Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Comparison of Two-Level Preconditioners using Deflation Techniques applied to Flow Problems.

Report for the
Delft Institute of Applied Mathematics
as part of

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

Jenny Tjan

Delft, Netherlands
August 2018

Comparison of Two-Level Preconditioners using Deflation Techniques applied to Flow Problems.

by

Jenny Tjan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 30, 2018 at 11:00 AM.

Student number:	4321375	
Project duration:	November 27, 2017 – August 30, 2018	
Thesis committee:	Prof. dr. ir. C. Vuik,	Delft University of Technology, supervisor
	MSc. G.B. Diaz Cortes,	Delft University of Technology, supervisor
	Dr. D. Pasetto,	Swiss Federal Institute of Technology
	Dr.ir. H. Schuttelaars,	Delft University of Technology

TU Delft

Abstract

We investigate the simulation of one-phase and two-phase flow through heterogeneous porous media. The derived matrix, resulting from reservoir simulation of groundwater flow problems, can result in a large and ill-conditioned system, i.e. the matrix has a high condition number, and the modelling takes large computation time. In this thesis report, the Two-Level Preconditioned Conjugate Gradient method with deflation techniques will be considered and have been investigated by [25, 30]. Recently, new Two-Level preconditioners are constructed using the AMG approach [21]. In this research we compare this approach with the standard Two-Level preconditioners. We found that the performance of these methods can be improved by using a special starting vector and previous time-step as initial condition. From the results we see that the performance and the memory storage of the methods are similar. However, the cheapest methods per iteration resulted DEF1, DEF2, R-BNN2, and ROM.

Acknowledgements

This thesis project is a collaboration between Delft University of Technology and University of Padova, whom I like to refer to as *our Italian friends*. I have been working on this thesis for 9 months and I don't want to brag or anything, but this report looks pretty good. This would not have been possible on my own and I want to use this part of my thesis to thank them all.

I remember my first meeting with Kees, one of my supervisors. After I told him that I am only an ordinary student, he told me that after a few years I will look back and be amazed by what I have done. To be honest, I can say that I am already amazed by looking at this report. This kind of positivity is what I have received during those months. I want to thank both of my supervisors, Kees and Gabriela, for their patience, enthusiasm, feedback, putting up with my bad humour, and encouraging me.

NEXT, I want to thank our Italian friends, especially Damiano, for being part of the research. It was great meeting you all in France and sharing the results of my research. As an inexperience grasshopper, this feels like I am really part of the big people's world of research.

THEN, I would like to thank Anne, Roel and Mike, who have taken the time and effort to proofread my thesis. Still, I believe Gabi should get the crown for this job.

FINALLY, I want to thank my parents and friends. My parents for the amazing food and making sure I am staying healthy. My friends for listening to my continuous rambling and struggles about my thesis. Special shoutout to Merel, Marieke, Roy and Amey for the food and coffee breaks. Especially my lovely sister Joanne, who only understands a small part of my research and still listens to help whenever she can and wherever she is. For the interested, the flops can be found in Appendix C.

Thank you for reading and now it is time to end this acknowledgement with a cheesy quote:

'Whatever good things we build end up building us.'

Jenny Tjan,
Rotterdam, August 2018

Contents

1 Introduction

Oil production starts with extracting crude oil from underground reservoirs. If the oil reservoir has been found, one of the techniques to extract as much as possible is to pump water in the reservoir to maintain the pressure of the production wells. This is an example of a two-phase flow through porous media studied by [6, 7, 21, 28], among others. This can be described with mathematical models and solved to find the solution. There has been great interest to model this phenomenon.

This report investigates the simulating of one-phase and two-phase flow through heterogeneous porous media. For this simulation, a geological model for the porous media and the mathematical model of the flow are needed. The result of the mathematical model of the flow problem can be linearized in a system of linear equations that can be written in the form:

$$\mathbf{Ax} = \mathbf{b}. \tag{1.1}$$

This system can be solved using e.g. iterative solvers. The derived matrix **A**, resulting from reservoir simulation or groundwater flow problems, can result in a large and ill-conditioned system, i.e. the condition number of the matrix **A** is high. Therefore, solving the system 1.1 takes a large computation time.

The iterative solvers can be improved using deflation techniques. In this thesis report, the method Two-Level Preconditioned Conjugate Gradient will be considered to solve this system. The flow problem can be reformulated in a general system:

$$\mathcal{P}\mathcal{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}, \quad \mathcal{P}, \mathcal{A} \in \mathbb{R}^{n \times n} \tag{1.2}$$

Where $\mathcal{P}$ is a traditional combination of an SPD preconditioner **M** and a correction matrix **Q**. The matrix $\mathcal{A}$ is a combination of deflation matrix **P** and the original matrix **A**. For this method, a set of deflation vectors are needed to construct Two-Level preconditioners. The choices for deflation vectors are subdomain vectors and eigenvectors.

Recently, Proper Orthogonal Decomposition (POD) based on known information has been found to be a good approach to accelerate the solving process [4, 21, 28, 30]. The POD method requires a set of snapshots, i.e. solutions of the linear system $\mathbf{Ax} = \mathbf{b}$, to construct POD basis vectors. These basis vectors are used as deflation vectors.

Research Question

The performance of the deflation techniques can be optimized by choosing the right deflation vectors. This will help to reduce the number of iterations and the computation time for solving. The deflation methods have been investigated by [25, 30]. Recently, new Two-Level preconditioners were constructed by using the AMG approach [21]. In this report, we want to find the similarities and differences between the new Two-Level preconditioners with the standard Two-Level preconditioners.

Outline of this Report

The outline of this thesis report is as follows. First, the preliminaries are given in Chapter 2. Secondly, a brief overview of the mathematical model of reservoir simulation is given in Chapter 3. Then, a small introduction is given to the iterative solvers and deflation techniques in Chapter 4. Thereafter, in Chapter 5, several choices of deflation vectors are presented that are used as deflation methods. Continuing, the Two-Level preconditioners method is presented in Chapter 6. In Chapter 7, theoretical results about the Two-Level preconditioners are presented. Further, we present in Chapter 8 three test cases that are used to perform numerical experiments in Chapter 9. Finally, the conclusion and discussion are given in Chapter 10.

2

Preliminaries

This section gives a brief introduction of linear algebra theory that will be used in this thesis report.

2.1. Notation

The column vector $\mathbf{x} \in \mathbb{R}^n$ will be denoted as

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \tag{2.1}$$

The matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ will be denoted as

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix}. \tag{2.2}$$

2.2. Definition

Definition 2.2.1. *Let* $\mathbf{A}$ *be an* $n \times n$ *matrix.* λ *is called an eigenvalue of* $\mathbf{A}$ *if there exists an* $\mathbf{v} \neq 0$ *such that*

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}. \tag{2.3}$$

The set of eigenvalues of $\mathbf{A}$ *is given by*

$$\sigma(\mathbf{A}) = \{\lambda_1, \dots, \lambda_n\}, \tag{2.4}$$

where λ_i *is an eigenvalue of* $\mathbf{A}$.

Definition 2.2.2. *Let* $\mathbf{A}$ *be an* $n \times n$ *matrix,* $\mathbf{A}$ *is called symmetric positive definite (SPD) if for every* $\mathbf{x} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0. \tag{2.5}$$

$\mathbf{A}$ *is called symmetric positive semi definite (SPSD) if for every* $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0. \tag{2.6}$$

Definition 2.2.3. *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*, the inner product is defined as*

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}. \tag{2.7}$$

Definition 2.2.4. *Let* $\mathbf{A}$ *be an* $n \times n$ *matrix, the 2-norm is defined as*

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{\max}(\mathbf{A}^\top \mathbf{A})}. \tag{2.8}$$

Definition 2.2.5. *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*, and* $\mathbf{A}$ *is SPD, the* $\mathbf{A}$*-norm and* $\mathbf{A}$*-inner product is defined respectively as*

$$\|\mathbf{x}\|_{\mathbf{A}} = \sqrt{\langle \mathbf{Ax}, \mathbf{x} \rangle} \quad \text{and} \quad \langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{A}} = \langle \mathbf{Ax}, \mathbf{y} \rangle. \tag{2.9}$$

Definition 2.2.6. *Let* $\mathbf{A}$ *be an* $n \times n$ *matrix with eigenvalues* $\lambda_1, \ldots, \lambda_n$*. The condition number of* A *is defined as*

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2. \tag{2.10}$$

If $\mathbf{A}$ *is SPD with real eigenvalues* $\lambda_1, \ldots, \lambda_n$*, then*

$$\kappa_2(\mathbf{A}) = \frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}, \tag{2.11}$$

where $\lambda_{\max}(\mathbf{A}) = \max_{1 \le i \le n} \lambda_i$ *and* $\lambda_{\min}(\mathbf{A}) = \min_{1 \le i \le n} \lambda_i$*.*

2.3. Lemma

In this section, a lemma from linear algebra will be given that are used to prove theorems in Section 7.3.2. Assume $\mathbf{A} \in \mathbb{R}^{n \times n}$ has spectrum

$$\sigma(\mathbf{A}) = \{\lambda_1, \ldots, \lambda_n\}. \tag{2.12}$$

Then, we have the following Lemma:

Lemma 2.3.1. *Let* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ *be arbitrary matrices. Now, the following equalities hold:*

(a) $\sigma(\mathbf{AB}) = \sigma(\mathbf{BA})$ *,*

(b) $\sigma(\mathbf{A} + \alpha\mathbf{I}) = \sigma(\mathbf{A}) + \alpha\sigma(\mathbf{I})$*, where* $\alpha \in \mathbb{R}$*,*

(c) $\sigma(\mathbf{A}) = \sigma(\mathbf{A}^\top)$*.*

Proof. (a) Let $\mathbf{v}$ be an eigenvector corresponding to eigenvalue λ of matrix $\mathbf{AB}$. Two cases will be considered and proven separately.

(i) If $\lambda = 0$, then

$$\det(\mathbf{AB}) = \det(\mathbf{BA}) = 0. \tag{2.13}$$

It follows that $\lambda = 0$ is an eigenvalue of $\mathbf{BA}$.

(ii) If $\lambda \neq 0$, then

$$\mathbf{ABv} = \lambda\mathbf{v} \tag{2.14}$$
$$\mathbf{BABv} = \lambda\mathbf{Bv} \tag{2.15}$$
$$\mathbf{BAw} = \lambda\mathbf{w}, \tag{2.16}$$

where $\mathbf{w} = \mathbf{Bv} \neq 0$.

It follows that λ is an eigenvalue of $\mathbf{BA}$.

(b) Let $\mathbf{v}$ be an eigenvector corresponding to eigenvalue λ of matrix $\mathbf{A}+\alpha\mathbf{I}$, where $\alpha \in \mathbb{R}$. Next, we get

$$(\mathbf{A} + \alpha\mathbf{I})\mathbf{v} = \lambda\mathbf{v} \tag{2.17}$$
$$\mathbf{A} = (\lambda - \alpha)\mathbf{v} \tag{2.18}$$
$$\mathbf{A} = \mu\mathbf{v}, \tag{2.19}$$

where $\mu = (\lambda - \alpha)$. Clearly, if λ is an eigenvalue of $\mathbf{A}+\alpha\mathbf{I}$, then $\lambda+\alpha$ is an eigenvalue of $\mathbf{A}$.

(c) It follows from the definition of determinants that

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \det(\mathbf{A}^\mathsf{T} - \lambda\mathbf{I}) \tag{2.20}$$

for all λ.

□

3

Reservoir Simulation

Two models are needed to describe phase flow through porous media, these are the mathematical model and the geological model [13]. The mathematical model consists of a set of partial differential equations to describe flow through porous media [8, 13]. The equations are derived from for the mass-conservation law and Darcy's law, which will be further explained in Section 3.2.1. The geological model describe the rock formation using rock porosity and rock permeability, more will be explained in Section 3.1.

3.1. Porous Media

The geological model describes the porous media rock formation, and it is constructed such that it reproduces geological heterogeneity in the reservoir rock. The rock formation is defined by rock porosity φ, i.e. the volume fraction of the pores, and the rock permeability $\mathbf{K}$, i.e. the ability to transport fluid. The porosity φ is defined as the percentage of void in the porous media and $1 - \varphi$ is the percentage of solid material, i.e. rock matrix. There are interconnected pore spaces in the porous media where fluid can flow through and disconnected pores where fluid can only be stored. Since the disconnected pores do not contribute to the flow, the effective porosity that will be considered is associated with connected pores where fluid can flow through. The rock permeability $\mathbf{K}$ describes the basic flow of porous media and it measures its ability to transmit a single fluid when the void space is filled with the fluid. Mathematically, the ability of a fluid to flow in a direction is described using a tensor.

This thesis report will only consider a mesoscopic model of the problem. The fundamental equations of the mathematical model describe the fluid flow as continuity of fluid phases and use Darcy's law to describe the speed of the fluid in porous media.

3.2. Single-Phase Flow

In this section, we give a basic review of the mathematical model of a single-phase flow through porous media, and the general model of the physical problem will be derived. However, obtaining a detailed solution of the general model requires a lot of computational time.To get a good approximation, it is sufficient to describe the physical problem with general trends in the reservoir flow pattern. Therefore, a few assumptions will be made to reduce the complexity of the model.

3.2.1. Mathematical Model

The mathematical model of a single-phase inflow and outflow through a porous medium is used to predict and analyse fluid flow, while considering conservation of mass:

$$\alpha \frac{\partial(\rho\varphi)}{\partial t} + \nabla(\alpha\rho\mathbf{v}) = \alpha\rho q, \tag{3.1}$$

where $\rho(t,\mathbf{x})$ is the fluid density, $\alpha(\mathbf{x})$ is a geometric factor, g is the gravitational constant and $q(t,\mathbf{x})$ is a source term. The geometric factor depends on the dimension of the problem. For the 1D problem, we have $\alpha(x) = A(x)$, where A is the cross-sectional area. For the 2D problem, the geometric factor $\alpha(\mathbf{x}) = h(x,y)$, where h is the reservoir height. We consider a two-dimensional model of the problem where $\alpha(\mathbf{x}) = h(x,y)$ is constant. The mesoscopic model considers Darcy's velocity $\mathbf{v}(t,\mathbf{x})$, that is defined as

$$\mathbf{v} = -\frac{\mathbf{K}}{\mu}(\nabla p - \rho g \nabla d), \tag{3.2}$$

where $p(t,\mathbf{x})$ is the pressure, μ is the fluid viscosity, $\mathbf{K}(\mathbf{x})$ is the rock permeability and $d(\mathbf{x})$ is the reservoir depth. Combining (3.1) and (3.2) gives

$$\frac{\partial(\rho\varphi)}{\partial t} - \nabla\left(\rho\frac{\mathbf{K}}{\mu}(\nabla p - \rho g \nabla d)\right) = \rho q. \tag{3.3}$$

The fluid viscosity and rock permeability hardly depend on the pressure in our case, so they will be taken independent of the pressure. Assuming the fluid density depends on the pressure, the liquid compressibility c_l can be defined as

$$c_l(p) := \frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}p}. \tag{3.4}$$

Similarly, the relation between rock porosity and pressure can be defined with the rock compressibility c_r

$$c_r(p) := \frac{1}{\varphi}\frac{\mathrm{d}\varphi}{\mathrm{d}p}. \tag{3.5}$$

Note that Equation (3.4) and (3.5) are first order ordinary differential equations. The total compressibility c_t be defined as

$$c_t = c_r + c_l. \tag{3.6}$$

Since fluid density and rock porosity depends on the pressure, the following relation is obtained by

$$\frac{\partial(\rho\varphi)}{\partial t} = \varphi\frac{\partial\rho}{\partial p}\frac{\partial p}{\partial t} + \rho\frac{\partial\varphi}{\partial p}\frac{\partial p}{\partial t} = \rho\varphi\frac{\partial p}{\partial t}\left(\frac{1}{\rho}\frac{\partial\rho}{\partial p} + \frac{1}{\varphi}\frac{\partial\varphi}{\partial p}\right). \tag{3.7}$$

Then, we substitute Equation (3.6) in Equation (3.3) using Equation (3.7) to get the general result given in Equation (3.8).

