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

November 21, 2016

Abstract

We study fast and robust iterative solvers for large systems of linear equations resulting from reservoir simulation trough 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 is 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, the deflation vectors are 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 large variations in the permeability coefficients and the SPE 10 benchmark model.

1 Introduction.

Often, most computational time in the simulation of multi-phase flow through porous media is taken up by a 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 variations of permeability within the medium. 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 large-scale problems. However, sometimes the iterative methods are not sufficient to solve the problems in a reasonable amount of time and preconditioners become necessary to accelerate the convergence of these methods [1]. As the systems become larger or ill-conditioned, new preconditioning techniques need to be developed to improve the performance of an iterative method [2].

Reduced Order Models (ROM) have also been studied to improve computational efficiency by reducing the model size without losing essential information [3, 4, 5]. A potential ROM to reduce the computing time for large-scale problems is Proper Orthogonal Decomposition (POD), a method that has been investigated in [6, 7, 8, 9, 10, 11] among others. The use of a POD-based preconditioner for the acceleration of the solution is proposed by Astrid et al. [8] to solve the pressure equation resulting from two-phase reservoir simulation, and by Pasetto et al. [10] 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. [8] use as snapshots solutions of the pressure equation computed in a short number of pre-simulations previous to the actual simulation with diverse well configurations. The snapshots computed by Pasetto et al. [10] 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.

A similar approach is used in [9], where a set of POD-based basis vectors is obtained from the initial time steps. However, in this case, the acceleration is achieved by only improving the initial guess.

Problems with a high contrast between the permeability coefficients are sometimes approached through the use of deflation techniques; see, e.g., [12]. The use of deflation techniques involves the search of good deflation vectors, which are usually problem-dependent. In [12], subdomain based deflation vectors are used for layered problems with a large contrast between the permeability coefficients. However, these deflation vectors cannot be used if the distribution of the permeability coefficients is not structured as is the case in, e.g., the well-known SPE 10 benchmark problem [13].

Algebraic Multigrid (AMG)[14], Multi-level and Domain Decomposition [15] preconditioners have been studied in combination deflation techniques to accelerate the convergence of iterative methods. Following the ideas of [8, 9, 10, 11], 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 [8, 9] and [10], 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. ([11]) 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. In this work, instead of computing the solution in the low dimension subspace, the basis obtained with POD is studied as an alternative choice of deflation vectors to accelerate the convergence of the pressure solution in reservoir simulation.

2 Flow through porous media

Petroleum reservoirs are layers of sedimentary rock, which vary in terms of their grain size, mineral and clay contents. These rocks contain grains and empty space, the void is called pore space. The pore space allows the rock to store and transmit fluids. The volume fraction of the rock that is void is called *rock porosity* (ϕ).
Some rocks are compressible and their porosity depends on the pressure, the dependence is called *rock compressibility* (c_r). The ability of the rock to transmit a single fluid when the void space is completely filled with fluid is known as *rock permeability* (K).
Reservoir simulation is a way to analyze and predict the fluid behavior in a reservoir by the analysis of its behavior in a model. The description of subsurface flow simulation involves two types of models: mathematical and geological models. The geological model is used to describe the reservoir, i.e., the porous rock formation. The mathematical modeling of porous media flow is performed taking into account mass conservation and Darcy's law, corresponding to the momentum conservation. The equations used to describe single-phase flow through a porous medium are:

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho v) = q, \qquad v = -\frac{K}{\mu}(\nabla p - \rho g \nabla z), \tag{2.1}$$

or

$$\frac{\partial(\rho\phi)}{\partial t} - \nabla \cdot \left(\frac{\rho K}{\mu}(\nabla p - \rho g \nabla z) \right) = q. \tag{2.2}$$

Where the primary unknown is the pressure p, g is the constant of gravity, d is the reservoir depth, ρ and μ are the fluid density and viscosity and q are the sources. 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 the following expression:

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

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

$$\phi(p) = \phi_0 e^{c_r(p-p_0)}. \tag{2.3}$$

The fluid density and the pressure are related via the fluid compressibility c_f, the relation is given by:

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

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

$$\rho(p) = \rho_0 e^{c_f(p-p_0)}. \tag{2.4}$$

To solve Equation (2.2), it is neccesary to supply conditions on the boundary of the domain. 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 do not depend on the pressure in Equation (2.2), we have an incompressible model, where permeability and porosity do nott change over time. Therefore, the incompressible mode is not time-dependent. Assuming no gravity terms and a fluid with constant viscosity, Equation (2.2) becomes:

$$-\frac{\rho}{\mu}\nabla\cdot(K\nabla p) = q. \tag{2.5}$$

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 and j in the y direction and l in the z direction (x_i, y_j, z_l) and $p_{i,j,l} = p(x_i, y_j, z_l)$ is the pressure at this point. For the x direction, we have (see [16]):

