
10^{-2}	$DICCG_{10}$	832	140	90	230	28
	$DICCG_6$		140	260	400	48
	$DICCG_7$		140	154	294	33
10^{-3}	$DICCG_{10}$	884	110	90	200	23
	$DICCG_6$		110	520	630	71
	$DICCG_7$		110	150	260	29

Table 11: Comparison between the ICCC and DICCG methods of the average number of linear iterations for the second NR iteration for various contrast between permeability layers.

Different grid sizes.

In the previous section, we presented the results for the different contrast between permeabilities, for a grid of 35×35 cells in a reservoir of 70×70 m^2. In this section, we change the size of the grid to 70 and 105 grid cells. We study a layered problem with a contrast between permeability layers of 10^{-1}.

Grid size 70×70**.**

In this case, we study the number of iterations needed to reach convergence for a problem with 70×70 grid cells in a reservoir of 70×70 m^2. As in the previous cases, only the first time step requires more than two NR iterations. Therefore, we solely study the behavior of the linear solvers during the first two NR iterations. In Figure 44 the eigenvalues of the snapshot correlation matrix are presented. We observe that there are 6 eigenvalues larger than the rest, which implies that most of the information is contained in these eigenvalues. Therefore, we study the deflation method with 10 snapshots as deflation vectors and 6 eigenvectors corresponding to the largest eigenvalues of Figure 44 are used as deflation vectors.

The number of iterations necessary to reach convergence with the linear solvers is presented for the first two NR iterations in Figure 45 for the ICCG method, Figure 46 for the deflated method $DICCG_{10}$ using 10 snapshots as deflation vectors, and Figure 47 with 6 POD basis vectors as deflation vectors.

From Figures 46 and 47 we observe that the number of iterations needed for the $DICCG_{10}$ and $DICCG_6$ methods is on average 84 and 126 for the first NR iteration, and 144 and 483 for the second NR iteration after the snapshots are computed, 390 linear iterations. Comparing with the ICCG method (Figure 46) that requires 1848 iterations for this problem, the number of iterations is considerably reduced. A summary of the number of linear iterations is presented in Tables 12 and 13.

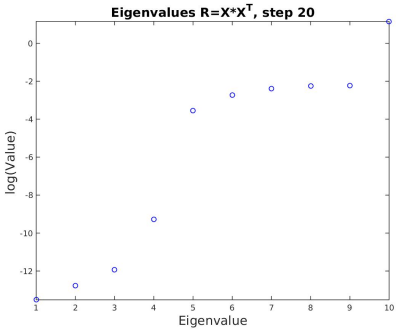

Figure 44: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20, Grid size 70×70.

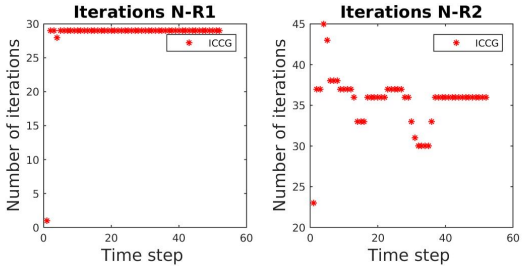

Figure 45: Number of iterations of the ICCG method for the first two NR iterations, grid size 70×70, contrast between permeability layers 10^{-1}.

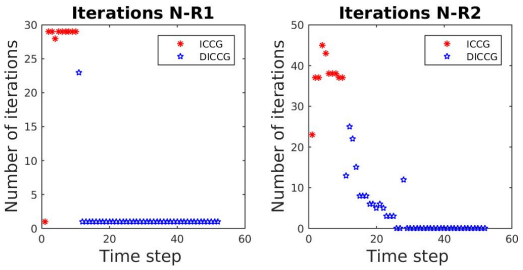

Figure 46: Number of iterations of the DICCG_{10} method for the first two NR iterations, grid size 70×70, contrast between permeability layers 10^{-1}.