The general nonlinear partial differential equation for the dependent variable pressure p is given by

$$c_t\rho\varphi\frac{\partial p}{\partial t} - \nabla\left(\rho\frac{\mathbf{K}}{\mu}(\nabla p - \rho g \nabla d)\right) = \rho q. \tag{3.8}$$

The quantities and dimensions of the used variables are given in Appendix A.

3.2.2. Boundary Conditions

In reservoir simulation, one would describe a closed flow system and provide boundary conditions to obtain a unique solution. For a closed flow system, the pressure related boundary conditions are Dirichlet boundary conditions. The homogeneous boundary condition is defined as:

$$p = 0 \quad \text{for } \mathbf{x} \in \partial\Omega, \tag{3.9}$$

where $\partial\Omega$ denotes the boundary of the porous media Ω. Another type of boundary condition that is often prescribed for this flow problem are in- and outflow related conditions, that corresponds to Neumann boundary conditions:

$$\mathbf{v} \cdot \mathbf{n} = 0 \quad \text{for } \mathbf{x} \in \partial\Omega, \tag{3.10}$$

where $\mathbf{n}$ is defined as a normal vector orthogonal to the boundary $\partial\Omega$.

The boundary conditions should be chosen such that the solution is well-posed.

3.2.3. Incompressible Model

The basic model for simulating one-phase flow through porous media is assuming the density and the porosity are pressure independent, i.e. $\frac{\partial \rho}{\partial p} = \frac{\partial \varphi}{\partial p} = 0$. Therefore, the incompressible model is also time-independent and Equation (3.8) becomes:

$$-\nabla\left(\rho\frac{\mathbf{K}}{\mu}(\nabla p - \rho g \nabla d)\right) = \rho q. \tag{3.11}$$

Assuming isotropic permeability, constant depth, fluid with constant velocity and density, Equation (3.11) becomes

$$-\frac{1}{\mu}\nabla(\mathbf{K}\nabla p) = q. \tag{3.12}$$

Equation (3.12) is an example of an elliptic equation with constant coefficients $\mu, \mathbf{K}$. This will be solved numerically using numerical solver given in Chapter 4 and Chapter 6. The numerical experiments in this thesis only consider incompressible models. For the compressible model of the one-phase flow, it can be found in Appendix B.

Discretization of the Incompressible Model

The method of lines is used to solve Equation (3.12). A finite difference scheme with cell central differences is used to approximate spatial derivatives. Assume a uniform grid with grid size $\Delta x, \Delta y, \Delta z$ for the dimension x, y, z, respectively. Let (i, j, l) be the cell for the x-direction, y-direction and z-direction respectively. The pressure in the cell (i, j, l) is defined as $p(x_i, y_j, z_l) = p_{i,j,l}$.

Equation 3.12 can be rewritten as

$$-\frac{1}{\mu}\left[\frac{\partial}{\partial x}\left(k\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial p}{\partial y}\right) + \frac{\partial}{\partial z}\left(k\frac{\partial p}{\partial z}\right)\right] = q. \tag{3.13}$$

The first term in the equation in x-direction can be approximated as

$$\frac{\partial}{\partial x}\left(k\frac{\partial p}{\partial x}\right) \approx \frac{k_{i+\frac{1}{2},j,l}(p_{i+1,j,l} - p_{i,j,l}) - k_{i-\frac{1}{2},j,l}(p_{i,j,l} - p_{i-1,j,l})}{(\Delta x)^2} + \mathcal{O}\big((\Delta x)^2\big), \tag{3.14}$$

where $k_{i+\frac{1}{2},j,l}$ denotes the harmonic averaging of grid-block permeabilities $(i+1,j,l)$ and (i,j,l) given by

$$k_{i+\frac{1}{2},j,l} = \frac{2}{\frac{1}{k_{i+1,j,l}} + \frac{1}{k_{i,j,l}}}. \tag{3.15}$$

Let the transmissibility between cell $(i+1,j,l)$ and (i,j,l) be given by

$$T_{i+\frac{1}{2},j,l} := \frac{1}{\mu}\frac{2\Delta y \Delta z}{\Delta x} k_{i+\frac{1}{2},j,l}. \tag{3.16}$$

Similar expressions can be obtained for the y,z-direction.

For a cell (i,j,l) the discretization of Equation (3.13) is given by

$$\begin{aligned} &-p_{i-1,j,l}T_{i-\frac{1}{2},j,l} - p_{i,j-1,l}T_{i,j-\frac{1}{2},l} - p_{i,j,l-1}T_{i,j,l-\frac{1}{2}} \\ &+p_{i-1,j,l}\left(T_{i-\frac{1}{2},j,l} + T_{i,j-\frac{1}{2},l} + T_{i,j,l-\frac{1}{2}} + T_{i+\frac{1}{2},j,l} + T_{i,j+\frac{1}{2},l} + T_{i,j,l+\frac{1}{2}}\right) \\ &-p_{i+1,j,l}T_{i+\frac{1}{2},j,l} - p_{i,j+1,l}T_{i,j+\frac{1}{2},l} - p_{i,j,l+1}T_{i,j,l+\frac{1}{2}} = \Delta x \Delta y \Delta z \; q_{i,j,l}. \end{aligned} \tag{3.17}$$

The transmissibility matrix $\mathbf{T}$ can be constructed with the latter equation and the given boundary conditions. Finally, Equation (3.12) can be written as

$$\mathbf{Tp} = \mathbf{q}, \tag{3.18}$$

which is a system of linear equations.

3.3. Two-Phase flow

$$\alpha\frac{\partial(\rho_w \varphi S_w)}{\partial t} + \nabla(\alpha\rho_w \mathbf{v}_w) = \alpha\rho_w q_w, \tag{3.19}$$

$$\alpha\frac{\partial(\rho_o \varphi S_o)}{\partial t} + \nabla(\alpha\rho_o \mathbf{v}_o) = \alpha\rho_o q_o, \tag{3.20}$$

where the subscript w,o denotes water and oil respectively. The difference of the models between one-phase flow and two-phase flow is the addition of saturation S_w, S_o, i.e. the fraction of the pore space occupied by the respective phase. Since we only consider 2D problems, we choose as geometric factor $\alpha(\mathbf{x}) = h(x,y)$, where h is the reservoir height. The Darcy's law for this model is defined as

$$\mathbf{v}_w = -\frac{k_{rw}}{\mu_w}(\nabla p_w - \rho_w g \nabla d), \tag{3.21}$$

$$\mathbf{v}_o = -\frac{k_{ro}}{\mu_o}(\nabla p_o - \rho_o g \nabla d), \tag{3.22}$$

where k_{rw}, k_{ro} are the relative permeabilities. This represents the additional resistance to flow of a phase caused by a different phase [8]. More details and information can be found in [3, 8].

3.4. Well Model

Usually, in reservoir simulation, the closed flow system is described in combination with a well model as source term. Fluids are injected or produced in a well at constant

bottom-hole pressure or at a constant rate. The inflow performance is defined by the bottom-hole pressure with surface flow rate. The simplest model is the Peaceman linear model [3, 8, 13, 22], which is defined by

$$q_{i,j,l} = J(p_{i,j,l} - \mathbf{p}^{bh}_{i,j,l}), \tag{3.23}$$

where $\mathbf{p}^{bh}_{i,j,l}$ is the bottom-hole pressure in cell (i,j,l) and J is the well production or injection productivity index.

4

Iterative Numerical Methods

The partial differential equation of a one-phase constant compressible flow has been discretized in the following form:

$$\mathbf{V}\frac{\mathrm{d}\rho(\mathbf{p})}{\mathrm{d}t} + \mathbf{T}\mathbf{p} = \mathbf{q}(\mathbf{p}), \tag{4.1}$$

where $\mathbf{V} \in \mathbb{R}^{n\times n}$ is the accumulation matrix which is strictly positive, $\mathbf{T}$ is the transmissibility matrix which is SPD, $\mathbf{p}$ is the pressure which is unknown and $\mathbf{q}$ is the source vector. The unknown time variable $\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t}$ is approximated by using the Euler Backwards method. Let the time step size be defined by $\Delta t^k = t^{k+1} - t^k$. Equation (4.1) is discretized in time by

$$\mathbf{V}\frac{\rho(\mathbf{p}^{k+1}) - \rho(\mathbf{p}^k)}{\Delta t^k} + \mathbf{T}\mathbf{p}^{k+1} = \mathbf{q}(\mathbf{p}^{k+1}). \tag{4.2}$$

The equation is nonlinear and is solved to find the unknown pressure $\mathbf{p}$ by using linearization methods, i.e. Newton-Raphson. For every timestep, it can be written as a system of linear equations in the form of

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4.3}$$

where $\mathbf{A}$ is a large SPD matrix, which makes it suitable to use iterative methods. This section will start with explaining Newton-Raphson and defining $\mathbf{A}$ more precisely. Thereafter, iteration methods like the Conjugate Gradient, preconditioner techniques and deflation methods will be explained.

4.1. Newton-Raphson

The Newton-Raphson method is used to linearize nonlinear equations. Firstly, for a one-dimensional case, function $h(x)$ would be defined such that $h(x) = 0$. The iteration steps are found by using a Taylor expansion. Start with an initial guess x^0 and for each iteration step, compute

$$x^{k+1} = x^k - \frac{h(x^k)}{h'(x^k)}, \tag{4.4}$$

while assuming $h'(x^k) \neq 0$ for every step k. Depending on the initial guess, this method will converge. For the multidimensional case, the same process can be used. Let $\mathbf{f}(\mathbf{x})$

be an n-dimensional function. Assume $\mathbf{x}^* = \mathbf{x}^k + \delta\mathbf{x}$ where $\mathbf{f}(\mathbf{x}^*) = 0$, then the Taylor expansion around point $\mathbf{x}^k$ is

$$\mathbf{f}(\mathbf{x}^k + \delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^k) + \mathbf{J}_\mathbf{f}(\mathbf{x}^k)\delta\mathbf{x}, \tag{4.5}$$

where $\mathbf{J}_\mathbf{f}$ is the Jacobian of $\mathbf{f}$. Recall, $\mathbf{f}(\mathbf{x}^*) = 0$, thus to find $\delta\mathbf{x}$ one needs to solve the linear system

$$\mathbf{J}_\mathbf{f}(\mathbf{x}^k)\delta\mathbf{x} = -\mathbf{f}(\mathbf{x}^k). \tag{4.6}$$

Thereafter, update $\mathbf{x}^{k+1} = \mathbf{x}^k + \delta\mathbf{x}$. The algorithm for every iteration is defined as

Algorithm 1 Newton-Raphson

1: Initial: $\mathbf{p}^0, \varepsilon$
2: **while** $\left|\mathbf{p}^{k+1} - \mathbf{p}^k\right| > \varepsilon$ **do**
3: Solve: $\mathbf{J}_\mathbf{f}(\mathbf{p}^k)\delta\mathbf{p} = -\mathbf{f}(\mathbf{p}^k)$
4: Update: $\mathbf{p}^{k+1} = \mathbf{p}^k + \delta\mathbf{p}$
5: $k = k + 1$

Example

The heat equation with nonlinear source term is defined as

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + T(T-1) \text{ for } 0 < x < 1, \;\; t > 0, \tag{4.7}$$

with homogeneous Dirichlet boundary conditions $T(0) = T(1) = 0$. To illustrate how Newton-Raphson works, only the steady state of the problem will be solved, i.e. $\frac{\partial T}{\partial t} = 0$. The analytic solution for this problem is

$$T(x) = 0 \quad \text{for } 0 < x < 1. \tag{4.8}$$

To solve this problem numerically, we use a uniform gridsize $n = 4$ with $\Delta x = 0.25$. Hence, the function $\mathbf{f}(\mathbf{T})$ can be defined as

$$\begin{aligned}\mathbf{f}(\mathbf{T}) \;=\; & \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} \\ & + \begin{bmatrix} T_1(T_1-1) & & & & \\ & T_2(T_2-1) & & & \\ & & T_3(T_3-1) & & \\ & & & T_4(T_4-1) & \\ & & & & T_5(T_5-1) \end{bmatrix}.\end{aligned} \tag{4.9}$$

Newton-Raphson will be used to solve $\mathbf{f}(\mathbf{T})$. The Jacobian matrix is defined as

$$\mathbf{J}_\mathbf{f}(\mathbf{T}) = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} + 2\begin{bmatrix} T_1 & & & & \\ & T_2 & & & \\ & & T_3 & & \\ & & & T_4 & \\ & & & & T_5 \end{bmatrix} - \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}. \tag{4.10}$$

Choose as initial condition

$$\mathbf{T}_{\text{int}} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}^{\mathsf{T}} \tag{4.11}$$

with stop criteria 10^{-4}. After 6 iterations the steady state solution is

$$\mathbf{T}_{\text{ss}} \approx \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}}, \tag{4.12}$$

with error $2.4559 \cdot 10^{-31}$. The solution found with the numerical scheme is close to the analytic solution with a small error.

For the original problem, Equation (4.2) is nonlinear and multidimensional. Define the function

$$\mathbf{f}(\mathbf{p}^{k+1}; \mathbf{p}^k) = \mathbf{V}\frac{\rho(\mathbf{p}^{k+1}) - \rho(\mathbf{p}^k)}{\Delta t^k} + \mathbf{T}\mathbf{p}^{k+1} - \bar{\mathbf{q}}(\mathbf{p}^{k+1}). \tag{4.13}$$

This will be used to find the solution for the pressure $\mathbf{p}$.