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

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):

$$k_{i-\frac{1}{2},j,l} = \frac{1}{\frac{1}{k_{i-1,j,l}} + \frac{1}{k_{i,j,l}}}. \tag{2.6}$$

After discretization, Equation (2.5) can be written as:

$$\mathbf{Tp} = \mathbf{q}, \tag{2.7}$$

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

$$T_{i-\frac{1}{2},j,l} = \frac{\Delta y\Delta z}{\mu\Delta x}k_{i-\frac{1}{2},j,l}, \tag{2.8}$$

System (2.7) 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 (see Equation (2.4)). Therefore, Equations (2.1) become:

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\cdot(\rho(p)v) = q, \qquad v = -\frac{K}{\mu}(\nabla p - \rho(p)g\nabla d), \tag{2.9}$$

Discretization
Using backward Euler time discretization, Equations (2.9) are approximated by:

$$\frac{(\phi\rho(p))^{n+1}-(\phi\rho(p))^{n}}{\Delta t^{n}}+\nabla\cdot(\rho(p)v)^{n+1}=q^{n+1}, \qquad v^{n+1}=-\frac{K}{\mu^{n+1}}(\nabla(p^{n+1})-g\rho^{n+1}\nabla d). \tag{2.10}$$

Assuming no gravity terms, constant fluid viscosity and constant rock porosity, Equations (2.10) become:

$$\phi\frac{\rho(p^{n+1})-\rho(p^{n})}{\Delta t^{n}}-\frac{1}{\mu}\nabla\cdot(\rho(p^{n+1})K\nabla p^{n+1})+q^{n+1}=0. \tag{2.11}$$

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 (2.11) can be discretized in space, using a finite differences scheme. After discretization Equation (2.11) reads:

$$\phi\frac{\rho(\mathbf{p}^{n+1})-\rho(\mathbf{p}^{n})}{\Delta t^{n}}-\frac{1}{\mu}\nabla\cdot(\rho(\mathbf{p}^{n+1})\mathbf{K}\nabla\mathbf{p}^{n+1})+\mathbf{q}^{n+1}=0. \tag{2.12}$$

As in the incompressible case, we need to define boundary condition to solve Equation (2.12). 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 neccesary 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 at constant surface rate or constant bottom-hole pressure (bhp) and are produced at constant bhp or a constant surface rate.
When the bhp is known, some models are developed to accurately compute the flow rate into the wells. A widely used model is Peaceman's 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 surface flow rate in a well, for a cell (i,j,l) that contains a well, this relationship is given by:

$$q_{(i,j,l)}=I_{(i,j,l)}(p_{(i,j,l)}-p_{bhp(i,j,l)}), \tag{2.13}$$

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 $p_{bhp(i,j,l)}$ is a prescribed pressure inside the well.

Incompressible fluid
Using the well model for an incompressible fluid, Equation (2.7) transforms into:

$$\mathbf{T}\mathbf{p}=\mathbf{I}_w(\mathbf{p}-\mathbf{p}_{bhp}). \tag{2.14}$$

Where $\mathbf{I}_w$ is a vector containing the productivity or injetivity indices of the wells present in the reservoir. It is zero for cells without wells and the value of the well index for each cell containing a well.

Compressible fluid
For a compressible fluid, using the well model, Equation (2.12) reads:

$$\phi\frac{\rho(\mathbf{p}^{n+1})-\rho(\mathbf{p}^{n})}{\Delta t^{n}}-\frac{1}{\mu}\nabla\cdot(\rho(\mathbf{p}^{n+1})\mathbf{K}\nabla\mathbf{p}^{n+1})+\mathbf{I}_{w}^{n+1}(\mathbf{p}^{n+1}-\mathbf{p}_{bhp}^{n+1})=0. \tag{2.15}$$