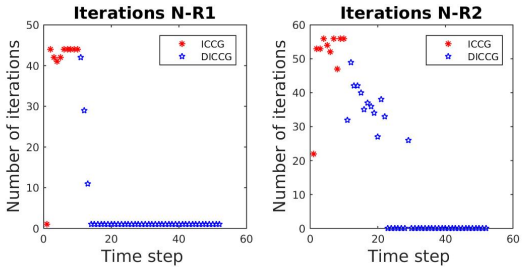

Figure 47: Number of iterations of the DICCG_6 method for the first two NR iterations, grid size 70×70, contrast between permeability layers 10^{-1}.

Grid size 105×105**.**

In this case, we study the number of iterations needed to reach convergence for a problem with 105×105 grid cells in a reservoir of 70×70 m^2. In Figure 48 the eigenvalues of the snapshot correlation matrix are presented. We observe that there are 6 eigenvalues larger than the rest, which implies that most of the information is contained in these eigenvalues. Therefore, we study the deflation method with 10 snapshots as deflation vectors and 6 eigenvectors corresponding to the largest eigenvalues of Figure 48 are used as deflation vectors.

The number of iterations necessary to reach convergence with the linear solvers is presented for the first two NR iterations in Figure 49 for the ICCG method, Figure 50 for the deflated method DICCG_{10} using 10 snapshots as deflation vectors, and Figure 51 with 6 POD basis vectors as deflation vectors.

The computation of the first 10 snapshots with ICCG requires an average of 390 linear

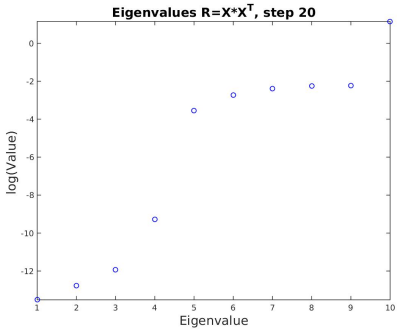

Figure 48: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20, grid size 105×105, contrast between permeability layers 10^{-1}.

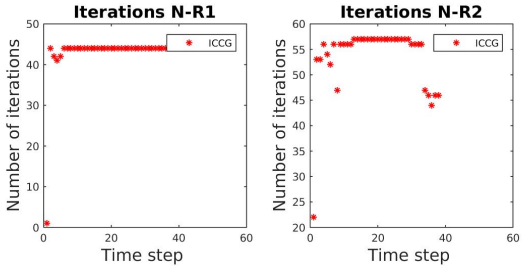

Figure 49: Number of iterations of the ICCG method for the first two NR iterations, grid size 105×105, contrast between permeability layers 10^{-1}.

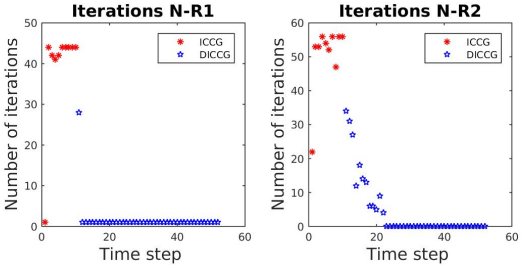

Figure 50: Number of iterations of the DICCG_{10} method for the first two NR iterations, grid size 105×105, contrast between permeability layers 10^{-1}.

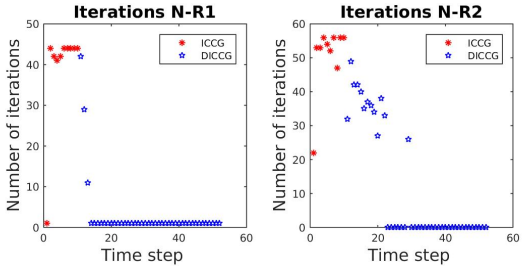

Figure 51: Number of iterations of the DICCG_6 method for the first two NR iterations, grid size 105×105, contrast between permeability layers 10^{-1}.

iterations for the first NR iteration and 510 for the second. From Figures 50 and 51 we observe that, after the computation of the snapshots, the number of iterations needed for the DICCG_{10} and DICCG_6 methods is on average 84 and 126 for the first NR iteration, and 180 and 468, on average, for the second NR iterations. Comparing with the ICCG method (Figure 50) that requires 2823 iterations for this problem, the number of iterations is considerably reduced. A summary of the number of linear iterations is presented in Tables 12 and 13.