4.2. Basic Iterative Method

It is time-consuming to solve the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with direct solvers for $\mathbf{A} \in \mathbb{R}^{n \times n}$. Therefore, another way to solve the system is by using iterative methods. The basic iterative method goes as follows: Split $\mathbf{A} = \mathbf{M} - \mathbf{N}$ such that $\mathbf{M}^{-1}$ exists. The iterative condition for $\mathbf{x}$ can be derived from

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Rightarrow \mathbf{M}\mathbf{x} = \mathbf{b} + \mathbf{N}\mathbf{x}. \tag{4.14}$$

Thus, the result is

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{M}^{-1}\mathbf{r}^k, \tag{4.15}$$

where $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$ is the residual. The residual denotes the difference between the iterative solution and true solution. There are different possible choices for $\mathbf{M}$. The Jacobi method uses $\mathbf{M} = diag(\mathbf{A})$ and Gauss-Seidel uses $\mathbf{M} = \mathbf{L}$, where $\mathbf{L}$ is the lower triangle of $\mathbf{A}$. The iterative method goes as follows: Choose initial guess $\mathbf{x}^0$ and after k iterations the iterative solution can be written as.

$$\begin{aligned} \mathbf{x}^0 &= \mathbf{x}^0 \\ \mathbf{x}^1 &= \mathbf{x}^1 + \mathbf{M}^{-1}\mathbf{r}^1 = \mathbf{x}^0 + \mathbf{M}^{-1}\mathbf{r}^0 \\ \mathbf{x}^2 &= \mathbf{x}^2 + \mathbf{M}^{-1}\mathbf{r}^2 \\ &= \ldots = \mathbf{x}^0 + \mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-1}\mathbf{r}^0 + 2\mathbf{M}^{-1}\mathbf{r}^0 \\ &\quad \text{etc.} \end{aligned}$$

It follows that the iterative solution can be written as

$$\mathbf{x}^k = \mathbf{x}^0 + \text{span}\{\mathbf{M}^{-1}\mathbf{r}^0, \mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-1}\mathbf{r}^0, \ldots, (\mathbf{M}^{-1}\mathbf{A})^{k-1}\mathbf{M}^{-1}\mathbf{r}^0\}. \tag{4.16}$$

The Krylov subspace of dimension k is defined as

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}) := \text{span}\{\mathbf{A}\mathbf{r}, \mathbf{A}^2\mathbf{r}, \ldots, \mathbf{A}^{k-1}\mathbf{r}\}. \tag{4.17}$$

Hence, the iterative solution can be written as

$$\mathbf{x}^k = \mathbf{x}^0 + \mathcal{K}_k(\mathbf{M}^{-1}\mathbf{A}, \mathbf{M}^{-1}\mathbf{r}^0). \tag{4.18}$$

The matrix $\mathbf{M}$ is also called a *preconditioner*, which will be explained in Section 4.4. In the following section, the Conjugate Gradient method will be explained by using $\mathbf{M} = \mathbf{I}$.

4.3. Conjugate Gradient

Conjugate Gradient (CG) is an iterative method that is used for SPD matrices. The purpose of CG is to construct a sequence $\{\mathbf{x}^k\}$ such that it minimizes the $\mathbf{A}$-norm of the error:

$$\min_{\mathbf{x}^k \in \mathcal{K}_k(\mathbf{A},\mathbf{r}^0)} \|\mathbf{x} - \mathbf{x}^k\|_{\mathbf{A}}, \tag{4.19}$$

where $\mathbf{x}$ is the true solution. CG uses search vectors $\{\mathbf{p}^i\}$ that are defined such that $\langle \mathbf{A}\mathbf{p}^i, \mathbf{p}^j \rangle = 0$ for every $i \neq j$. Also, the residuals should be orthogonal, hence $\langle \mathbf{r}^i, \mathbf{r}^j \rangle = 0$ for every $i \neq j$. With every iteration step, there will be updates for the solution and residual, defined as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k \quad \text{and} \quad \mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k, \tag{4.20}$$

respectively, where α^k is chosen such that it minimizes Equation (4.19). Therefore $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}{\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \rangle}$. The search vectors are updated as

$$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta^k \mathbf{p}^k. \tag{4.21}$$

The method is summarized in Algorithm 2 and can be found in [23, 24].

Algorithm 2 Conjugate Gradient

1: Initial: $\mathbf{x}^0, \varepsilon$
2: Compute: $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$ and $\mathbf{p}^0 = \mathbf{r}^0$
3: **for** $k = 0, \ldots$ **do**
4: **while** $\mathbf{r}^k > \varepsilon$ **do**
5: $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}{\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \rangle}$
6: $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
7: $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
8: $\beta^k = \frac{\langle \mathbf{r}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}$
9: $\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta^k \mathbf{p}^k$

Convergence

It can be proven that the CG method converges and the prove can be found in [23, 25, 31]. It depends mainly on the condition number. After k iterations, the error in the $\mathbf{A}$-norm is bounded by

$$\|\mathbf{x} - \mathbf{x}^k\|_{\mathbf{A}} \leq 2\|\mathbf{x} - \mathbf{x}^0\|_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} \right)^k. \tag{4.22}$$

4.4. Preconditioner

The convergence depends on the condition number of the matrix. Preconditioners can be used to achieve a faster convergence by reducing the condition number. The preconditioner matrix $\mathbf{M}$ is applied to the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ as

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}. \tag{4.23}$$

In the system given in Equation (4.23), $\mathbf{M}^{-1}\mathbf{A}$ is not necessary SPD, thus the system is redefined as

$$\overline{\mathbf{A}}\overline{\mathbf{x}} = \overline{\mathbf{b}}, \tag{4.24}$$

where $\overline{\mathbf{A}} = \mathbf{M}^{-\frac{1}{2}}\mathbf{A}\mathbf{M}^{-\frac{1}{2}}$, $\overline{\mathbf{x}} = \mathbf{M}^{\frac{1}{2}}\mathbf{x}$ and $\overline{\mathbf{b}} = \mathbf{M}^{-\frac{1}{2}}\mathbf{b}$. We need the extra conditions that $\mathbf{M}$ should be SPD and $\mathbf{M}^{-\frac{1}{2}}$ should exists and be symmetric to ensure convergence. It follows that $\overline{\mathbf{A}}$ is SPD, the proof can be found in [25]. There are many matrices that can be used as preconditioner. If $\mathbf{M} = \mathbf{I}$, then this is the iterative method from before and the condition number remain unchanged. If $\mathbf{M} = \mathbf{A}$, the condition number is equal to 1 and the solution can be found in one step. It is often hard to compute $\mathbf{A}^{-1}$, so it is often not chosen as preconditioner.

This report uses Incomplete Cholesky as preconditioner, that will be denoted by $\mathbf{M}_{IC0}$. This preconditioner is an SPD approximation of the Cholesky factorization where the number of fill-in is chosen. This entails a decomposition of the form $\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathsf{T}} - \mathbf{A}_r$, where $\mathbf{L}$ is the lower triangle with the same zero pattern as matrix $\mathbf{A}$ and $\mathbf{A}_r$ is the residual or error of the factorization. The matrix $\mathbf{A}$ is approximated with $\mathbf{L}\mathbf{L}^{\mathsf{T}}$. More information can be found in [23, 31].

4.4.1. Preconditioned Conjugate Gradient

The new system using a preconditioner is defined as

$$\overline{\mathbf{A}}\overline{\mathbf{x}} = \overline{\mathbf{b}}, \tag{4.25}$$

where $\overline{\mathbf{A}}$ is SPD, thus Conjugate Gradient algorithm can be used. The derivation of this method, also called Preconditioned Conjugate Gradient (PCG), can also be found in [25] and is given in Algorithm 3.

Algorithm 3 Preconditioned Conjugate Gradient

Initial: $\mathbf{x}^0, \varepsilon$
Compute: $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0, \mathbf{z}^0 = \mathbf{M}^{-1}\mathbf{r}^0$ and $\mathbf{p}^0 = \mathbf{z}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\mathbf{z}^{k+1} = \mathbf{M}^{-1}\mathbf{r}^k$
        $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{z}^k \rangle}{\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \rangle}$
        $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
        $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
        $\beta^k = \frac{\langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{z}^k, \mathbf{r}^k \rangle}$
        $\mathbf{p}^{k+1} = \mathbf{z}^{k+1} + \beta^k \mathbf{p}^k$

To use preconditioned Conjugate Gradient method in practise it is needed that $\mathbf{M}^{-1}$ is inexpensive to apply and cheap to compute.

Convergence

The error is bounded by the next inequality:

$$\|\mathbf{x} - \mathbf{x}^k\|_{\mathbf{A}} \leq 2\|\mathbf{x} - \mathbf{x}^0\|_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{M}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{M}^{-1}\mathbf{A})} + 1} \right)^k. \tag{4.26}$$

The advantages of choosing a good preconditioner is that the condition number is being lowered and a faster convergence is achieved.

4.5. Deflation Method

The eigenvalues of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$ are not always favourable and using PCG does not nearly lead to iteration numbers as low as required. Hence, deflation methods will be considered to accelerate the solving process. See [2, 9, 10, 32, 33] The deflation method reduces the condition number by setting the extreme eigenvalues equal to zero such that the convergence bound is small. The method is defined by using the next definition:

Definition 4.5.1. *Given an $\mathbf{A} \in \mathbb{R}^{n\times n}$ which is SPD and given a deflation-subspace matrix $\mathbf{Z}$ of size $n \times m$ where $m \ll n$, the deflation method is defined as*

$$\mathbf{P} = \mathbf{I} - \mathbf{A}\mathbf{Q} \qquad \mathbf{P} \in \mathbb{R}^{n\times n}, \quad \mathbf{Q} \in \mathbb{R}^{n\times n}, \tag{4.27}$$

where $\mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^{\mathsf{T}}$ with $\mathbf{Z} \in \mathbb{R}^{n\times m}$, $\mathbf{E} = \mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}$ and $\mathbf{E} \in \mathbb{R}^{m\times m}$. The columns of the deflation-subspace matrix $\mathbf{Z}$ are called deflation vectors. The deflation vectors are chosen such that the matrix $\mathbf{E}$, also known as coarse matrix, is nonsingular. This matrix is used to construct the correction matrix $\mathbf{Q}$. The choices for deflation subspace matrix $\mathbf{Z}$ will be discussed in Chapter 5. Then we state the following properties:

Lemma 4.5.1. *Let $\mathbf{A}, \mathbf{P}, \mathbf{Q}$ and $\mathbf{Z}$ be defined as in Definition 4.5.1. Then,*

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

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

c. $\mathbf{Q}\mathbf{A} = \mathbf{I} - \mathbf{P}^{\mathsf{T}}$

d. $\mathbf{P}^{\mathsf{T}}\mathbf{Z} = \mathbf{P}^{\mathsf{T}}\mathbf{Q} = \mathbf{0}$

e. $\mathbf{P}\mathbf{A}\mathbf{Z} = \mathbf{P}\mathbf{A}\mathbf{Q} = \mathbf{0}$

f. $\mathbf{Q}^{\mathsf{T}} = \mathbf{Q}$

g. $\mathbf{Q}\mathbf{A}\mathbf{Q} = \mathbf{Q}$

h. $\mathbf{Q}\mathbf{A}\mathbf{P}^{\mathsf{T}} = 0$

i. $\mathbf{Q}\mathbf{P} = 0$

j. $\mathbf{Q}\mathbf{A}\mathbf{Z} = \mathbf{Z}$

The proof can be found in [26, Sect. 3].

Assume we have the system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4.28}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is sparse and SPD, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. We decompose $\mathbf{x}$ as:

$$\mathbf{x} = (\mathbf{I} - \mathbf{P}^{\mathsf{T}})\mathbf{x} + \mathbf{P}^{\mathsf{T}}\mathbf{x} \tag{4.29}$$

to obtain

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.30}$$

$$\Rightarrow \quad \mathbf{A}(\mathbf{I} - \mathbf{P}^{\mathsf{T}})\mathbf{x} + \mathbf{A}\mathbf{P}^{\mathsf{T}}\mathbf{x} = \mathbf{b} \tag{4.31}$$

$$\Rightarrow \quad \mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{A}\mathbf{P}^{\mathsf{T}}\mathbf{x} = \mathbf{b} \tag{4.32}$$

$$\Rightarrow \quad \mathbf{A}\mathbf{P}^{\mathsf{T}}\mathbf{x} = (\mathbf{I} - \mathbf{A}\mathbf{Q})\mathbf{b} \tag{4.33}$$

$$\Rightarrow \quad \mathbf{P}\mathbf{A}\tilde{\mathbf{x}} = \mathbf{P}\mathbf{b}. \tag{4.34}$$

It follows from Lemma 4.5.1d. that $\mathbf{PA}$ is a singular matrix since it contains zero eigenvalues. Hence, after using the deflation method, the system can be written as

$$\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b}, \tag{4.35}$$

where $\hat{\mathbf{x}}$ is the non-unique solution of the deflated system. The solution of the original system $\mathbf{Ax} = \mathbf{b}$ is found by using [25]

$$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^{\mathsf{T}}\hat{\mathbf{x}}. \tag{4.36}$$

4.6. Deflated Preconditioned Conjugated Gradient

This was proposed by [25, 30, 34] and the method uses the basis matrix as deflation-subspace matrix to reduce the amount of iterations required to solve the system. The system $\mathbf{Ax} = \mathbf{b}$ is solved by defining the following system:

$$\tilde{\mathbf{P}}\tilde{\mathbf{A}}\tilde{\hat{\mathbf{x}}} = \tilde{\mathbf{P}}\tilde{\mathbf{b}}, \tag{4.37}$$

with

$$\tilde{\mathbf{A}} := \mathbf{M}^{-\frac{1}{2}}\mathbf{A}\mathbf{M}^{-\frac{1}{2}}, \quad \tilde{\hat{x}} := \mathbf{M}^{\frac{1}{2}}\hat{\mathbf{x}}, \quad \tilde{\mathbf{b}} := \mathbf{M}^{-\frac{1}{2}}\mathbf{b} \tag{4.38}$$

and

$$\tilde{\mathbf{P}} := \mathbf{I} - \tilde{\mathbf{A}}\tilde{\mathbf{Q}}, \quad \tilde{\mathbf{Q}} := \tilde{\mathbf{Z}}\tilde{\mathbf{E}}^{-1}\tilde{\mathbf{Z}}^{\mathsf{T}}, \quad \tilde{\mathbf{E}} := \tilde{\mathbf{Z}}^{\mathsf{T}}\tilde{\mathbf{A}}\tilde{\mathbf{Z}}, \tag{4.39}$$

where $\tilde{\hat{\mathbf{x}}}$ is the non-unique deflation solution. The true solution of the deflation method can be found with:

$$\tilde{\mathbf{x}} := \tilde{\mathbf{Q}}\tilde{\mathbf{b}} + \tilde{\mathbf{P}}^{\mathsf{T}}\tilde{\hat{\mathbf{x}}}. \tag{4.40}$$

Therefore, the true solution of the system $\mathbf{Ax} = \mathbf{b}$ is

$$\mathbf{x} = \mathbf{M}^{-\frac{1}{2}}\tilde{\mathbf{x}}. \tag{4.41}$$

This is summarized and given in Algorithm 4.