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 the time step $n+1$. Therefore, Equation (2.15) can be seen as a function that depends on $\mathbf{p}^{n+1}$ and $\mathbf{p}^{n}$, i.e.,

$$\mathbf{F}(\mathbf{p}^{n+1};\mathbf{p}^{n})=0. \tag{2.16}$$

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

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

where $\mathbf{J}(\mathbf{p}^{k};\mathbf{p}^{n})=\frac{\partial\mathbf{F}(\mathbf{p}^{k})}{\partial\mathbf{p}^{k}}$ 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:

$$\mathbf{J}(\mathbf{p}^{k})\delta\mathbf{p}^{k+1}=\mathbf{b}(\mathbf{p}^{k}). \tag{2.17}$$

with $\mathbf{b}(\mathbf{p}^{k})$ being the function evaluated at iteration step k, $\mathbf{b}(\mathbf{p}^{k})=-\mathbf{F}(\mathbf{p}^{k};\mathbf{p}^{n})$.
The procedure to solve a compressible flow problem consists of three stages. During the first stage, we select a time and solve Equation (2.15) for this particular time, i.e., we have a solution for each time step. In the second stage, we linearize the equations with the NR method, i.e., we perform a series of iterations to find the zeros of Equation (2.16). For every NR iteration the linear system in Equation (2.17) is solved. 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 $\mathbf{F}(\mathbf{p}^{n+1}; \mathbf{p}^n) = 0$
        for $lin\_iter = 0, ...,$                        %Linear iteration
            Solve $\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1} = \mathbf{b}(\mathbf{p}^k)$ 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

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.1}$$

for both compressible and incompressible models. Since $\mathbf{A}$ is SPD we choose as iterative method the Conjugate Gradient (CG) method accelerated by the Incomplete Cholesky preconditioner. In this work, we will 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:

$$||\mathbf{x} - \mathbf{x}^{k+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}^0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} \right)^{k+1} \text{[1]}. \tag{3.2}$$

The pseudo code for CG is given in Algorithm 2.

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

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

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

Preconditioning

To accelerate the convergence of 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 (3.1) by a matrix $\mathbf{M}^{-1}$.

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}. \tag{3.3}$$

The new system has the same solution but can provide a substantial improvement on the spectrum. For this preconditioned system, the error is bounded by:

$$||\mathbf{x} - \mathbf{x}^k||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}^0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{A})} + 1} \right)^k . \tag{3.4}$$

$\mathbf{M}$ 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 ([12]). 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 ([17, 15]):

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

where

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

with

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

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.
Some properties of the previous matrices are [15]:

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

We can split the vector $\mathbf{x}$ as:

$$\mathbf{x} = \mathbf{x} - \mathbf{P}^T\mathbf{x} + \mathbf{P}^T\mathbf{x} = (\mathbf{I} - \mathbf{P}^T)\mathbf{x} + \mathbf{P}^T\mathbf{x}. \tag{3.5}$$

Multiplying expression (3.5) by $\mathbf{A}$, using the properties above, 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:

$$\begin{aligned} \mathbf{P}\mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{P}^2\mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{b}, \\ \mathbf{P}\mathbf{A}\mathbf{x} &= \mathbf{P}\mathbf{b}, \end{aligned} \tag{3.6}$$

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 (3.6) can contain components of the null space of $\mathbf{P}\mathbf{A}$. A solution of this system, called the deflated solution, is denoted by $\hat{\mathbf{x}}$. The deflated system for $\hat{\mathbf{x}}$ is:

$$\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}. \tag{3.7}$$

The relation between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is (see Appendix D):

$$\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^T\hat{\mathbf{x}}. \tag{3.8}$$

Deflated CG Method
To obtain the solution of linear system (3.1), we solve the deflated system:

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

with the CG method, for a deflated solution $\hat{\mathbf{x}}$. Thereafter, the solution $\mathbf{x}$ of the original system is obtained from Equation (3.8):

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

Deflated PCG Method

The deflated linear system can also be preconditioned by an SPD matrix $\mathbf{M}$. The deflated preconditioned system to solve with CG is [15]:

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

where:

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

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

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

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

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 [18], subdomain deflation vectors [2] or multigrid and multilevel based deflation techniques [15, 19]. 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 [18]. 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 [2].

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 $\mathbf{Z}$ [15].

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 [8]:

$$\mathbf{x} \approx \sum_{i=1}^{l} c_i \psi_i, \tag{4.1}$$

where $\psi_i \in \mathbb{R}^n$ are the basis vectors and c_i are their corresponding coefficients. In matrix notation, equation (4.1) 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 [9]. 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 $\mathbf{R}$.

$$\mathbf{R} := \frac{1}{m}\mathbf{X}\mathbf{X}^T \equiv \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^T, \qquad \mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_m], \tag{4.2}$$

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 [9]:

$$\frac{\sum_{j=1}^{l} \sigma_j}{\sum_{j=1}^{m} \sigma_j} \leq \alpha, \qquad 0 < \alpha \leq 1, \tag{4.3}$$

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 $||\mathbf{x}_i||_2 = 1$.