In Tables 14 and 15 we compare the number of iterations necessary to reach convergence with the ICCG method and the deflation methods DICCG_{10}, DICCG_6 with various grid sizes. We observe that we have a considerable reduction on the number of linear iterations when we use deflation methods. For the first NR iteration, we need around 24% of the

1^{st} NR Iteration							
Size	Method	Time steps		Average L-iter		Tot L-iter	
35×35	ICCG	52		15		780	
70×70		52		28		1479	
105×105		52		43		2238	
		ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG
35×35	$DICCG_{10}$	10	42	14	1	140	42
	$DICCG_6$	10	42	14	2	140	84
70×70	$DICCG_{10}$	10	42	26	2	260	84
	$DICCG_6$	10	42	26	3	260	126
105×105	$DICCG_{10}$	10	42	39	2	390	84
	$DICCG_6$	10	42	39	3	390	126

Table 12: Average number of linear iterations for the first NR iteration for various grid sizes.

2^{nd} NR Iteration							
Size	Method	Time steps		Average L-iter		Tot L-iter	
35×35	ICCG	52		19		988	
70×70		52		36		1848	
105×105		52		54		2823	
		ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG
35×35	$DICCG_{10}$	10	26	18	3	180	78
	$DICCG_6$	10	18	18	11	180	198
70×70	$DICCG_{10}$	10	16	37	9	370	144
	$DICCG_6$	10	21	37	23	370	483
105×105	$DICCG_{10}$	10	12	51	15	510	180
	$DICCG_6$	10	13	51	36	510	468

Table 13: Average number of linear iterations for the first NR iteration for various grid sizes.

number of ICCG iterations for all cases. We also note that the difference between the deflation methods is small for this case. For the second NR iteration, we also have a reduction in the linear iterations. We require around 25% for the $DICCG_{10}$ method, and around 40% for the $DICCG_6$ method. This means that, for this case, the performance of the $DICCG_6$

method is slightly better, but with the $DICCG_6$ we also have a good improvement with respect to the ICCG method.

1^{st} NR Iteration						
Size	Method	Total ICCG (only)	ICCG Snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
35×35	$DICCG_{10}$	780	140	42	182	23
	$DICCG_6$		140	84	224	29
70×70	$DICCG_{10}$	1479	260	84	344	23
	$DICCG_6$		260	126	386	26
105×105	$DICCG_{10}$	2238	390	84	474	21
	$DICCG_6$		390	126	516	23

Table 14: Comparison between the ICCG and DICCG methods of the average number of linear iterations for the first NR iteration for various grid sizes.

2^{nd} NR Iteration						
Size	Method	Total ICCG (only)	ICCG Snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
35×35	$DICCG_{10}$	988	180	78	258	26
	$DICCG_6$		180	198	378	38
70×70	$DICCG_{10}$	1848	370	144	514	28
	$DICCG_6$		370	483	853	46
105×105	$DICCG_{10}$	2823	510	180	690	24
	$DICCG_6$		510	468	978	34

Table 15: Comparison between the ICCG and DICCG methods of the average number of linear iterations for the second NR iteration for various grid sizes.