Algorithm 4 Deflated Preconditioned Conjugated Gradient

1: Initial: $\hat{\mathbf{x}}^0, \varepsilon$
2: Compute: $\tilde{\mathbf{r}}^0 = \tilde{\mathbf{b}} - \tilde{\mathbf{A}}\tilde{\mathbf{x}}^0, \hat{\tilde{\mathbf{r}}}^0 = \tilde{\mathbf{P}}\tilde{\mathbf{r}}^0$ and $\tilde{\mathbf{p}}^0 = \hat{\tilde{\mathbf{r}}}^0$
3: **for** $k = 0, \ldots$ **do**
4: **while** $\tilde{\mathbf{r}}^k > \varepsilon$ **do**
5: $\hat{\tilde{\mathbf{z}}}^k = \tilde{\mathbf{P}}\tilde{\mathbf{A}}\tilde{\mathbf{p}}^k$
6: $\alpha^k = \dfrac{\left\langle \tilde{\mathbf{r}}^k, \tilde{\mathbf{r}}^k \right\rangle}{\left\langle \tilde{\mathbf{p}}^k, \hat{\tilde{\mathbf{z}}}^k \right\rangle}$
7: $\hat{\tilde{\mathbf{x}}}^{k+1} = \hat{\tilde{\mathbf{x}}}^k + \alpha^k \tilde{\mathbf{p}}^k$
8: $\beta^k = \dfrac{\left\langle \hat{\tilde{\mathbf{x}}}^{k+1}, \hat{\tilde{\mathbf{x}}}^{k+1} \right\rangle}{\left\langle \hat{\tilde{\mathbf{x}}}^k, \hat{\tilde{\mathbf{x}}}^k \right\rangle}$
9: $\hat{\tilde{\mathbf{r}}}^{k+1} = \hat{\tilde{\mathbf{r}}}^k - \alpha^k \hat{\tilde{\mathbf{z}}}^k$
10: $\tilde{\mathbf{p}}^{k+1} = \hat{\tilde{\mathbf{r}}}^{k+1} + \beta^k \tilde{\mathbf{p}}^k$
11: $\tilde{\mathbf{x}}^{k+1} := \tilde{\mathbf{Q}}\tilde{\mathbf{b}} + \tilde{\mathbf{P}}^{\mathsf{T}}\tilde{\hat{\mathbf{x}}}^{k+1}$
12: $\mathbf{x}^{k+1} = \mathbf{M}^{-\frac{1}{2}}\tilde{\mathbf{x}}^{k+1}$

Algorithm 4 is not being used since it is not practical. A more practical algorithm is given by [25] and is found in Algorithm 5.

Algorithm 5 Deflated Preconditioned Conjugated Gradient (Practical version)

1: Initial: $\mathbf{x}^0, \varepsilon$
2: Compute: $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0, \hat{\mathbf{r}}^0 = \mathbf{P}\mathbf{r}^0, \boldsymbol{z}^0 = \mathbf{M}^{-1}\hat{\mathbf{r}}^0$ and $\mathbf{p}^0 = \boldsymbol{z}^0$
3: **for** $k = 0, \ldots$ **do**
4: **while** $\mathbf{r}^k > \varepsilon$ **do**
5: $\alpha^k = \frac{\langle \hat{\mathbf{r}}^k, \boldsymbol{z}^k \rangle}{\mathbf{p}^k, \mathbf{PAp}^k}$
6: $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \alpha^k \mathbf{p}^k$
7: $\hat{\mathbf{r}}^{k+1} = \hat{\mathbf{r}}^k - \alpha^k \mathbf{PAp}^k$
8: $\hat{\boldsymbol{z}}^{k+1} = \mathbf{M}^{-1}\hat{\mathbf{r}}^k$
9: $\beta^k = \frac{\langle \hat{\mathbf{r}}^{k+1}, \boldsymbol{z}^{k+1} \rangle}{\langle \hat{\mathbf{r}}^k, \boldsymbol{z}^k \rangle}$
10: $\mathbf{p}^{k+1} = \boldsymbol{z}^0 + \beta^k \mathbf{p}^k$
11: $\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^{\mathsf{T}}\mathbf{x}^{k+1}$

Accuracy/Convergence

By using DCG, from Lemma 4.5.1 follows that the smallest eigenvalue will be equal to zero. Thus, another condition number will be defined for the convergence.

Definition 4.6.1. *Assume* $\mathbf{PA}$ *is SPSD with eigenvalues* $\lambda_1, \ldots, \lambda_n$*. The effective condition number is defined as*

$$\kappa_{eff}(\mathbf{PA}) = \frac{\lambda_{\max}(\mathbf{PA})}{\lambda_{\min}(\mathbf{PA})}, \tag{4.42}$$

where $\lambda_{\min}$ *is the smallest nonzero eigenvalue.*

The error in the $\mathbf{A}$-norm is given by

$$\|\mathbf{x} - \mathbf{x}^k\|_{\mathbf{A}} \leq 2\|\mathbf{x} - \mathbf{x}^0\|_{\mathbf{A}} \left(\frac{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})} - 1}{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})} + 1} \right)^k. \tag{4.43}$$

4.7. Comparison of Deflated Methods with the Original Matrix

The convergence of the methods depends on the condition number. In this section, we show that the conditioned number of the deflated system is smaller or equal than the original system, i.e. the spectrum of $\mathbf{PA}$ is smaller or equal than the original matrix $\mathbf{A}$. See the following Theorem:

Theorem 4.7.1. *Let* $\mathbf{A}$ *and* $\mathbf{P}$ *be given as in Definition 4.5.1, then the next inequality holds:*

$$\kappa_{eff}(\mathbf{PA}) \leq \kappa(\mathbf{A}), \tag{4.44}$$

for all deflation vectors $\mathbf{Z}$*.*

The proof can be found in [25, Sect. 3]. Theorem 4.7.1 can be generalized by using an arbitrary SPD preconditioner $\mathbf{M}$. This results in Theorem 4.7.2:

Theorem 4.7.2. *Let* $\mathbf{A}$ *and* $\mathbf{P}$ *be given as in Definition 4.5.1. Let* $\mathbf{M}$ *be an arbitrary SPD preconditioner matrix, then the next inequality holds:*

$$\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA}) \leq \kappa\left(\mathbf{M}^{-1}\mathbf{A}\right), \tag{4.45}$$

for all deflation vectors $\mathbf{Z}$*.*

The proof and details can be found in [25, 26]. It follows from Theorem 4.7.2 that the condition number of the deflation preconditioned system $\mathbf{M}^{-1}\mathbf{PA}$ is smaller or equal to the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$. Therefore, the convergence speed of using DPCG is faster or equal to PCG.

4.8. Overview of Methods I

The mathematical model for water simulation is an nonlinear equation. The equation is solved using an iterative method Newton-Raphson. After linearization, the equation can be written in the form:

$$\mathbf{Ax} = \mathbf{b}. \tag{4.46}$$

Since $\mathbf{A}$ is SPD, we use iterative solvers to solve the system. The discussed methods are the CG method using a preconditioner with deflation techniques. Deflation-subspace matrix $\mathbf{Z}$ is needed for the deflation method. This will be discussed in Chapter 5.

5

Deflation Subspace-Vectors $\mathbf{Z}$

The reason to apply deflation techniques is to accelerate the solving process and reducing the number of iterations. Note that the deflation matrix $\mathbf{P}$ is defined as:

$$\mathbf{P} = \mathbf{I} - \mathbf{AQ} \qquad \mathbf{P} \in \mathbb{R}^{n \times n}, \quad \mathbf{Q} \in \mathbb{R}^{n \times n}, \tag{5.1}$$

where $\mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^{\mathsf{T}}$ with $\mathbf{Z} \in \mathbb{R}^{n \times m}$, $\mathbf{E} = \mathbf{Z}^{\mathsf{T}}\mathbf{AZ}$ and $\mathbf{E} \in \mathbb{R}^{m \times m}$. To construct the deflation matrix $\mathbf{P}$, the deflation-subspace matrix $\mathbf{Z}$ is needed. As proposed in [25], the optimal deflation vectors should satisfy the following requirements:

- The deflation-subspace vectors of $\mathbf{Z}$ are sparse
- The deflation-subspace vectors approximate the eigenspace corresponding to the unfavourable eigenvalues
- The costs of constructing the deflation vectors is cheap
- The approach is easily implemented with the Two-Level PCG method.

As proposed in [25], there are several choices to choose as deflation vectors. The most common choices for deflation vectors for the deflation-subspace matrix are

1. Deflation Subdomain vectors
2. Eigenvectors corresponding to the smallest eigenvalues of matrix $\mathbf{M}^{-1}\mathbf{A}$
3. POD-based basis vectors

This section will review the different deflation vectors.

5.1. Subdomain Vectors

The first possibility is to use subdomain vectors. The underlying idea is dividing the domain Ω in several domains Ω_i. Each subdomain Ω_i corresponds to one subdomain vector. The discretization of the subdomain is denoted with Ω_{h_i}. Each subdomain consist of ones for grid points in the interior of the discretization of subdomain Ω_{h_i} and zero for other domains Ω_{h_j}, where $j \neq i$. The subdomain vectors are relative cheap to construct and orthogonal. This will be illustrated with two examples. More details can be found in [1, 20, 25, 27, 34].

Example: Test case 1

Suppose we have the Laplace Equation. The mesh is divided into three subdomains Ω_i that is described in Figure 5.1.

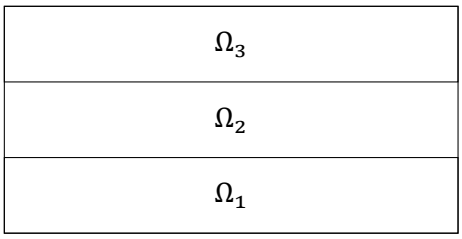


Figure 5.1: Domain divided into 3 subdomains.

Note that the domain consist of one type of layer. The number of subdomain vectors does not affect the solving process, that will be showed in Section 9.1.1. Suppose the matrix $\mathbf{A} \in \mathbb{R}^{9\times9}$ and we use lexicography numbering. The discretization of each subdomain $\Omega_{h_i},\ i = 1,2,3$, contain 3 grid points. Then, the mathematical representation of the deflation-subspace matrix $\mathbf{Z} \in \mathbb{R}^{9\times3}$ is given as:

$$\mathbf{Z} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^{\mathsf{T}} \tag{5.2}$$

Example: Test case 2

The domain defined in Test case 2 contains two different layers that has a huge contrast in permeability coefficient. Therefore, the number of layers is chosen such that one layer corresponds with a layer with the same rock permeability.

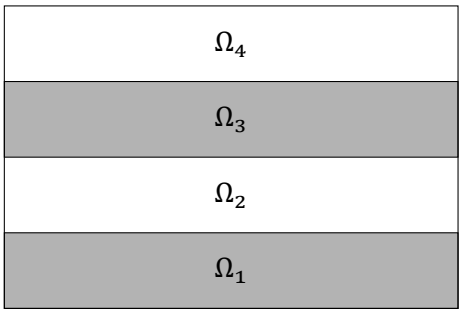


Figure 5.2: Domain in 4 subdomains.

Suppose the matrix $\mathbf{A} \in \mathbb{R}^{16\times16}$. and we use lexicography numbering. The discretization of each subdomain $\Omega_{h_i},\ i = 1,2,3,4$, contain 4 grid points. Then, the mathematical representation of the layers for deflation-subspace matrix $\mathbf{Z} \in \mathbb{R}^{16\times4}$ is:

$$\mathbf{Z} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}^{\mathsf{T}} \tag{5.3}$$

5.2. Eigenvectors

Choosing the eigenvectors corresponding to the unfavourable eigenvalues of the pre-conditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ ensures that the smallest eigenvalues will be transformed to zero for the deflated system, as in [25, 30]. The disadvantages of choosing eigenvectors is the cost of obtaining the eigenvectors. If the matrix $\mathbf{A}$ is large, it would be hard to obtain the exact eigenvectors. Therefore, using approximated eigenvectors can be a good substitution. For this report, we use exact eigenvectors for eigenvalue analysis. More details can be found in Section 7.3.

5.3. Proper Orthogonal Decomposition

Another choice is to use the Proper Orthogonal Decomposition (POD) method [4, 12, 18, 21, 28–30] to construct deflation vectors. Hence, an overview will be given in this section. The POD method is a Model Order Reduction-based method (MOR), which reduces a large system to a smaller system such that it is easier to solve. The solution of the system can be approximated by

$$\mathbf{x} \approx \sum_{i=1}^{m} c_i \psi_i, \tag{5.4}$$

where $c_i \in \mathbb{R}$ and $\{\psi_i\}$ are basis vectors of the basis matrix ψ, which will be specified later on.

The POD basis will be constructed as follows: First, we define l different right-hand sides $\mathbf{b}_i$. Then, we solve the system $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$ to obtain $\mathbf{x}_i$. The set $\{\mathbf{x}_i\}$ is called snapshots. Next, the correlation matrix is defined as:

$$\mathbf{R} = \frac{1}{l}\mathbf{X}\mathbf{X}^{\mathsf{T}}, \tag{5.5}$$

where $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \dots & \mathbf{x}_l\end{bmatrix}$. Note that the correlation matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$ is SPSD.

Proof. Let $\mathbf{y} \in \mathbb{R}^n$, then

$$\mathbf{y}^{\mathsf{T}}\mathbf{R}\mathbf{y} = \mathbf{y}^{\mathsf{T}}\frac{1}{l}\mathbf{X}\mathbf{X}^{\mathsf{T}}\mathbf{y} = \frac{1}{l}(\mathbf{X}^{\mathsf{T}}\mathbf{y})(\mathbf{X}^{\mathsf{T}}\mathbf{y}) = \frac{1}{l}(\mathbf{X}^{\mathsf{T}}\mathbf{y})^2 \geq 0, \tag{5.6}$$

where $l > 0$. Also,

$$\mathbf{R}^{\mathsf{T}} = \left(\frac{1}{m}\mathbf{X}\mathbf{X}^{\mathsf{T}}\right)^{\mathsf{T}} = \frac{1}{m}\mathbf{X}\mathbf{X}^{\mathsf{T}} = \mathbf{R}. \tag{5.7}$$

Hence, $\mathbf{R}$ is SPSD. □

The eigenvectors of $\mathbf{R}$ are used as vectors for the basis matrix ψ. Note that $\mathbf{R} \in \mathbb{R}^{n\times n}$, instead of computing the eigenvalues and eigenvectors of the correlation matrix $\mathbf{R}$, it is easier to find the eigenvalues and eigenvectors of

$$\tilde{\mathbf{R}} := \frac{1}{l}\mathbf{X}^{\mathsf{T}}\mathbf{X}, \tag{5.8}$$

since the dimension is $l \times l$ and $l \ll n$. Assume $\tilde{\mathbf{R}}$ has eigenvalues defined as

$$\lambda_1 > \lambda_2 > \dots > \lambda_l. \tag{5.9}$$

The relation between the eigenvectors of $\tilde{\mathbf{R}}$ and $\mathbf{R}$ is as follows. If $\mathbf{v}$ is an eigenvector of $\tilde{\mathbf{R}}$, then $\mathbf{Xv}$ is an eigenvector of $\mathbf{R}$. Not every eigenvector is used as basis vector, the dimension m is chosen such that it only represents the m largest eigenvalues of $\mathbf{R}$, where $m \ll l \ll n$. The quantity m is chosen such that the next equality holds

$$\frac{\max\limits_{1\leq m\leq l} \sum\limits_{i=1}^{m} \lambda_i(\mathbf{R})}{\sum\limits_{i=1}^{l} \lambda_i(\mathbf{R})} \leq \alpha, \tag{5.10}$$

where $0 < \alpha \leq 1$ is close to 1. Therefore, the basis matrix $\psi \in \mathbb{R}^{n\times m}$ is defined as

$$\psi := \begin{bmatrix} \psi_1 & \dots & \psi_m \end{bmatrix}, \tag{5.11}$$

where $\{\psi_i\}$ are eigenvectors of the matrix $\mathbf{R}$.