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 the 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:

$$\mathbf{Ax} = \mathbf{b}. \tag{5.1}$$

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

$$\mathbf{Ax}_i = \mathbf{b}_i. \tag{5.2}$$

The following equivalence holds

$$\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i \qquad \Leftrightarrow \qquad \mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{b}_i. \tag{5.3}$$

Proof $\Rightarrow$

$$\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i \Rightarrow \mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{b}_i. \tag{5.4}$$

Substituting $\mathbf{x}$ from (5.4) into $\mathbf{Ax} = \mathbf{b}$ leads to:

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

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

$$\mathbf{A}c_1\mathbf{x}_1 + ... + \mathbf{A}c_m\mathbf{x}_m = c_1\mathbf{b}_1 + ... + c_m\mathbf{b}_m = \mathbf{Bc}. \tag{5.5}$$

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 (5.1) and (5.5) we get:

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

Proof $\Leftarrow$

$$\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i \Leftarrow \mathbf{b} = \sum_{i=1}^{m} c_i \mathbf{b}_i. \tag{5.6}$$

Substituting $\mathbf{b}$ from (5.6) into $\mathbf{Ax} = \mathbf{b}$ leads to:

$$\mathbf{Ax} = \sum_{i=1}^{m} c_i \mathbf{b}_i. \tag{5.7}$$

Since $\mathbf{A}$ is non-singular, multiplying (5.2) and (5.6) 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

$$\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i. \tag{5.8}$$

☒

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

$$\mathbf{Z} = \begin{bmatrix} \mathbf{x}_1 & ... & ... & \mathbf{x}_m \end{bmatrix}, \tag{5.9}$$

such that $\mathbf{x} = \sum_{i=1}^{m} c_i \mathbf{x}_i$, with $\mathbf{x}_i$ *l.i.*, then the solution of system (5.1) is obtained with one iteration of DCG.

Proof.

The relation between $\hat{\mathbf{x}}$ and $\mathbf{x}$ is given in Equation (3.8):

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

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

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

Therefore,

$$\mathbf{x} = \mathbf{Qb}, \tag{5.10}$$

is the solution to the original system.
For the second term of Equation (3.8), $\mathbf{P}^T\hat{\mathbf{x}}$, we compute $\hat{\mathbf{x}}$ from Equation (3.7):

$$\begin{aligned}
\mathbf{PA}\hat{\mathbf{x}} &= \mathbf{Pb} \\
\mathbf{AP}^T\hat{\mathbf{x}} &= (\mathbf{I} - \mathbf{AQ})\mathbf{b} \qquad \text{using 3 f) and definition of } \mathbf{P}, \\
\mathbf{AP}^T\hat{\mathbf{x}} &= \mathbf{b} - \mathbf{AQb} \\
\mathbf{AP}^T\hat{\mathbf{x}} &= \mathbf{b} - \mathbf{Ax} = 0 \qquad \text{taking } \mathbf{Qb} = \mathbf{x} \text{ from above}, \\
\mathbf{P}^T\hat{\mathbf{x}} &= 0 \qquad \text{as } \mathbf{A} \text{ is invertible.}
\end{aligned}$$

Then we have obtain the solution

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

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 $\mathbf{Ax} = \mathbf{b}$, we cannot compute the relative error e_r (Equation (5.11)) of the approximation with respect to the true solution because the true solution is unknown,

$$e_r = \frac{||\mathbf{x} - \mathbf{x}^k||_2}{||\mathbf{x}||_2}. \tag{5.11}$$

Instead, we compute the relative residual r_r (Equation (5.12)),

$$r_r = \frac{||\mathbf{r}^k||_2}{||\mathbf{b}||_2} \leq \epsilon, \tag{5.12}$$

and we set a stopping criterium ϵ or tolerance, that is related to the relative error as follows [20] (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 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 (5.8), 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 $i-th$ 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 (5.10)):

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

If the approximation of $\mathbf{x}$ 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 $\mathbf{x} \in \mathit{span}\{\mathbf{x}_1, ..., \mathbf{x}_m\}$, 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 $\mathbf{b} \perp Ker(\mathbf{A})$ [21]. 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.