SPE 10

We study the SPE 10 benchmark in 2D and 3D. For the 2D case, we use the second layer that consists of 60×220 grid cells. For the 3D case, we use the complete model that consist of $60 \times 220 \times 85$ grid cells. To solve the linear system obtained after the NR linearization, we use 10 snapshots (the previous 10 time step solutions), and POD basis vectors as deflation vectors, the number of POD basis vectors depends on the problem. As in the previous experiments, only the first time step requires more than two NR iterations. Therefore, we solely study the behavior of the linear solvers during the first two NR iterations.

Second layer SPE 10

The second layer of the SPE 10 benchmark is studied, this layer contains 60×220 grid cells and has a contrast in permeability of 2.8×10^7.

In Figure 52 the eigenvalues of the snapshot correlation matrix are presented. We observe that there are 7 eigenvalues larger than the rest, which implies that most of the information is contained in these eigenvalues. Therefore, we study the deflation method with 10 snapshots as deflation vectors and 7 POD basis vectors, the largest eigenvectors corresponding to the largest eigenvalues of Figure 52 used as deflation vectors.

The number of iterations necessary to reach convergence with the linear solvers for each time step is presented for the first two NR iterations in Figure 53 for the ICCG method, Figure 54 for the deflated method DICCG_{10} using 10 snapshots as deflation vectors, and Figure 55 with 7 POD basis vectors as deflation vectors (DICCG_7).

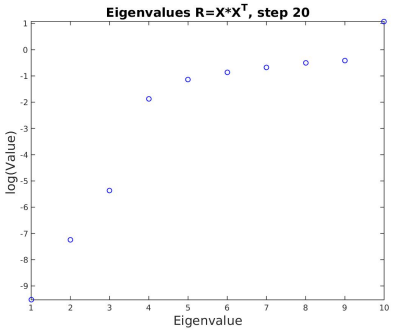

Figure 52: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20, second layer of the SPE 10 benchmark.

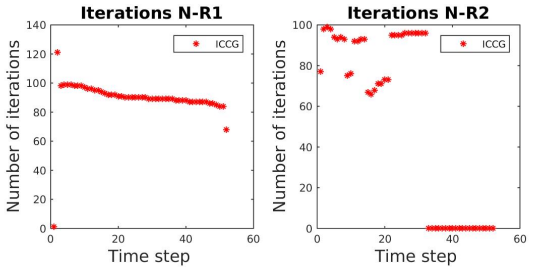

Figure 53: Number of iterations of the ICCG method for the first two NR iterations, second layer of the SPE 10 benchmark.

From Figures 53, 54 and 55 we observe that there is a considerable reduction in the number of iterations necessary to solve the linear system when we used the DICCG_{10} and DICCG_7 methods compared with the ICCG method. For the first NR iteration, from Tables 19 and 17 we observe that the total average number of linear iterations necessary to achieve convergence with the ICCG method is 4644. For the deflated methods it is necessary to perform 994 and 1120 iterations for DICCG_{10} and DICCG_6 which is the 21% and 24% of the linear iterations required with the ICCG method. For the second NR iteration, from Tables 19 and 17 we observe that the total average number of linear iterations necessary

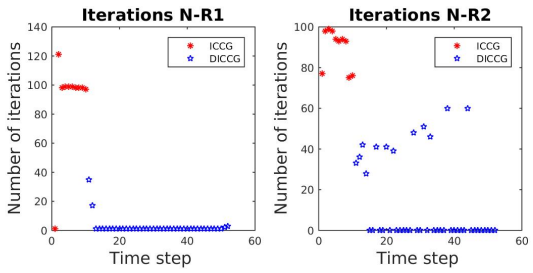

Figure 54: Number of iterations of the DICCG_{10} method for the first two NR iterations, second layer of the SPE 10 benchmark.