6

Two-Level Preconditioner Conjugate Gradient

The Two-Level Preconditioned Conjugate Gradient method is defined as

$$\mathcal{P}\mathcal{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}, \quad \mathcal{P}, \mathcal{A} \in \mathbb{R}^{n \times n} \tag{6.1}$$

where $\mathcal{P}$ is called an operator and the matrix $\mathcal{A}$ is a combination of the deflation matrix $\mathbf{P}$ and the original matrix $\mathbf{A}$. The operator $\mathcal{P}$ is known as a Two-Level preconditioner since it combines a traditional preconditioner $\mathbf{M}$ and a correction matrix $\mathbf{Q}$. Examples of traditional preconditioners are Jacobi matrix $\mathbf{M} = diag(\mathbf{A})$ and incomplete Cholesky preconditioner with zero fill in $\mathbf{M} = \mathbf{M}_{IC0}$. See Definition 4.5.1 for the definition of the correction matrix. If $\mathcal{P} = \mathbf{I}$ is the identity matrix, $\mathcal{A} = \mathbf{A}$, $\underline{\mathbf{x}} = \mathbf{x}$ and $\underline{\mathbf{b}} = \mathbf{b}$, we get the standard Conjugate Gradient method. If $\mathcal{P} = \mathbf{M}^{-1}$ is a preconditioner and $\underline{\mathbf{b}} = \mathbf{M}^{-1}\mathbf{b}$, Equation 6.1 reduces to the Preconditioned Conjugate Gradient method. There are multiple ways to construct the operator $\mathcal{P}$. This can be done in an additive way or multiplicative way. More details can be found in [26, Sect. 2].

6.1. Additive Preconditioner

Assume that $\mathbf{C}_1, \mathbf{C}_2$ are arbitrary symmetric positive semi-definite (SPSD) preconditioners, then the additive combination

$$\mathcal{P}_{a_2} = \mathbf{C}_1 + \mathbf{C}_2 \tag{6.2}$$

is an SPSD preconditioner. Therefore, a linear combination of different preconditioners with different weight is a preconditioner. As a consequence, the generalization is written as

$$\mathcal{P}_{a_k} = \sum_{i=1}^{k} c_i \mathbf{C}_i, \tag{6.3}$$

where $c_i > 0$ and $\mathbf{C}_i$ is an SPSD preconditioner.

6.2. Multiplicative Preconditioner

Assume $\mathbf{C}_1, \mathbf{C}_2$ are arbitrary SPSD preconditioners, then the two preconditioners can be combined in the following way:

$$\mathbf{x}^{i+\frac{1}{2}} = \mathbf{x}^i + \mathbf{C}_1(\mathbf{b} - \mathbf{A}\mathbf{x}^i), \tag{6.4}$$

$$\mathbf{x}^{i+1} = \mathbf{x}^{i+\frac{1}{2}} + \mathbf{C}_2(\mathbf{b} - \mathbf{A}\mathbf{x}^{i+\frac{1}{2}}). \tag{6.5}$$

With this method, we obtain the following preconditioner

$$\mathcal{P}_{m_2} = \mathbf{C}_1 + \mathbf{C}_2 - \mathbf{C}_2\mathbf{A}\mathbf{C}_1. \tag{6.6}$$

$\mathcal{P}_{m_2}$ can be interpreted as a multiplicative operator constructed from two preconditioners [25, 26]. Similarly, we can combine three preconditioners $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ to get

$$\mathcal{P}_{m_3} = \mathbf{C}_1 + \mathbf{C}_2 + \mathbf{C}_3 - \mathbf{C}_2\mathbf{A}\mathbf{C}_1 - \mathbf{C}_3\mathbf{A}\mathbf{C}_2 - \mathbf{C}_3\mathbf{A}\mathbf{C}_1 + \mathbf{C}_3\mathbf{A}\mathbf{C}_2\mathbf{A}\mathbf{C}_1. \tag{6.7}$$

This method can also be generalized to $\mathcal{P}_{m_k}$ by using k preconditioners.

6.3. Deflation method

Recall that the Deflated Preconditioned Conjugated Gradient (DPCG) has been defined in Section 4.6 and is written as:

$$\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{P}\mathbf{b}, \tag{6.8}$$

where $\hat{\mathbf{x}}$ is the non unique solution of this system. The unique solution of the original system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is found by using

$$\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^{\mathsf{T}}\hat{\mathbf{x}}, \tag{6.9}$$

This method can be written in the form of a Two-Level PCG method by substituting

$$\mathcal{P} = \mathbf{M}^{-1}, \quad \mathcal{A} = \mathbf{P}\mathbf{A} \quad \text{and } \underline{\mathbf{b}} = \mathbf{M}^{-1}\mathbf{P}\mathbf{b}. \tag{6.10}$$

This method is also known as "Deflation Variant 1" (DEF1).

An alternative way to describe the deflation technique has been proposed by [11, 25, 26]: Let $\overline{\mathbf{x}}$ be an arbitrary starting vector and we define the special starting vector as

$$\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}. \tag{6.11}$$

Then, the solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be constructed in the form:

$$\mathbf{x} = \mathbf{x}^0 + \mathbf{P}^{\mathsf{T}}\mathbf{y}, \tag{6.12}$$

where $\mathbf{y}$ is the unique solution of the deflated system:

$$\mathbf{A}\mathbf{P}^{\mathsf{T}}\mathbf{y} = \mathbf{r}_0, \quad \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0. \tag{6.13}$$

The latter expression can be solved using a preconditioner, leading to

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{P}^{\mathsf{T}}\mathbf{y} = \mathbf{M}^{-1}\mathbf{r}_0, \tag{6.14}$$

after which Equation (6.12) to find solution $\mathbf{x}$. Multiplying with $\mathbf{P}^{\mathsf{T}}$ on both sides will give us

$$\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{b}. \tag{6.15}$$

The resulting algorithm will be referred to "Deflation Variant 2" (DEF2).

The difference between DEF1 and DEF2 lies in their flipped two-level preconditioner. To obtain the unique solution, the operation $\mathbf{w} = \mathbf{Q}\mathbf{b} + \mathbf{P}^{\mathsf{T}}\overline{\mathbf{w}}$ is executed after the iteration steps in the DEF1 method, while it is executed before in the DEF2 method. Therefore, the methods have different robustness properties. More details can be found in [25, 26].

6.4. Adapted Deflation methods

If we use $\mathbf{C}_1 = \mathbf{Q}$ and $\mathbf{C}_2 = \mathbf{M}^{-1}$ in Equation (6.6), where $\mathbf{M}$ is a traditional preconditioner, we obtain as preconditioner

$$\mathcal{P}_{\text{A-DEF1}} = \mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}. \tag{6.16}$$

This method will be called Adapted Deflation Variant 1 (A-DEF1). Similarly, $\mathcal{P}_{\text{A-DEF2}}$ can be constructed by using $\mathbf{C}_1 = \mathbf{M}^{-1}$ and $\mathbf{C}_2 = \mathbf{Q}$ in Equation (6.6) to obtain

$$\mathcal{P}_{\text{A-DEF2}} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1} + \mathbf{Q}. \tag{6.17}$$

As a consequence, the operators of the adapted methods are not symmetric. The difference between the DEF method and the A-DEF method lies in the addition of the correction matrix $\mathbf{Q}$.

6.5. Reduced Order Model-based

The operator $\mathcal{P}$ can be constructed by using the algebraic multi-grid methods (AMG) approach. This two-level preconditioner $\mathcal{P}_{\text{ROM}}$ is an approximation of the inverse of $\mathbf{A}$ and has been proposed by [21]. The preconditioner is given by:

$$\mathcal{P}_{\text{ROM}} = \mathbf{M}^{-1} + \mathbf{Q}(1 - \mathbf{A}\mathbf{M}^{-1}), \tag{6.18}$$

where $\mathbf{Q} = \mathbf{Z}^{\mathsf{T}}\mathbf{E}^{-1}\mathbf{Z}$ and $\mathbf{E} = \mathbf{Z}^{\mathsf{T}}\mathbf{A}\mathbf{Z}$ are defined in Definition 4.5.1. Note that $\mathcal{P}_{\text{ROM}}$ is not always symmetric. To obtain the SPD variant, using the formula $\frac{\mathbf{A}+\mathbf{A}^{\mathsf{T}}}{2}$ is considered instead. The symmetric version of the operator of the ROM method is defined as:

$$\mathcal{P}_{\text{SROM}} = \mathbf{M}^{-1} + \mathbf{Q} - \frac{1}{2}\left(\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q}\right). \tag{6.19}$$

We prove in Lemma 7.3.2 of Section 7.3.1 that the Two-Level preconditioner of ROM is the same as A-DEF2. Hence, $\mathcal{P}_{\text{SROM}}$ is an additive preconditioner consisting of the operator $\mathcal{P}_{\text{A-DEF1}}$ and $\mathcal{P}_{\text{A-DEF2}}$ and can also be written as

$$\mathcal{P}_{\text{SROM}} = \frac{1}{2}(\mathcal{P}_{\text{A-DEF1}} + \mathcal{P}_{\text{A-DEF2}}). \tag{6.20}$$

6.6. Abstract Balancing Methods

As mentioned above, $\mathcal{P}_{\text{A-DEF1}}$, $\mathcal{P}_{\text{A-DEF2}}$ are not always symmetric operators. Another way to construct a symmetric operator is using the multiplicative method with three preconditioners. If we uses $\mathbf{C}_1 = \mathbf{Q}$, $\mathbf{C}_2 = \mathbf{M}^{-1}$ and $\mathbf{C}_3 = \mathbf{Q}$ in Equation 6.7, the result is

$$\mathcal{P}_{\text{BNN}} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}. \tag{6.21}$$

The operator $\mathcal{P}_{\text{BNN}}$ is known as the Balancing-Neumann-Neumann (BNN) operator and a well-known operator in the Domain Decomposition Method (DDM) and investigated by [15–17]. This operator is symmetric and therefore, an SPD preconditioner. Moreover, we will consider a reduced version of the BNN operator by removing the correction matrix $\mathbf{Q}$. This will give us

$$\mathcal{P}_{\text{R-BNN1}} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{P}, \tag{6.22}$$

which still is a symmetric preconditioner. Furthermore, if we reduce the preconditioner, we obtain

$$\mathcal{P}_{\text{R-BNN2}} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}. \tag{6.23}$$

This is similar to DEF2 and the only difference lies in the implementation of the method. Both $\mathcal{P}_{\text{R-BNN1}}$ and $\mathcal{P}_{\text{R-BNN2}}$ have the same properties as $\mathcal{P}_{\text{BNN}}$ with the special starting vector. More details can be found in [25, 26].

6.7. Overview of methods II

The formula for the Two-level Preconditioned Conjugated Gradient method is given by

$$\mathcal{P}\mathcal{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}, \quad \mathcal{P}, \mathcal{A} \in \mathbb{R}^{n\times n}. \tag{6.24}$$

Let $\mathbf{A}$ be the original system, $\mathbf{P}$ the deflation matrix, $\mathbf{Q}$ the correction matrix, $\mathbf{Z}$ the deflation subspace matrix and $\mathbf{M}$ the traditional SPD preconditioner. For the Two-Level PCG method, let $\mathcal{A} = \mathbf{A}$ and $\mathcal{P}$ be the preconditioner operator. We define $\underline{\mathbf{x}} = \mathbf{x}$ and $\underline{\mathbf{b}} = \mathcal{P}\mathbf{b}$. The various mentioned methods with initial condition are given in Table 6.1.

Table 6.1: Overview of the various methods

Name	Method	Initial $\mathbf{x}^0$	Operator $\mathcal{P}$
PREC	Traditional Preconditioned CG	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}$
DEF1	Deflation Variant 1	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}\mathbf{P}$
DEF2	Deflation Variant 2	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1}$
A-DEF1	Adapted Deflation Variant 1	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}$
A-DEF2	Adapted Deflation Variant 2	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{Q}$
BNN	Balancing-Neumann-Neumann	$\overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}$
R-BNN1	Reduced Balancing Variant 1	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}$
R-BNN2	Reduced Balancing Variant 2	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1}$
ROM	ROM-based preconditioner	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q}(1 - \mathbf{A}\mathbf{M}^{-1})$
SROM	SROM-based preconditioner	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q} - \frac{1}{2}\left(\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q}\right)$

The algorithm of the Two-Level preconditioners is presented in Algorithm 6 and the corresponding matrices of each method are given in Table 6.2. This implementation is used for the numerical experiments in Chapter 9.

Algorithm 6 Generalized Two-Level Preconditioner Method

Initial: $\overline{\mathbf{x}}, \mathcal{V}_{\text{start}}, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{V}_{\text{end}}, \varepsilon$
Set $\mathbf{x}^0 = \mathcal{V}_{\text{start}}$
Compute: $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\mathbf{z}^0 = \mathcal{M}_1\mathbf{r}^0$ and $\mathbf{p}^0 = \mathcal{M}_2\mathbf{z}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\mathbf{w}^k = \mathcal{M}_3\mathbf{A}\mathbf{p}$
        $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{z}^k \rangle}{\langle \mathbf{p}^k, \mathbf{w}^k \rangle}$
        $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{p}^k$
        $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k\mathbf{A}\mathbf{p}^k$
        $\mathbf{z}^{k+1} = \mathcal{M}_1\mathbf{r}^{k+1}$
        $\beta^k = \frac{\langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{z}^k, \mathbf{r}^k \rangle}$
        $\mathbf{p}^{k+1} = \mathcal{M}_2\mathbf{z}^{k+1} + \beta^k\mathbf{p}^k$
$\mathbf{x}_{it} = \mathcal{V}_{\text{end}}$

Table 6.2: Choices for parameters for various method used in Algorithm 6.

Name	$\mathcal{V}_{\text{start}}$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{V}_{\text{end}}$
PREC	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
DEF1	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}$	$\mathbf{I}$	$\mathbf{P}$	$\mathbf{Qb} + \mathbf{P}^{\mathsf{T}}\mathbf{x}_{k+1}$
DEF2	$\mathbf{Qb} + \mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}$	$\mathbf{M}^{-1}$	$\mathbf{P}^{\mathsf{T}}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
A-DEF1	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
A-DEF2	$\mathbf{Qb} + \mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}$	$\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1} + \mathbf{Q}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
BNN	$\overline{\mathbf{x}}$	$\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
R-BNN1	$\mathbf{Qb} + \mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}$	$\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{P}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
R-BNN2	$\mathbf{Qb} + \mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}$	$\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
ROM	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q}(1 - \mathbf{A}\mathbf{M}^{-1})$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
SROM	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q} - \frac{1}{2}\left(\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q}\right)$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$

7

Theoretical Comparison between Two-Level Preconditioners

In this Chapter, different Two-Level preconditioners discussed above are compared. First, we compare the costs of computational complexity of each method. This will be done for the initial step and iteration step. Thereafter, we also show the memory storage of each method. The computational complexity depends on the sparsity of the matrix and the number of deflation vectors. Next, we present the eigenvalue distribution corresponding to the preconditioned matrix using a Two-Level preconditioner and equivalence lemma's will be proved. Finally, concluding remarks will be given.

7.1. Computational Complexity

The number of flops per iteration has been calculated and more details can be found in Appendix C. The matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ has a sparsity of sn per row and the deflation subspace-matrix $\mathbf{Z} \in \mathbb{R}^{n \times m}$ is a full matrix. The overview of the flops can be found in Table 7.1 and the memory storage can be found in Table 7.3.

Table 7.1: Overview table of the flop count of the above discussed methods.