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 linear system is performed with the Conjugate Gradient method preconditioned with Incomplete Cholesky (ICCG) and the Deflated Conjugate Gradient method preconditioned with Incomplete Cholesky (DICCG). We propose the use of snapshots and the snapshots-based POD basis vectors as deflation vectors for the above-mentioned method.
In the present section, we give a general overview of the experiments we perform, but the specifications are presented below for each problem separately. We solve the elliptic problem (incompressible flow) and the parabolic problem (compressible flow). Neumann boundary conditions (no-flux) are imposed for an academic layered problem and for the SPE 10 benchmark problem.

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 single-phase model the following properties of the fluid are used:

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

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

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

The matrices corresponding to the linear systems $\mathbf{A}$ and right-hand sides $\mathbf{b}$ are obtained with the Matlab Reservoir Simulation Toolbox (MRST) [22].

Snapshots
As mentioned before, 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. These solutions, called *snapshots*, are obtained with the ICCG method. For the compressible problem, the snapshots are the solutions of the previous time steps with the same well configuration. We also propose the use of a linearly independent basis of POD as deflation vectors, obtained from the previously computed snapshots for the incompressible and compressible cases. 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 or deflated system:

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

6.2 Incompressible Problem

We simulate single-phase flow through a porous medium, for an incompressible fluid (see Equation (2.7)), with the previously mentioned characteristics. Homogeneous Neumann boundary conditions are imposed on all boundaries.

A set of four linearly independent snapshots is used as deflation vectors. We also use a linearly dependent set of 15 snapshots and a basis of POD (linearly independent set) obtained from these 15 snapshots. We set the same boundary conditions as in the original problem for all the snapshots. The four linearly independent snapshots ($\mathbf{z}_1$-$\mathbf{z}_4$) 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 equals 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. The snapshots and the solutions are obtained with a tolerance of 10^{-11}.

A summary of the configurations is presented in Table 1.

System configuration					
Well pressures (bars)					
	$W1$	$W2$	$W3$	$W4$	$W5$
	-1	-1	-1	-1	4
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 1: Table with the well configuration of the system and the snapshots.

Heterogeneous permeability layers

A Cartesian grid of 64×64 grid cells and length (Lx,Ly) of $70 \times 70m^2$ with 8 layers of the same size is studied. Four layers have a permeability σ_1 and they are followed by a layer with a different permeability value σ_2 (see Figure 1). 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 in permeability value between the layers for the ICCG and DICCG methods. The permeability σ_2 varies from $\sigma_2 = 10^{-1}mD$ to $\sigma_2 = 10^{-3}mD$.

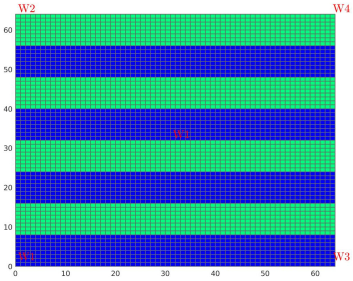

Figure 1: Heterogeneous permeability, 5 wells.

Four wells are positioned in the corners with a bhp (bottom hole pressure) of -1 bar. One well is positioned in the center of the domain and has a bhp of +4 bars (see Figure 1).

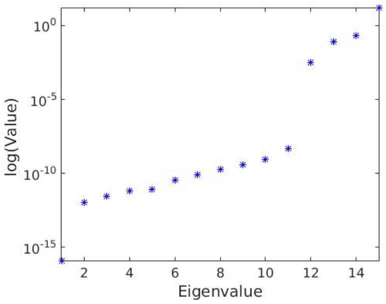

Figure 2: Eigenvalues of the snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, if 15 snapshots are used.

Table 2 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 2. We observe that there are 4 eigenvalues much larger than the rest, these largest eigenvalues are responsible for the divergence of the method. For the DICCG$_{POD}$ method we use the eigenvectors corresponding to the larger eigenvalues as deflation vectors.
In Table 2, 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.

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

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 2: Number of iterations for different contrast in the permeability of the layers for the ICCG and DICCG methods.