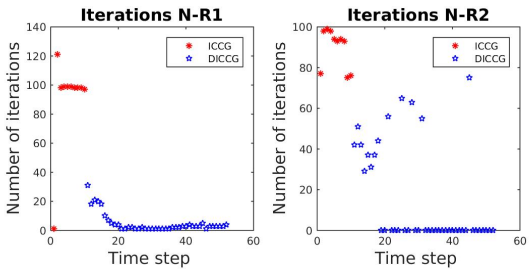

Figure 55: Number of iterations of the DICCG_7 method for the first two NR iterations, second layer of the SPE 10 benchmark.

to achieve convergence with the ICCG method is 2808. For the deflated methods it is necessary to perform 1428 and 624 iterations for DICCG_{10} and DICCG_6, which is the 50% and 54% of the linear iterations required with the ICCG method. We also observe that the number of total iterations required for the deflated methods is less than the required total iterations with the ICCG method, which means that an important part of the work necessary to solve this problem is performed to compute the initial snapshots.

	1^{st} NR Iteration							
Method	Time steps		Average L-iter		Tot L-iter			
ICCG	52		89		4644			%
	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	Total	
DICCG_{10}	10	42	91	2	910	84	994	21
DICCG_7	10	42	91	5	910	210	1120	24

Table 16: Average number of linear iterations for the first NR iteration, second layer of the SPE 10 benchmark.

	2^{nd} NR Iteration							
Method	Time steps		Average L-iter		Tot L-iter			
	52		88		2808			%
	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	Total	
$DICCG_{10}$	10	12	90	44	900	528	1428	50
$DICCG_7$	10	13	90	48	900	624	1524	54

Table 17: Average number of linear iterations for the second NR iteration, second layer of the SPE 10 benchmark.

SPE 10 full model

In this section, we study the SPE 10 complete benchmark that consist of $60 \times 220 \times 85$ grid cells and has a contrast in permeability of 3×10^7. In Figure 56 the eigenvalues of the snapshot correlation matrix are presented. We observe that there are 4 eigenvalues larger than the rest, which implies that most of the information is contained in these eigenvalues. Therefore, we study the deflation method with 10 snapshots as deflation vectors and 4 POD basis vectors, the largest eigenvectors corresponding to the largest eigenvalues of Figure 56 used as deflation vectors.

The number of iterations necessary to reach convergence with the linear solvers is presented for the first two NR iterations in Figure 53 for the ICCG method, Figure 58 for the deflated method $DICCG_{10}$ using 10 snapshots as deflation vectors, and Figure 59 with 4 POD basis vectors as deflation vectors ($DICCG_4$).

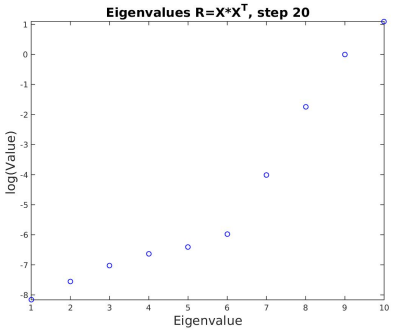

Figure 56: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20, full SPE 10 benchmark.

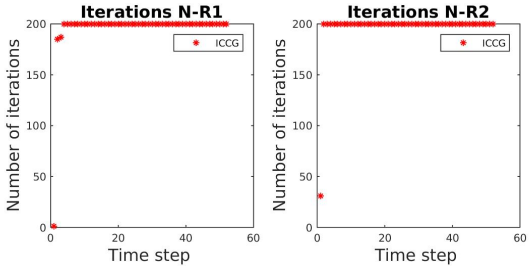

Figure 57: Number of iterations of the ICCG method for the first two NR iterations, full SPE 10 benchmark.