Methods	**Flops**	
	Initial	Iterations
CG	$(2s+2)n-1$	$(2s+9)n$
PCG	$\frac{1}{2}(11s+1)n-1$	$(4s+10)n$
DCG	$(6m+2s+9)mn+(2s+2)n+\frac{1}{3}m^3-m^2-3m-1$	$(4m+2s+9)n-m$
DEF1	$(6m+2s+9)mn+\frac{1}{2}(11s+1)n+\frac{1}{3}m^3-m^2-3m-1$	$(4m+4s+10)n-m$
DEF2	$(6m+2s+9)mn+\frac{1}{2}(11s+1)n+\frac{1}{3}m^3-m^2-3m-1$	$(4m+4s+10)n-m$
A-DEF1	$(6m+2s+3)mn+\frac{1}{2}(11s+3)n+\frac{1}{3}m^3-m^2-m-1$	$(6m+4s+10)n-m$
A-DEF2	$(6m+2s+13)mn+\frac{1}{2}(11s+3)n+\frac{1}{3}m^3-m^2-4m-1$	$(8m+4s+10)n-2m$
BNN	$(6m+2s+9)mn+\frac{1}{2}(11s+3)n+\frac{1}{3}m^3-m^2-3m-1$	$(12m+4s+10)n-3m$
R-BNN1	$(6m+2s+13)mn+\frac{1}{2}(11s+3)n+\frac{1}{3}m^3-m^2-4m-1$	$(8m+4s+10)n-2m$
R-BNN2	$(6m+2s+9)mn+\frac{1}{2}(11s+3)n+\frac{1}{3}m^3-m^2-4m-1$	$(4m+4s+10)n-m$
ROM	$(4m+2s+2)mn+\frac{1}{2}(15s+3)n+\frac{1}{3}m^3-m^2-m-1$	$(4m+6s+10)n-m$
SROM	$(2m+4s+8)mn+\frac{1}{2}(11s+7)n+\frac{1}{3}m^3+3m^2-1$	$(8m+4s+12)n+4m^2$

We focus on the number of flops per iterations. The flops in terms of order n are given in Table 7.2.

Table 7.2: Overview table of the iteration flop count of the above discussed methods in order n.

Methods	Flops per Iteration
CG	$(2s+9)n$
PCG	$(4s+10)n$
DCG	$(4m+2s+9)n$
DEF1	$(4m+4s+10)n$
DEF2	$(4m+4s+10)n$
A-DEF1	$(6m+4s+10)n$
A-DEF2	$(8m+4s+10)n$
BNN	$(12m+4s+10)n$
R-BNN1	$(8m+4s+10)n$
R-BNN2	$(4m+4s+10)n$
ROM	$(4m+6s+10)n$
SROM	$(8m+4s+12)n$

Clearly, the cheapest methods per iteration of the deflation methods are the DEF1 method, the DEF2 method, the ROM method and the R-BNN2 method, all using $\mathcal{O}(4mn)$ flops per iteration. The most expensive method is the BNN method. Then, the R-BNN1 method and the SROM PCG method are expensive with using $\mathcal{O}(8mn)$ flops per iteration.

7.2. Memory Storage

The memory storage of each method can be found in Table 7.3. All the deflation methods use approximately the same memory storage. For more details, we refer to Appendix C.

Table 7.3: Overview table of the memory storage of the above discussed methods.

Methods	Memory positions
CG	$(5+s)n+4$
PCG	$\frac{1}{2}(3s+13)n+4$
DCG	$(4m+s+6)n+4$
DEF1	$\frac{1}{2}(8m+3s+13)n+4$
DEF2	$\frac{1}{2}(8m+3s+13)n+4$
A-DEF1	$\frac{1}{2}(8m+3s+13)n+4$
A-DEF2	$\frac{1}{2}(8m+3s+13)n+4$
BNN	$\frac{1}{2}(8m+3s+13)n+4$
R-BNN1	$\frac{1}{2}(8m+3s+13)n+4$
R-BNN2	$\frac{1}{2}(8m+3s+13)n+4$
ROM	$\frac{1}{2}(6m+3s+13)n+4$
SROM	$\frac{1}{2}(6m+3s+13)n+\frac{1}{2}(m+1)m+4$

7.3. Comparison of the Spectrum

The convergence of the methods depends on the condition number of the matrix. Using the right Two-Level preconditioner, $\mathcal{P}\mathbf{A}$ will have a smaller conditioner number and resulting faster convergence. It is shown in Section 4.7 that the condition number of using the DEF1 method is equal or lower than the preconditioned system. In this section, the theoretical behaviour of the eigenvalues of the Two-Level preconditioner

applied to the matrix $\mathbf{A}$ will be analyzed. Recall, the definition of deflation method is given by:

Definition 7.3.1. *Given an $\mathbf{A} \in \mathbb{R}^{n\times n}$ which is SPD and given a deflation-subspace matrix $\mathbf{Z}$ of size $n \times m$ where $m \ll n$, the deflation method is defined as*

$$\mathbf{P} = \mathbf{I} - \mathbf{AQ} \qquad \mathbf{P} \in \mathbb{R}^{n\times n}, \quad \mathbf{Q} \in \mathbb{R}^{n\times n}, \tag{7.1}$$

where $\mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T$ with $\mathbf{Z} \in \mathbb{R}^{n\times m}$, $\mathbf{E} = \mathbf{Z}^T\mathbf{A}\mathbf{Z}$ and $\mathbf{E} \in \mathbb{R}^{m\times m}$. This results in the following properties:

Lemma 7.3.1. *Let $\mathbf{A}, \mathbf{P}, \mathbf{Q}$ and $\mathbf{Z}$ be defined as in Definition 7.3.1. Then,*

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

b. $\mathbf{PA} = \mathbf{AP}^T$

c. $\mathbf{QA} = \mathbf{I} - \mathbf{P}^T$

d. $\mathbf{P}^T\mathbf{Z} = \mathbf{P}^T\mathbf{Q} = \mathbf{0}$

e. $\mathbf{PAZ} = \mathbf{PAQ} = \mathbf{0}$

f. $\mathbf{Q}^T = \mathbf{Q}$

g. $\mathbf{QAQ} = \mathbf{Q}$

h. $\mathbf{QAP}^T = 0$

i. $\mathbf{QP} = 0$

j. $\mathbf{QAZ} = \mathbf{Z}$

The proofs can be found in [26, Sect. 3].

7.3.1. Theoretical Comparison of the A-DEF2 method and the ROM method

The Two-Level preconditioner of the A-DEF2 method and the ROM method are similar, as will be proven in the next Lemma:

Lemma 7.3.2. *The A-DEF2 method and the ROM method have the same Two-Level preconditioners.*

Proof. Recall that the Two-Level preconditioners are defined as

$$\begin{aligned} \mathcal{P}_{\text{A-DEF2}} &= \mathbf{P}^T\mathbf{M}^{-1} + \mathbf{Q} \\ \mathcal{P}_{\text{ROM}} &= \mathbf{M}^{-1} + \mathbf{Q}(\mathbf{I} - \mathbf{AM}^{-1}). \end{aligned}$$

Continuing,

$$\begin{aligned} \mathcal{P}_{\text{A-DEF2}} &= \mathbf{P}^T\mathbf{M}^{-1} + \mathbf{Q} && (7.2)\\ &= (\mathbf{I} - \mathbf{QA})\mathbf{M}^{-1} + \mathbf{Q} && (7.3)\\ &= \mathbf{M}^{-1} - \mathbf{QAM}^{-1} + \mathbf{Q} && (7.4)\\ &= \mathbf{M}^{-1} + \mathbf{Q}(\mathbf{I} - \mathbf{AM}^{-1}) && (7.5)\\ &= \mathcal{P}_{\text{ROM}}. && (7.6) \end{aligned}$$

□

The difference between these two methods is the different starting vector. For the A-DEF2 method, it is needed to solve it for the first step by using the special starting vector given in Equation (6.11). The ROM uses the initial condition as starting vector.

7.3.2. Spectra Analysis of Deflation Methods

In this section, the relation of the spectra between the Two-Level preconditioner applied to the matrix $\mathbf{A}$ is shown. First, we show that using certain Two-Level preconditioners belongs to a class. We will prove that DEF1, DEF2, R-BNN1 and R-BNN2 belongs to the same class. Then, another class consists of A-DEF1, A-DEF2, ROM and BNN. Thereafter, we will prove a lemma that connects these two classes in Lemma 7.3.5. Finally, we will investigate the spectra after applying the SROM Two-Level preconditioner using specific deflation vectors.

Lemma 7.3.3. *The spectra corresponding to the Two-Level preconditioners of DEF1, DEF2, R-BNN1 and R-BNN2 applied to the original matrix* $\mathbf{A}$ *are the same. i.e.*

$$\sigma(\mathbf{M}^{-1}\mathbf{PA}) = \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}) = \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{PA}) \tag{7.7}$$

Proof. It is known that DEF2 and R-BNN2 have the same Two-Level preconditioner, this is given in Table 6.1 in Section 6.7. Hence, the spectra of these methods are the same. Now, we prove the first equality.

$$\begin{aligned}
\sigma(\mathbf{M}^{-1}\mathbf{PA}) &\overset{L.2.3.1a}{=} \sigma(\mathbf{AM}^{-1}\mathbf{P}) && (7.8)\\
&\overset{L.2.3.1c}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}). && (7.9)
\end{aligned}$$

For the second equality we have:

$$\begin{aligned}
\sigma(\mathbf{M}^{-1}\mathbf{PA}) &\overset{L.7.3.1a}{=} \sigma(\mathbf{M}^{-1}\mathbf{P}^2\mathbf{A}) && (7.10)\\
&\overset{L.7.3.1b}{=} \sigma(\mathbf{M}^{-1}\mathbf{PAP}^\top) && (7.11)\\
&\overset{L.2.3.1a}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{PA}). && (7.12)
\end{aligned}$$

□

Next, we prove that the spectra of A-DEF1, A-DEF2, ROM and BNN are the same. Clearly, it has been shown that the Two-Level preconditioners of the A-DEF2 method and the ROM method are the same. Therefore, it is only needed to prove that A-DEF1, A-DEF2, and BNN have the same spectrum. This is proven in the following Lemma:

Lemma 7.3.4. *The spectra corresponding to the Two-Level preconditioners of A-DEF1, A-DEF2, ROM and BNN applied to the original matrix* $\mathbf{A}$ *are the same, i.e.*

$$\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A} + \mathbf{QA}) = \sigma(\mathbf{M}^{-1}\mathbf{PA} + \mathbf{QA}) = \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{PA} + \mathbf{QA}) \tag{7.13}$$

Proof. As shown in Lemma 7.3.2, the Two-Level preconditioners of A-DEF2 and ROM are the same. We prove the first equality.

$$\begin{aligned}
\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A} + \mathbf{QA}) &\overset{L.7.3.1c}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A} + \mathbf{I} - \mathbf{P}^\top) && (7.14)\\
&\overset{L.2.3.1b}{=} \sigma(\mathbf{P}^\top(\mathbf{M}^{-1}\mathbf{A} - I)) + \sigma(\mathbf{I}) && (7.15)\\
&\overset{L.2.3.1a}{=} \sigma((\mathbf{M}^{-1}\mathbf{A} - I)\mathbf{P}^\top) + \sigma(\mathbf{I}) && (7.16)\\
&\overset{L.2.3.1b}{=} \sigma(\mathbf{M}^{-1}\mathbf{AP}^\top - \mathbf{P}^\top + \mathbf{I}) && (7.17)\\
&\overset{L.7.3.1b}{=} \sigma(\mathbf{M}^{-1}\mathbf{PA} - \mathbf{P}^\top + \mathbf{I}) && (7.18)\\
&\overset{L.7.3.1c}{=} \sigma(\mathbf{M}^{-1}\mathbf{PA} + \mathbf{QA}). && (7.19)
\end{aligned}$$

For the following equality we have

$$\begin{aligned}
\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}+\mathbf{Q}\mathbf{A}) &\overset{L.7.3.1c}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}+\mathbf{I}-\mathbf{P}^\top) && (7.20)\\
&\overset{L.2.3.1b}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}-\mathbf{P}^\top)+\sigma(\mathbf{I}) && (7.21)\\
&\overset{L.7.3.1b}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}\mathbf{P}^\top-\mathbf{P}^\top)+\sigma(\mathbf{I}) && (7.22)\\
&\overset{L.2.3.1c}{=} \sigma(\mathbf{P}\mathbf{A}\mathbf{M}^{-1}\mathbf{P}-\mathbf{P})+\sigma(\mathbf{I}) && (7.23)\\
&= \sigma(\mathbf{P}(\mathbf{A}\mathbf{M}^{-1}\mathbf{P}-I))+\sigma(\mathbf{I}) && (7.24)\\
&\overset{L.2.3.1a}{=} \sigma((\mathbf{A}\mathbf{M}^{-1}\mathbf{P}-\mathbf{I})\mathbf{P})+\sigma(\mathbf{I}) && (7.25)\\
&= \sigma(\mathbf{A}\mathbf{M}^{-1}\mathbf{P}^2-\mathbf{P})+\sigma(\mathbf{I}) && (7.26)\\
&\overset{L.7.3.1a}{=} \sigma(\mathbf{A}\mathbf{M}^{-1}\mathbf{P}-\mathbf{P})+\sigma(\mathbf{I}) && (7.27)\\
&\overset{L.2.3.1c}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}-\mathbf{P}^\top)+\sigma(\mathbf{I}) && (7.28)\\
&\overset{L.2.3.1b}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}-\mathbf{P}^\top+\mathbf{I}) && (7.29)\\
&\overset{L.7.3.1c}{=} \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}-\mathbf{Q}\mathbf{A}) && (7.30)
\end{aligned}$$

This concludes the proof. □

Because of the result of the previous lemma, it is not needed to investigate the spectrum of every method separately. The methods can be divided into different classes. The first class consists of DEF1, DEF2, R-BNN1 and R-BNN2. While another class consists of BNN, A-DEF1, A-DEF2 and ROM. Therefore, it is enough to prove using one method of each class, i.e. DEF1 and BNN. Then, we arrive to the next Lemma.