SPE 10 model

This model has large variations in the permeability coefficients, the contrast between coefficients is 3×10^7. The model contains 60 x 220 x 85 cells (Figure 3) and five wells positioned as in the previous experiment. Snapshots are obtained solving the system with different well configurations (see Table 1). As before, we simulate single-phase incompressible flow.
The number of iterations required to achieve convergence with the ICCG and DICCG methods for various grid sizes is presented in Table 3. In this table, we observe that for the ICCG method we require 1011 iterations to reach the desired accuracy. Meanwhile, for the deflated methods DICCG$_4$ and DICCG$_{POD}$ only two iterations are required. In theory, only one iteration is necessary to reach the solution with the deflated methods. However, the large contrast in the permeability field may require higher accuracy for the snapshots to find the solution with deflation within one iteration within the imposed tolerance [23]. In this case, the first iteration has a relative residual smaller than 10^{-10} for the DICCG$_4$ and DICCG$_{POD}$ methods, therefore after a first iteration we have a good approximation.

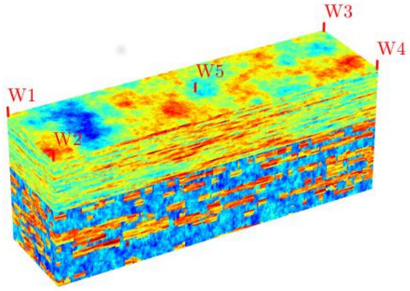

Figure 3: SPE 10 benchmark, permeability field.

Method	Iterations
ICCG	1011
DICCG$_{15}$	2000*
DICCG$_4$	2
DICCG$_{POD}$	2

Table 3: Table with the number of iterations for ICCG and DICCG methods.

For the deflated method with 15 linearly dependent snapshots (DICCG$_{15}$), we observe that the desired accuracy is not reached after 2000 iterations, as the deflation methods are linearly dependent, the deflation method is unstable because the matrix E is a nearly singular matrix (new report).

6.3 Compressible Problem

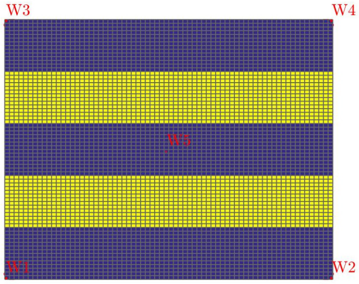

Figure 4: Heterogeneous permeability, 5 wells, compressible problem.

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 (2.4). We solve Equation (2.12) for a fluid with the a compressibility of $c = 1 \times 10^{-3}$. Equation (2.12) is non-linear due to the dependence of the density on the pressure. Therefore, we need to linearize this equation via the Newton-Raphson method and solve the resulting linear system. After linearization, we obtain the linear system (2.17) 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 [22]. 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 4). The first layer has a permeability of $\sigma_1 = 30mD$, and the permeability of the second layer is varied $\sigma_2 =$ [3mD, 0.3mD, 0.03mD]. 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 the grid size is 35 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 bottom hole pressure (bhp) in the corner wells is 100 bars and in the central well is 600 bars. 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}.

In Figure 5, the solution obtained with the ICCG method is presented for a contrast in permeability layers of 10^{-1}. 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 (1,1) 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.

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 6 for the 20^{th} time

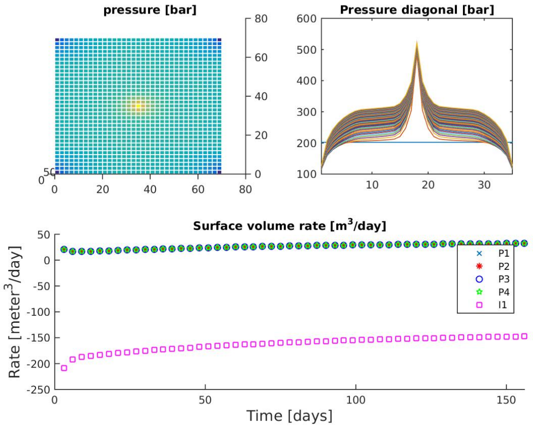

Figure 5: Solution of the compressible problem solved with the ICCG method for a layered problem with a contrast between permeability layers of 10^{-1}.

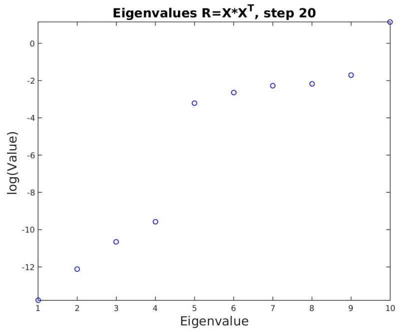

Figure 6: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, contrast between permeability layers of 10^{-1}.

step. In Figure 6, we observe that six eigenvalues are larger than the rest. Then, we use the eigenvectors corresponding to these six eigenvalues as deflation vectors (DICCG$_{POD_6}$). For a contrast between permeability layers of 10^{-2} and 10^{-3} we have 7 larger eigenvalues, therefore, we use 7 POD basis vectors as deflation vectors (DICCG$_{POD_7}$). For all the experiments, only the first time step requires more than two NR iterations. Hence, we solely study the behavior of the linear solvers during the first two NR iterations.
In Table 4 and Table 5 we compare the number of iterations necessary to reach convergence with the ICCG method and the deflation methods DICCG$_{10}$, DICCG$_{POD_6}$ and DICCG$_{POD_7}$.