For the first NR iteration, we observe that the average number of iterations required for the ICCG method (Figure 58) is considerably reduced. The average number of iterations required with the ICCG method is 10173 for the first NR iteration and 10231 for the second (see Tables 19 and 17). With the deflated methods $DICCG_{10}$ and $DICCG_4$, for the first NR iteration, we only need to perform 28% and 32% of the linear iterations required with

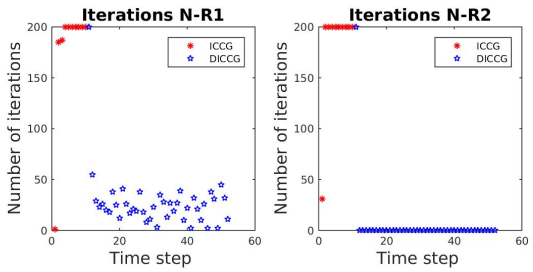

Figure 58: Number of iterations of the DICCG_{10} method for the first two NR iterations, full SPE 10 benchmark.

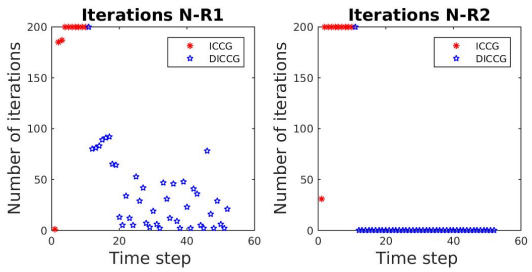

Figure 59: Number of iterations of the DICCG_4 method for the first two NR iterations, full SPE 10 benchmark.

the ICCG method. For the second NR iteration, the deflated methods require only 20% of the ICCG linear iterations. For the first NR iteration, we need 1770 linear iterations to compute the snapshots and 1134 to compute the solution of the rest of the snapshots. For the second NR iteration, the number of linear iterations is 1830 for the snapshots and 200 for the deflated methods. This shows that the largest amount of work is carried out in the computation of the snapshots, which is more evident for the second NR iteration.

	1^{st} NR Iteration							
Method	Time steps		Average L-iter		Tot L-iter			
ICCG	52		196		10173			%
	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	Total	
DICCG_{10}	10	42	177	27	1770	1134	2904	28
DICCG_4	10	42	177	37	1770	1554	3324	32

Table 18: Average number of linear iterations for the first NR iteration, full SPE 10 benchmark.

	2^{nd} NR Iteration							
Method	Time steps		Average L-iter		Tot L-iter			
ICCG	52		197		10231			%
	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	ICCG Snapshots	DICCG	Total	
$DICCG_{10}$	10	1	183	200	1830	200	2030	20
$DICCG_4$	10	1	183	200	1830	200	2030	20

Table 19: Average number of linear iterations for the second NR iteration, full SPE 10 benchmark.

Conclusions

Acceleration of the Conjugate Gradient (CG) method for systems with high contrast in permeability and for large systems is studied in this work. Preconditioning techniques are combined with deflation to speed-up convergence of the CG method. In this work, the deflated Conjugated Gradient preconditioned with Incomplete Cholesky method (DICCG) is studied with *snapshots*, solutions of the system with diverse characteristics, and POD basis vectors as deflation vectors. The performance of this method is compared with the Conjugate Gradient method preconditioned with Incomplete Cholesky (ICCG).
Flow through a porous medium is studied for an incompressible and a compressible fluid. We study an academic layer problem with different permeability values in the layers and the 2D and 3D SPE 10 benchmark, a model with large variations in the permeability field, both problems with diverse grid sizes.
For the incompressible case, the number of linear iterations required with ICCG is reduced to only a few iterations with DICCG. The number of linear iterations required with deflation methods is independent of the contrast in permeability or the size of the grid for both, the academic and the SPE 10 problems. To solve the incompressible problem, we propose the use of solutions of the problem with different well configurations as deflation vectors. Results show that, if we have a linearly dependent set of deflation vectors, we have an unstable method that leads to a bad approximation of the solution. The combination of POD with deflation techniques is shown to be a way to obtain the main information about the system to speed-up the iterative method and to avoid instabilities. For this problem, we prove two Lemmas where we show that if we have a linearly independent set of deflation vectors that span the solution, convergence is achieved in one iteration with deflation. To select this linear set of vectors, it is necessary to take into account the boundary conditions of the problem. To find the solution within one iteration it is also necessary to take into account the condition number and, therefore, a correct accuracy of the snapshots.
For the compressible case, we propose the use of solutions of previous time steps, *snapshots*, and POD basis vectors as deflation vectors. We use 10 snapshots as deflation vectors. Computing the POD basis is also done with 10 snapshots. The required number of POD basis vectors to achieve a good acceleration of the method depends on the problem. Only a limited number of deflation vectors is necessary to obtain a good speed-up (less than eight for the problems here studied). The performance of the DICCG method with snapshots and POD basis vectors as deflation vectors is similar. We observe an important reduction of the number of linear iterations with the DICCG method with respect to the ICCG method. The number of POD basis vectors used as deflation vectors increases when we have a higher contrast between permeability layers varying from 6 when we have a contrast of 10^{-1} to 7 when we have a contrast of 10^{-2} and 10^{-3}. For a grid of 35×35 grid cells, with the DICCG method, we only need to compute on average 23% and 33% of the number of ICCG iterations for the first and second NR iterations. We also observe that a considerable part of this work is carried out to compute the snapshots with the ICCG method. The performance of the deflated method with 10 snapshots as deflation vectors is similar with different grid sizes. Meanwhile, a slightly larger variation is observed for the