Lemma 7.3.5. *Let the spectra of DEF1 and BNN applied to the original system* $\mathbf{A}$ *be given by:*

$$\sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A})=\{\lambda_1,\dots,\lambda_n\},\quad \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}+\mathbf{Q}\mathbf{A})=\{\mu_1,\dots,\mu_n\} \tag{7.31}$$

respectively. Then, the eigenvalues within these spectra can be reordered such that

$$\lambda_i=0,\quad \mu_i=1,\quad i=1,\dots,m \tag{7.32}$$

and

$$\lambda_i=\mu_i,\quad i=m+1,\dots,n. \tag{7.33}$$

Proof. First, we consider the Two-Level preconditioner of BNN and it follows from Lemma 7.3.1 that

$$(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}+\mathbf{Q})\mathbf{A}\mathbf{Z}=\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{Z}+\mathbf{Q}\mathbf{A}\mathbf{Z}\overset{L.7.3.1e}{=}\mathbf{0}+\mathbf{Q}\mathbf{A}\mathbf{Z}\overset{L.7.3.1j}{=}\mathbf{Z}. \tag{7.34}$$

Then, applying this to the Two-Level preconditioner of DEF1 we get

$$\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{Z}\overset{L.7.3.1e}{=}0. \tag{7.35}$$

It follows that de columns of $\mathbf{Z}$ are the eigenvectors of $\mathcal{P}_{\text{BNN}}\mathbf{A}$ correspond to the eigenvalue equal to 1. The same set of vectors are eigenvectors of $\mathcal{P}_{\text{DEF1}}\mathbf{A}$ corresponding to the eigenvalue equal to 0. Next, from Theorem 2.8 found in [19], it is sufficient to proof that if

$$\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}+\mathbf{Q}\mathbf{A})=\{1,\dots,1,\mu_{m+1},\dots,\mu_n\} \tag{7.36}$$

holds, then

$$\sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A})=\{0,\dots,0,\mu_{m+1},\dots,\mu_n\}. \tag{7.37}$$

Let the eigenvalue μ_i correspond with $\mathbf{v}_i$, where $i = m+1, \dots n$. Then

$$(\mathbf{P}^\top \mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{A}\mathbf{v}_i = \mu_i \mathbf{v}_i \tag{7.38}$$

implies

$$\mathbf{P}^\top(\mathbf{P}^\top \mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{A}\mathbf{v}_i = \mu_i \mathbf{P}^\top \mathbf{v}_i. \tag{7.39}$$

Now, we rewrite the left-hand side of Equation (7.39) to get

$$\begin{aligned} \mathbf{P}^\top(\mathbf{P}^\top \mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{A} &= (\mathbf{P}^\top)^{\mathbf{2}}\mathbf{M}^{-1}\mathbf{P}\mathbf{A} + \mathbf{P}^\top\mathbf{Q}\mathbf{A} && (7.40)\\ &\overset{L.7.3.1a}{=} \mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}^{\mathbf{2}}\mathbf{A} + \mathbf{P}^\top\mathbf{Q}\mathbf{A} && (7.41)\\ &\overset{L.7.3.1b}{=} \mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{P}^\top + \mathbf{P}^\top\mathbf{Q}\mathbf{A} && (7.42)\\ &\overset{L.7.3.1d}{=} \mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{P}^\top. && (7.43) \end{aligned}$$

Equation (7.39) can now be written as

$$\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{w}_i = \mu_i \mathbf{w}_i, \tag{7.44}$$

where $\mathbf{w}_i := \mathbf{P}^\top \mathbf{v}_i$. Note that $\mathbf{w}_i \neq 0$ since that follows from Lemma 7.3.1d and $\mathbf{v}_i \notin \text{Col}(\mathbf{Z})$. Hence, μ_i is also an eigenvalue of $\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}$. Therefore, from Lemma 7.3.5 we have

$$\sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A}) = \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}) \tag{7.45}$$

such that μ_i is an eigenvalue of DEF1. □

7.3.3. Spectrum Analysis of SROM

Recall that the Two-Level preconditioner of SROM is a linear combination of A-DEF1 and A-DEF2. To investigate this Two-Level preconditioner, we start by using the identity as traditional matrix and the eigenvectors of $\mathbf{A}$ as deflation matrix. Then, the theorem using an arbitrary traditional preconditioner can be generalized by using eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$.

Eigenvectors as Deflation Vectors

In this section, we use eigenvectors of $\mathbf{A}$ as deflation vectors and proof a few properties given in Theorem 7.3.1.

Theorem 7.3.1. *Suppose we have $\mathbf{A} \in \mathbb{R}^{n\times n}$ with spectrum $\sigma(A) = \{\lambda_1, \dots, \lambda_n\}$. The eigenvectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are chosen to be orthogonal, i.e. $\mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}$, so that*

$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i. \tag{7.46}$$

Suppose the deflation subspace-matrix $\mathbf{Z} \in \mathbb{R}^{n\times m}$ consists of m eigenvectors of $\mathbf{A}$, i.e.

$$\mathbf{Z} = \begin{bmatrix}\mathbf{v}_1 \dots \mathbf{v}_m\end{bmatrix}. \tag{7.47}$$

The complement of the deflation sub-space matrix is defined as

$$\mathbf{Z}^{\mathrm{c}} = \begin{bmatrix}\mathbf{v}_{m+1} \dots \mathbf{v}_n\end{bmatrix}. \tag{7.48}$$

Let $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m) \in \mathbb{R}^{m\times m}$ and $\Lambda^{\mathrm{c}} = \text{diag}(\lambda_{m+1}, \dots, \lambda_n)$. Let the inverse be denoted by $\Lambda^{-1} = \text{diag}(\lambda_1^{-1}, \dots, \lambda_m^{-1})$. Then the following statements hold:

(a) $\mathbf{A}\mathbf{Z} = \mathbf{Z}\Lambda$

(b) $\mathbf{E} = \Lambda$ *and* $\mathbf{E}^{-1} = \Lambda^{-1}$.

(c) $\mathbf{QZ} = \mathbf{Z}\Lambda^{-1}$ *and* $\mathbf{QZ}^{\mathsf{C}} = 0$

(d) $\mathbf{PZ} = 0$ *and* $\mathbf{PZ}^{\mathsf{C}} = \mathbf{Z}^{\mathsf{C}}$

(e) $\mathbf{P}^{\mathsf{T}}\mathbf{Z} = 0$ *and* $\mathbf{P}^{\mathsf{T}}\mathbf{Z}^{\mathsf{C}} = \mathbf{Z}^{\mathsf{C}}$

Proof. (a) $\mathbf{AZ} = \mathbf{Z}\Lambda$

$$\mathbf{AZ} = \mathbf{A}\left[\mathbf{v}_1 \dots \mathbf{v}_m\right] = \left[\mathbf{A}\mathbf{v}_1 \dots \mathbf{A}\mathbf{v}_m\right] = \left[\lambda_1\mathbf{v}_1 \dots \lambda_m\mathbf{v}_m\right] = \left[\mathbf{v}_1 \dots \mathbf{v}_m\right]\Lambda = \mathbf{Z}\Lambda \tag{7.49}$$

(b) $\mathbf{E} = \Lambda$ and $\mathbf{E}^{-1} = \Lambda^{-1}$.

$$\mathbf{E} = \mathbf{Z}^{\mathsf{T}}\mathbf{AZ} = \mathbf{Z}^{\mathsf{T}}\mathbf{Z}\Lambda = \Lambda \tag{7.50}$$

The last equality holds because of the orthogonality property. Because Λ is a diagonal matrix, we have that

$$\mathbf{E}^{-1} = \Lambda^{-1} \tag{7.51}$$

(c) First, we prove $\mathbf{Q}\mathbf{v}_i = \begin{cases} \frac{1}{\lambda_i}\mathbf{v}_i & \text{for } \mathbf{v}_i \in \mathbf{Z} \\ 0 & \text{for } \mathbf{v}_i \notin \mathbf{Z} \end{cases}$

Let $\mathbf{v}_i \in \text{Col}(\mathbf{Z})$, then

$$\begin{aligned} \mathbf{Q}\mathbf{v}_i &= \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^{\mathsf{T}}\mathbf{v}_i && (7.52)\\ &= \left[\mathbf{v}_1 \dots \mathbf{v}_m\right]\Lambda^{-1}\left[\mathbf{v}_1 \dots \mathbf{v}_m\right]^{\mathsf{T}}\mathbf{v}_i && (7.53)\\ &= \left[\mathbf{v}_1 \dots \mathbf{v}_m\right]\Lambda^{-1}\mathbf{e}_i && (7.54)\\ &= \left[\mathbf{v}_1 \dots \mathbf{v}_m\right]\frac{1}{\lambda_i} = \frac{1}{\lambda_i}\mathbf{v}_i. && (7.55) \end{aligned}$$

where $\mathbf{e}_i$ is a unit vector with 1 on position i. If $\mathbf{v}_i \notin \text{Col}(\mathbf{Z})$, we know that $\mathbf{Z}^{\mathsf{T}}\mathbf{v}_i = 0$ because of the orthogonality property. It follows that

$$\mathbf{QZ} = \left[\mathbf{Q}\mathbf{v}_1 \dots \mathbf{Q}\mathbf{v}_m\right] = \left[\tfrac{1}{\lambda_1}\mathbf{v}_1 \dots \tfrac{1}{\lambda_m}\mathbf{v}_m\right] = \mathbf{Z}\Lambda^{-1} \tag{7.56}$$

and

$$\mathbf{QZ}^{\mathsf{C}} = 0. \tag{7.57}$$

(d) We know that

$$\mathbf{P} = \mathbf{I} - \mathbf{AQ}. \tag{7.58}$$

Thus,

$$\mathbf{PZ} = (\mathbf{I} - \mathbf{AQ})\mathbf{Z} = \mathbf{Z} - \mathbf{AQZ} = \mathbf{Z} - \mathbf{AZ}\Lambda^{-1} = \mathbf{Z} - \mathbf{Z} = 0. \tag{7.59}$$

Likewise,

$$\mathbf{PZ}^{\mathsf{C}} = (\mathbf{I} - \mathbf{AQ})\mathbf{Z}^{\mathsf{C}} = \mathbf{Z}^{\mathsf{C}} - \mathbf{AQZ}^{\mathsf{C}} = \mathbf{Z}^{\mathsf{C}}. \tag{7.60}$$

The last equality follows from Theorem 7.3.1c.

(e) We know from Lemma 7.3.1d that

$$\mathbf{P}^{\mathsf{T}}\mathbf{Z} = 0 \tag{7.61}$$

On the other hand,

$$\begin{aligned}\mathbf{P}^\top\mathbf{Z}^c &= (\mathbf{I}-\mathbf{QA})\mathbf{Z}^c = \mathbf{Z}^c - \mathbf{QAZ}^c && (7.62)\\ &= \mathbf{Z}^c - \mathbf{QZ}^c\Lambda^c && (7.63)\\ &= \mathbf{Z}^c && (7.64)\end{aligned}$$

The last equality follows from Theorem 7.3.1c.

□

Lemma 7.3.1 will be used to investigate the spectrum of $\mathcal{P}_{\text{SROM}}\mathbf{A}$.

Lemma 7.3.6. *Assume $\mathbf{A} \in \mathbb{R}^{n\times n}$ has eigenvalues $\sigma(\lambda_1, \dots, \lambda_n)$ with λ_i corresponding to eigenvector $\mathbf{v}_i$ and let $\mathbf{M} = \mathbf{I}$ be the traditional preconditioner. If the deflation subspace matrix is defined as $\mathbf{Z} = [\mathbf{v}_1 \dots \mathbf{v}_m]$, then $\mathcal{P}_{SROM}\mathbf{A}$ has eigenvalues $\{1, \dots, 1, \lambda_{m+1}, \dots, \lambda_n\}$ and the same eigenvectors as $\mathbf{A}$.*

Proof. First, we prove $\mathbf{Z}$ are eigenvectors of $\mathcal{P}_{\text{SROM}}\mathbf{A}$.

$$\begin{aligned}\mathcal{P}_{\text{SROM}}\mathbf{AZ} &= \left[\mathbf{I}+\mathbf{Q}-\frac{1}{2}(\mathbf{QA}+\mathbf{AQ})\right]\mathbf{AZ} && (7.65)\\ &= \mathbf{AZ}+\mathbf{QAZ}-\frac{1}{2}(\mathbf{QA}+\mathbf{AQ})\mathbf{AZ} && (7.66)\\ &\overset{L.7.3.1c}{=} \mathbf{AZ}+\mathbf{QAZ}-\frac{1}{2}\big(2\mathbf{I}-\mathbf{P}^\top-\mathbf{P}\big)\mathbf{AZ} && (7.67)\\ &= \mathbf{AZ}+\mathbf{QAZ}-\mathbf{AZ}+\frac{1}{2}\mathbf{P}^\top\mathbf{AZ}+\frac{1}{2}\mathbf{PAZ} && (7.68)\\ &\overset{L.7.3.1e,j}{=} \mathbf{Z}+\frac{1}{2}\mathbf{P}^\top\mathbf{AZ} && (7.69)\\ &\overset{Thm.7.3.1a}{=} \mathbf{Z}+\frac{1}{2}\mathbf{P}^\top\mathbf{Z}\Lambda && (7.70)\\ &\overset{Thm.7.3.1d}{=} \mathbf{Z} && (7.71)\end{aligned}$$

It follows that $\mathbf{v}_i \in \text{Col}(\mathbf{Z})$ are eigenvectors corresponding to eigenvalues 1. Continuing,

$$\begin{aligned}\mathcal{P}_{\text{SROM}}\mathbf{AZ}^c &= \left[\mathbf{I}+\mathbf{Q}-\frac{1}{2}(\mathbf{QA}+\mathbf{AQ})\right]\mathbf{AZ}^c && (7.72)\\ &= \mathbf{AZ}^c+\mathbf{QAZ}^c-\frac{1}{2}(\mathbf{QA}+\mathbf{AQ})\mathbf{AZ}^c && (7.73)\\ &\overset{L.7.3.1c}{=} \mathbf{AZ}^c+\mathbf{QAZ}^c-\frac{1}{2}\big(2\mathbf{I}-\mathbf{P}^\top-\mathbf{P}\big)\mathbf{AZ}^c && (7.74)\\ &\overset{L.7.3.1c}{=} \mathbf{AZ}^c+\mathbf{QAZ}^c-\mathbf{AZ}^c+\frac{1}{2}\mathbf{P}^\top\mathbf{AZ}^c+\frac{1}{2}\mathbf{PAZ}^c && (7.75)\\ &\overset{L.7.3.1a}{=} \mathbf{QZ}^c\Lambda^c+\frac{1}{2}\mathbf{P}^\top\mathbf{Z}^c\Lambda^c+\frac{1}{2}\mathbf{PZ}^c\Lambda^c && (7.76)\\ &\overset{L.7.3.1c,e}{=} \frac{1}{2}\mathbf{Z}^c\Lambda^c+\frac{1}{2}\mathbf{Z}^c\Lambda^c=\mathbf{Z}^c\Lambda^c && (7.77)\\ &= \left[\lambda_{m+1}\mathbf{v}_{m+1}\dots\lambda_n\mathbf{v}_n\right]. && (7.78)\end{aligned}$$

It follows that $\mathbf{v}_i \in \text{Col}(\mathbf{Z}^c)$ are eigenvectors corresponding to eigenvalue λ_i, where $i = m+1, \dots n$. □

It follows from Lemma 7.3.6 that the SROM method could belong to the same class as the ROM method, i.e. $\sigma(\mathcal{P}_{\text{SROM}}\mathbf{A}) = \{1, \dots, 1, \lambda_{m+1}, \dots, \lambda_n\}$. We generalize Lemma 7.3.6 with using $\mathbf{M}^{-1}$ as a traditional preconditioner, i.e. $\mathbf{M}$ is SPD, to obtain Lemma 7.3.7.