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$_{POD_6}$			84	224	29
10^{-2}	DICCG$_{10}$	624	100	42	142	23
	DICCG$_{POD_7}$			42	142	23
10^{-3}	DICCG$_{10}$	364	20	42	62	17
	DICCG$_{POD_7}$			42	62	17

Table 4: 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.

For the first NR iteration (see Table 4), 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 for the ICCG method, 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 (computed with ICCG) and 42 and 84 for the 42 remaining time steps computed with DICCG$_{10}$ and DICCG$_{POD_6}$. 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 only the ICCG method.
When we have a contrast in permeability of 10^{-2}, the required average of linear iterations to solve the 52 time steps is 624 for the ICCG method. With the deflated methods, taking into account the computation of the snapshots, we require 142 iterations for the DICCG$_{10}$ and DICCG$_{POD_7}$ methods, which is the 23% of the ICCG iterations. Finally, for a contrast between permeability layers of 10^{-3} we require 364 linear iterations for the 52 time steps with the ICCG method. Meanwhile the required iterations for the DICCG$_{10}$ and DICCG$_{POD_7}$ methods is 62 . That is 17% of the ICCG iterations (see Table 4).

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$_{POD_6}$			198	378	38
10^{-2}	DICCG$_{10}$	832	140	90	230	28
	DICCG$_{POD_7}$			154	294	33
10^{-3}	DICCG$_{10}$	884	110	90	200	23
	DICCG$_{POD_7}$			150	260	29

Table 5: 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.

For the second NR iteration (see Table 5), 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 methods, we need 180 linear iterations to compute the snapshots during the first ten time steps with the ICCG method and 78 and 198 for the 42 remaining time steps with the DICCG$_{10}$ and DICCG$_{POD_6}$ 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 5). 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$_{10}$ and 294 for the DICCG$_{POD_7}$ method. That means the 28% and 33% of the ICCG iterations. For a contrast between permeability layers of 10^{-3}, we require 884 linear iterations for the 52 time steps. For the DICCG$_{10}$ and DICCG$_{POD_7}$ methods, we require 200 and 260 iterations. That is 23% 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.

SPE 10 model

We study the complete SPE 10 benchmark that consists of $60 \times 220 \times 85$ grid cells and has a contrast in permeability of 3×10^7. 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.
In Figure 7 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 in Figure 7.

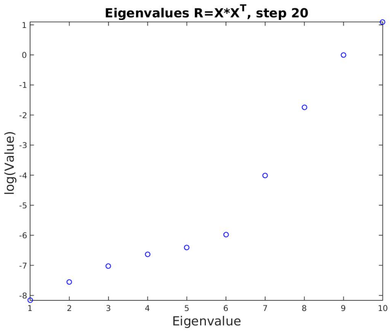

Figure 7: Eigenvalues of the data snapshot correlation matrix $\mathbf{R} = \mathbf{X}\mathbf{X}^T$, time step 20, full SPE 10 benchmark.

1^{st} NR Iteration					
Method	Total ICCG (only)	ICCG Snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
DICCG$_{10}$			1134	2904	28
DICCG$_{POD_4}$	10173	1770	1554	3324	32

Table 6: Average number of linear iterations for the first NR iteration, full SPE 10 benchmark.

1^{st} NR Iteration					
Method	Total ICCG (only)	ICCG Snapshots	DICCG	Total ICCG+DICCG	% of total ICCG
DICCG$_{10}$	10231	1830	200	2030	20
DICCG$_{POD_4}$			200	2030	20

Table 7: Average number of linear iterations for the second NR iteration, full SPE 10 benchmark.