deflated method with POD basis vectors as deflation vectors, which might indicate that more POD basis vectors are necessary. However, we observe an important reduction in the number of linear iterations when compared with the ICCG method in all cases. For the deflated method with snapshots as deflation vectors, we require on average 24% of the linear iterations required with the ICCG method for all the grid sizes. The deflated method with POD basis vectors as deflation vectors requires on average 32% of the ICCG linear iterations. For the second layer of the SPE 10 problem, with the deflated methods, we require on average 22% of the ICCG iterations to achieve convergence for the first NR iteration, and 52% for the second. For the complete model, we require 30% for the first NR iteration and 20% for the second.
For the full SPE 10 problem with four deflation vectors, each DICCG iteration needs around 1.4 times the number of flops of the ICCG iteration and the computation of the POD basis requires around 10^4 flops, which is less than the number of cells of the problem (see Appendix E).
The deflation techniques here presented are not restricted to these methods and could be combined with different preconditioners, e.g. SSOR or AMG, and diverse iterative 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] 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.

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

[3] A. Antoulas. *Approximation of large-scale dynamical systems*. SIAM, 2005.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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*, January 2011.

[12] S. Krogstad et al. A sparse basis POD for model reduction of multiphase compressible flow. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2011.

[13] Y. Efendiev, J. Galvis and E. Gildin. Local–global multiscale model reduction for flows in high-contrast heterogeneous media. *Journal of Computational Physics*, 231(24):8100–8113, 2012.

[14] R. Jiang. Pressure Preconditioning Using Proper Orthogonal Decomposition. Master's thesis, Stanford University, 2013.

[15] M. 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*, 2016.

[16] 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.

[17] 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.

[18] H. Klie, M.F. Wheeler, K. Stueben, T. Clees et al. Deflation amg solvers for highly ill-conditioned reservoir simulation problems. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2007.

[19] 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.

[20] K. Carlberg, V. Forstall, and R. Tuminaro. Krylov-subspace recycling via the POD-augmented conjugate-gradient algorithm. *arXiv preprint arXiv:1512.05820*, 2015.

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

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

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

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

[25] 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.

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

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

[28] G. Strang. *Linear Algebra and Its Applications*. Wellesley-Cambridge Press, 2009.

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

[30] 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.