For the first NR iteration, we observe that the average number of iterations required for the ICCG method is considerably reduced. For the ICCG method we require 10173

iterations for the first NR iteration and 10231 for the second (see Table 6 and Table 7). With the deflated methods DICCG$_{10}$ and DICCG$_{POD_4}$, for the first NR iteration, we only need to perform 28% and 32% of the linear iterations required with the ICCG method.
For the second NR iteration, the deflated methods require only 20% of the ICCG linear iterations. We also observe that for the first NR iteration we need 1770 linear iterations to compute the ten initial snapshots (computed with ICCG) and 1134 to compute the solution of the rest of the solutions (computed with DICCG). For the second NR iteration, the number of linear iterations is 1830 for the ten initial snapshots and 200 for the deflated methods. This shows that the largest amount of work is carried out for the computation of the snapshots, which is more evident for the second NR iteration.

Conclusions

In this work, we combine preconditioning with deflation and POD methods to accelerate the convergence of iterative methods for large systems and systems with high-contrast in permeability. In particular, we study the Conjugate Gradient (CG) method preconditioned with Incomplete Cholesky. However, 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. 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.
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 complete SPE 10 benchmark.
To solve the incompressible problem, we propose the use of solutions of the problem with different well configurations as deflation vectors. We observe that the number of linear iterations required with ICCG is reduced to only a few iterations with DICCG and this number is independent of the contrast in permeability for the deflation methods. Results also 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. 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 the compressible case, we propose the use of solutions of previous time steps, *snapshots*, and POD basis vectors computed from this snapshots as deflation vectors. With DICCG we reduce the number of iterations necessary to achieve convergence to around 25% of the number of iterations necessary for the ICCG method. We observe that the performance of the DICCG method with snapshots and POD basis vectors as deflation vectors is similar. The required number of POD basis vectors to achieve a good acceleration of the method depends on the problem. However, only a limited number of deflation vectors is necessary to obtain a good speed-up (seven at most for the problems here studied). For the SPE 10 problem, we reduce the number of deflation vectors to 4 POD basis vectors, which can represent an important reduction in the number of operations required for the method.

References

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

[2] 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.

[3] 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.

[4] M. Barone; I.Kalashnikova; M. Brake and D. Segalman. Reduced order modeling of fluid/structure interaction. *Sandia National Laboratories Report, SAND No*, 7189, 2009.

[5] I. Kalashnikova; S. Arunajatesan; M.F. Barone; B.G. van Bloemen Waanders and J.A. Fike . Reduced order modeling for prediction and control of large-scale systems. *Sandia National Laboratories Report, SAND*, (2014-4693), 2014.

[6] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical analysis*, 40(2):492–515, 2002.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] Carlberg, Kevin and Forstall, Virginia and Tuminaro, Ray. Krylov-subspace recycling via the POD-augmented conjugate-gradient algorithm. *arXiv preprint arXiv:1512.05820*, 2015.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

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

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

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

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

[23] 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
α	Geometric factor	
ρ	Fluid density	kg/m^3
μ	Fluid viscosity	$Pa \cdot s$
p	Pressure	Pa
g	Gravity	m/s^2
d	Reservoir depth	m
c_l	Liquid compressibility	Pa^{-1}
q	Sources	

Table 8: 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 [20, 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
$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T, \qquad \mathbf{U} \in \mathbb{R}^{n\times n}, \qquad \mathbf{\Sigma} \in \mathbb{R}^{n\times m}, \qquad \mathbf{V} \in \mathbb{R}^{m\times m}.$
Then we have

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

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

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

As mentioned above, the solution to Equation (3.6) can contain components of $\mathcal{N}(\mathbf{PA})$. Therefore, a solution to Equation (3.7), $\hat{\mathbf{x}}$ can be decomposed as:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{y}, \tag{D.1}$$

with $\mathbf{y} \in \mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{PA})$, and $\mathbf{x}$ the solution to Equation (3.1).
Note: If $\mathbf{y} \in \mathcal{R}(\mathbf{Z})$, then

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

$$\mathbf{PA}y = \mathbf{PA}(\mathbf{z}_1\alpha_1 + ... + \mathbf{z}_m\alpha_m) = \mathbf{PAZ}\alpha, \tag{D.2}$$

from 3 h) $\mathbf{PAZ} = \mathbf{0}^{n\times l}$, then

$$\mathbf{PAy} = \mathbf{0}. \tag{D.3}$$

Therefore $\mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{PA})$.
Multiplying Equation (D.1) by $\mathbf{P}^T$ we obtain:

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

combining Equation (D.3) with 3 f), we have:

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

Therefore

$$\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x}. \tag{D.4}$$

Substitution to Equation (D.4) and 3 g) in Equation (3.5) leads to:

$$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T\hat{\mathbf{x}}, \tag{D.5}$$

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