A List of notation

Symbol	Quantity	Unit
ϕ	Rock porosity	
$\mathbf{K}$	Rock permeability	$Darcy\ (D)$
c_r	Rock compressibility	Pa^{-1}
$\mathbf{v}$	Darcy's velocity	m/d
ρ	Fluid density	kg/m^3
μ	Fluid viscosity	$Pa \cdot s$
p	Pressure	Pa
g	Gravity	m/s^2
c_f	Fluid compressibility	Pa^{-1}
q	Sources	

Table 20: Notation

B 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 [27, 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:

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

C Singular Value Decomposition for POD

If we perform SVD in $\mathbf{X}$, we have
$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T, \quad \mathbf{U} \in \mathbb{R}^{n\times n}, \quad \mathbf{\Sigma} \in \mathbb{R}^{n\times m}, \quad \mathbf{V} \in \mathbb{R}^{m\times m}.$
Then we have

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

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

If we compute Λ^T, we can compute U as follows:

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

D Deflation method

In this appendix, we explain how to obtain the solution of the linear system (18) with deflation. Some properties of the matrices used for deflation that will help us to find the solution of system (18) are [19]:

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 (18), we start with the splitting:

$$\mathbf{x} = \mathbf{x} - \mathbf{P}^T\mathbf{x} + \mathbf{P}^T\mathbf{x} = (\mathbf{I} - \mathbf{P}^T)\mathbf{x} + \mathbf{P}^T\mathbf{x}. \tag{39}$$

Multiplying expression (39) by $\mathbf{A}$, using the properties of the deflation matrices, we have:

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

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 (40) 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:

$$\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}. \tag{40}$$

As the solution of Equation (40) can contain components of $\mathcal{N}(\mathbf{P}\mathbf{A})$, $\hat{\mathbf{x}}$ can be decomposed as:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{y}, \tag{41}$$

with $\mathbf{y} \in \mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{P}\mathbf{A})$, and $\mathbf{x}$ the solution of Equation (18).
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:

$$\mathbf{P}^T\mathbf{y} = \mathbf{0}. \tag{42}$$

Multiplying Equation (41) by $\mathbf{P}^T$ we obtain:

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

substituting Equation (42) we arrive to:

$$\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x}. \tag{43}$$

Substitution of Equation (43) and property c) in Equation (39) leads to:

$$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T\hat{\mathbf{x}}, \tag{44}$$

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

E 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 21) and DICCG method (see Table 23).
With the numbers previously computed, we compute the number of operations necessary to perform the ICCG (see Table 22) and DICCG methods (see Table 24). In Table 25, 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).

Let $\mathbf{A}$, $\mathbf{B} \in \mathbb{R}^{n\times n}$, and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$.

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) $=$ $= 2m^2n - nm$
$\mathbf{A} \in \mathbb{R}^{m\times n}\mathbf{B} \in \mathbb{R}^{n\times p}$		
$\mathbf{AB}$	$mp(2n - 1)$	
$\mathbf{A} = \mathbf{LL}^T$	$1/3n^3$	
$\mathbf{Lx} = \mathbf{y}$	n^2	nm
$\mathbf{L}^T\mathbf{x} = \mathbf{y}$	n^2	nm

Table 21: Number of operations between matrices and vectors.

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 22: Number of operations needed to perform the ICCG method.

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

Table 23: Number of operations needed to compute some matrices and vectors necessary to perform the DICCG 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 24: Number of operations needed to perform the DICCG method.

			m	p=10	p=4
	ICCG	(4m+11)n	23n	23n	23n
m=3	DICCG	(4m+11+4p)n	(23+4p)n	63n	39n
	DICCG/ICCG			63/23=2.7	39/23=1.7
	ICCG	(4m+11)n	31n	31n	31n
m=5	DICCG	(4m+11+4p)n	(31+4p)n	71n	47n
	DICCG/ICCG			71/31=2.3	47/31=1.5
	ICCG	(4m+11)n	39n	39n	39n
m=7	DICCG	(4m+11+4p)n	(39+4p)n	79n	55n
	DICCG/ICCG			79/39=2	55/39=1.4

Table 25: Number of operations for the ICCG and DICCG methods for different sparsity of the matrices and different deflation vectors.