Lemma 7.3.7. *Assume $\mathbf{M}^{-1}\mathbf{A} \in \mathbb{R}^{n\times n}$ has eigenvalues $\sigma(\lambda_1, \dots, \lambda_n)$ with λ_i corresponding to eigenvector $\mathbf{v}_i$ of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ and take $\mathbf{M}$ as a traditional SPD preconditioner. If the deflation-subspace matrix is defined as $\mathbf{Z} = [\mathbf{v}_1 \dots \mathbf{v}_m]$, then*

$$\sigma(\mathcal{P}_{SROM}\mathbf{A}) = \{1, \dots, 1, \lambda_{m+1}, \dots, \lambda_n\}. \tag{7.79}$$

Proof. First, we assume that the spectrum of SROM applied to the original system $\mathbf{A}$ is defined by

$$\sigma(\mathcal{P}_{\text{SROM}}\mathbf{A}) = \{\mu_i, \dots, \mu_n\}, \tag{7.80}$$

and the spectrum of DEF2 is defined by

$$\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}) = \{\nu_1, \dots, \nu_n\}, \tag{7.81}$$

where the eigenvalues are ordered such that $\nu_i = 0$ for $i = 1, \dots, m$ and $\nu_i = \lambda_i$ for $i = m+1, \dots, n$. Using Lemma 7.3.1c and the definition of deflation method we obtain the following expression:

$$\mathcal{P}_{\text{SROM}} = \mathbf{M}^{-1} + \mathbf{Q} - \frac{1}{2}\left(\mathbf{QAM}^{-1} + \mathbf{M}^{-1}\mathbf{AQ}\right) = \mathbf{Q} + \frac{1}{2}\left(\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{P}\right). \tag{7.82}$$

Let $\mathbf{v}_i \in \text{Col}(Z)$, then

$$\begin{aligned}
\mathcal{P}_{\text{SROM}}\mathbf{Av}_i &= \left[\mathbf{Q} + \frac{1}{2}\left(\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{P}\right)\right]\mathbf{Av}_i && (7.83)\\
&= \mathbf{QAv}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{Av}_i + \frac{1}{2}\mathbf{M}^{-1}\mathbf{PAv}_i && (7.84)\\
&\overset{L.7.3.1e,j}{=} \mathbf{v}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{Av}_i && (7.85)\\
&= \mathbf{v}_i + \frac{1}{2}\lambda_i\mathbf{P}^\top\mathbf{v}_i && (7.86)\\
&\overset{L.7.3.1d}{=} \mathbf{v}_i && (7.87)
\end{aligned}$$

Continuing, let $\mathbf{v}_i \notin \text{Col}(Z)$ be an eigenvector corresponding to eigenvalue μ_i:

$$\begin{aligned}
&\left[\mathbf{Q} + \frac{1}{2}\left(\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{P}\right)\right]\mathbf{Av}_i = \mu_i\mathbf{v}_i && (7.88)\\
\Rightarrow\quad &\mathbf{QAv}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{Av}_i + \frac{1}{2}\mathbf{M}^{-1}\mathbf{PAv}_i = \mu_i\mathbf{v}_i && (7.89)\\
\overset{\mathbf{P}^\top}{\Rightarrow}\quad &\mathbf{P}^\top\mathbf{QAv}_i + \frac{1}{2}(\mathbf{P}^\top)^2\mathbf{M}^{-1}\mathbf{Av}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{PAv}_i = \mu_i\mathbf{P}^\top\mathbf{v}_i && (7.90)\\
\overset{L.7.3.1a,b,d}{\Rightarrow}\quad &\frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{Av}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{AP}^\top\mathbf{v}_i = \mu_i\mathbf{P}^\top\mathbf{v}_i && (7.91)\\
\Rightarrow\quad &\frac{1}{2}\lambda_i\mathbf{P}^\top\mathbf{v}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{AP}^\top\mathbf{v}_i = \mu_i\mathbf{P}^\top\mathbf{v}_i && (7.92)\\
\overset{\mathbf{w}_i:=\mathbf{P}^\top\mathbf{v}_i}{\Rightarrow}\quad &\frac{1}{2}\lambda_i\mathbf{w}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{Aw}_i = \mu_i\mathbf{w}_i && (7.93)\\
\Rightarrow\quad &\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{Aw}_i = (2\mu_i - \lambda_i)\mathbf{w}_i && (7.94)
\end{aligned}$$

Recall that we have $\mathbf{w}_i \neq 0$ since $\mathbf{v}_i \notin \text{Col}(\mathbf{Z})$. It follows that, an eigenvector of $\mathcal{P}_{\text{SROM}}\mathbf{A}$ is an eigenvector of $\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{A}$. The eigenvector $\mathbf{v}_i$ corresponds to eigenvalue $2\mu_i - \lambda_i = \nu_i$, thus $\mu_i = \frac{1}{2}(\nu_i + \lambda_i)$. Recall that the spectrum is ordered such that $\nu_i = 0$ for $i = 1, \dots, m$ and $\nu_i = \lambda_i$. Thus, the spectrum of $\mathcal{P}_{\text{SROM}}\mathbf{A}$ is given by

$$\sigma(\mathcal{P}_{\text{SROM}}\mathbf{A}) = \{1, \dots, 1, \lambda_{m+1}, \dots, \lambda_n\}. \tag{7.95}$$

□

We suspect that the Two-Level preconditioner of SROM belongs in the same class as A-DEF1, A-DEF2, ROM and BNN. This is certainly true for using eigenvectors of the preconditioned system as deflation vectors. The theory will be tested with numerical experiments in Section 9.2.5.

7.4. Concluding Remarks

In this chapter, we discussed theoretical aspects of the studied methods. For the computational complexity, the cheapest methods per iteration are DEF1, DEF2, ROM and R-BNN2, using $\mathcal{O}(4mn)$ flops per iteration. The most expensive method per iteration is BNN using $\mathcal{O}(12mn)$ flops per iteration. The memory storage of each method are similar. For the spectral analysis, the methods can be divided into two classes. One class sets some eigenvalues to zero (DEF1, DEF2, R-BNN1 and R-BNN2), while the other class sets the same eigenvalues to one (BNN, A-DEF1, A-DEF2 and ROM). The relation between those two classes is summarized in Lemma 7.3.5. We have proven that $n - m$ eigenvalues the spectra of the two classes are the same. The spectral of the Two-Level preconditioner of SROM applied to the system $\mathbf{A}$ belongs in the same class as A-DEF1,A-DEF2,BNN,ROM under certain conditions.

8

Test cases

The mathematical model are presented in Chapter 3 and the numerical methods are given in Chapter 4 and Chapter 6. In this section, we discusses different test cases to investigate the methods using Two-Level Preconditioner. We will consider three test cases:

1. Laplace Equation
2. Multilayer Problem
3. SPE10

The Laplace equation is an example of an incompressible model with constant rock permeability. The Multilayer Problem is an example of an incompressible model with different rock permeabilities. Both cases are one-phase flow problem. The SPE10 is an example of a incompressible model of a two-phase problem. The domain of all test cases are two-dimensional. The various methods will be applied to these Test cases. Numerical experiments using these test cases will be given in Chapter 9

8.1. Test Case 1: Laplace Equation

The simple problem of simulating one-phase flow through porous media with a constant rock permeability is the Laplace equation problem. The Laplace equation is a well-known problem and will be explained below.

Problem Statement

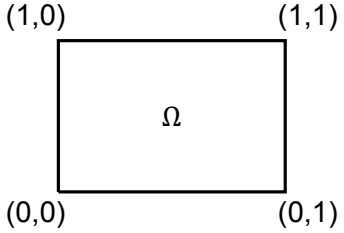


Figure 8.1: Porous Media

Consider the Laplace equation on the unit square $\Omega = [0,1] \times [0,1]$ with homogeneous boundary conditions and rock permeability $\mathbf{K} = 1$. The problem is formulated as:

$$\begin{cases} -\Delta \mathbf{p} &= \mathbf{q} \quad \text{for } \mathbf{x} \in \Omega \\ \mathbf{p} &= 0 \quad \text{for } \mathbf{x} \in \partial\Omega \end{cases}, \tag{8.1}$$

where $\mathbf{p}$ is the pressure and $\mathbf{q}$ is the source vector.

Discretization Scheme

Suppose that we divide each axis of Ω into n equal subintervals with length $h = 1/n$. The matrix $\mathbf{A}$ is constructed by eliminating the boundary conditions. It follows that the size of $\mathbf{A}$ is $(n-1)^2 \times (n-1)^2$. For this problem, we use lexicographic ordering $(i,j) \mapsto i + (j-1)(n-1)$ for $1 \leq i,j \leq n-1$ and define $\overline{\mathbf{p}}(x_i, y_j) = \overline{\mathbf{p}}_{i,j}$. For the internal nodes of the problem, we use the central difference approximation that is defined as:

$$\frac{\partial^2 p}{\partial x^2} \approx \frac{\overline{\mathbf{p}}_{i-1,j} - 2\overline{\mathbf{p}}_{i,j} + \overline{\mathbf{p}}_{i+1,j}}{h^2} + \mathcal{O}(h^2). \tag{8.2}$$

A Similar expression can be obtained for the y-direction. For a cell (i,j) the discretization of second order is given by:

$$\frac{-\overline{\mathbf{p}}_{i-1,j} - \overline{\mathbf{p}}_{i,j-1} + 4\overline{\mathbf{p}}_{i,j} - \overline{\mathbf{p}}_{i+1,j} - \overline{\mathbf{p}}_{i,j+1}}{h^2} = \mathbf{q}_{i,j}. \tag{8.3}$$

This can be summarized in a matrix $\mathbf{A}$ so that:

$$\mathbf{A}\overline{\mathbf{p}} = \mathbf{q}, \tag{8.4}$$

where the source vector $\mathbf{q}$ is a random vector for this problem.

Properties

The matrix $\mathbf{A}$ is SPD and sparse. Figure 8.2 illustrates the sparsity of $\mathbf{A}$. Figure 8.2a shows that $\mathbf{A}$ has size 100×100 and contains 460 nonzero elements. If the size is increased to $\mathbf{A} \in \mathbb{R}^{900\times900}$, the sparsity structure remains the same and it contains 4380 nonzero elements. This can be found in Figure 8.2b.

8.2. Test Case 2: Multilayer Problem

The structure of the porous media of the previous problem only consists of constant rock permeability. In this section, the rock formulation of the porous media consists of two types of layers with different rock permeability.

Problem Statement

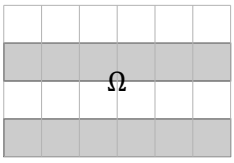
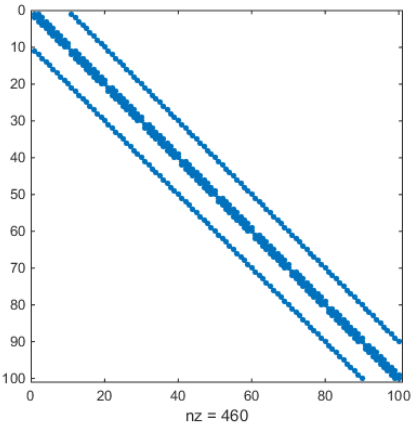
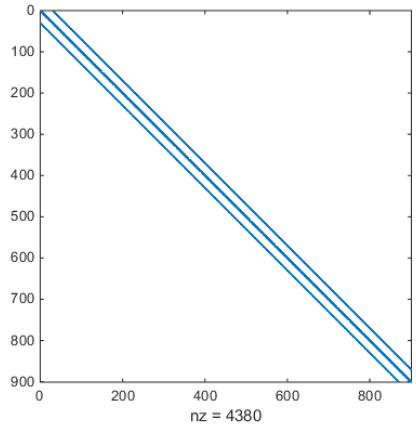


Figure 8.3: Porous media with multiple layers and different permeability

Consider the incompressible model given in Equation (3.12) defined on the unit square $\Omega = [0,1] \times [0,1]$. The boundary conditions are described with $\mathbf{p}$=p_{bc1} bar at the bottom



(a) A is 100×100



(b) A is 900×900

Figure 8.2: Nonzero structure of **A** different sizes for n

part of the porous media, $\mathbf{p}$=p_{bc2} bar at the top part of the porous media and no-flow conditions elsewhere. The flow problem is formulated by:

$$\begin{cases} -\nabla(\mathbf{K}\nabla\mathbf{p}) &= \mathbf{q} & \text{for } \mathbf{x} \in \Omega \\ \mathbf{p}(x,0) &= p_{bc1} & \text{for } 0 \leq x \leq 1 \\ \mathbf{p}(x,1) &= p_{bc2} & \text{for } 0 \leq x \leq 1 \\ \vec{\mathbf{v}} \cdot n &= 0 & \text{Elsewhere} \end{cases}, \tag{8.5}$$

where **K** is the rock permeability matrix, **p** is the pressure and **q** is the source vector.

Discretization Scheme

Suppose we divide each axis of Ω into n equal subintervals with length $h = 1/n$. It follows that the size of **A** is $n^2 \times n^2$. For this problem, we use lexicographic ordering $(i,j) \mapsto i + (j-1)n$ for $1 \leq i,j \leq n$ and define $\overline{\mathbf{p}}(x_i, y_j) = \overline{\mathbf{p}}_{i,j}$. For the internal nodes of the problem, we use central difference approximation defined in test case 1. This can be summarized in a matrix **A** such that:

$$\mathbf{A}\overline{\mathbf{p}} = \mathbf{q}, \tag{8.6}$$

with the source vector **q**. The properties of **A** are the same as those of the Laplace Equation defined in Test Case 1, i.e. **A** is sparse and SPD. The difference lies in the fact that the matrix **A** is ill-conditioned due to the high contrast in rock permeability **K**. More details will be explained in Section 9.2.

8.3. Test Case 3: SPE10 Model

In this section, we perform a two-phase (oil and water) flow simulation for the SPE10 model. It consists of 5 wells, the injection well I is located in the middle of the domain and the other wells are production wells P1-P4. The wells and permeability field is presented in Figure 8.4.

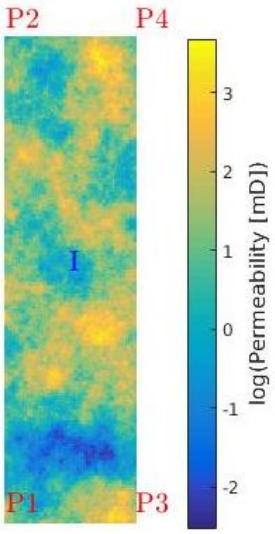


Figure 8.4: Rock permeability of the upper layer of the SPE10 model.

More information about the SPE10 model can be found as Model 1 in [14].

8.4. Termination Criterion

For all three test cases, the system that is solved is of the form

$$\mathbf{Ap} = \mathbf{q}. \tag{8.7}$$

For the termination criterion of the numerical methods we specify the stopping criterion and maximum number of iterations. The relative residual is defined as:

$$\frac{\|\mathbf{q} - \mathbf{A}\bar{\mathbf{p}}^k\|_2}{\|\mathbf{q}\|_2} \leq \epsilon, \tag{8.8}$$

where ϵ is the stopping criterion. The norm of the relative error is defined as:

$$\text{Error} = \frac{\|\mathbf{p} - \bar{\mathbf{p}}^k\|_2}{\|\mathbf{p}\|_2} \tag{8.9}$$

to compare the numerical solution with the true solution.

For all methods, we terminate the iterative process if the relative residual has reached the stopping criterion or if the maximum number of iterations has been reached.

9

Comparison between Two-Level Preconditioners using Numerical Experiments

In this Chapter, several numerical experiments will be performed for the test cases defined in Section 8. The chosen initial vector $\overline{\mathbf{p}}$ is a random vector for all test cases. Recall that the system to solve is

$$\mathbf{Ap} = \mathbf{q}, \tag{9.1}$$

where $\mathbf{p}$ is the pressure and $\mathbf{q}$ is the source vector. Two-Level preconditioners will be applied to this system to show the differences between the methods and to validate the theory presented in Section 7.

9.1. Test Case 1: Laplace Equation

For this Test Case, the system is defined as a matrix $\mathbf{A} \in \mathbb{R}^{900\times900}$ and the source vector $\mathbf{q} \in \mathbb{R}^{900}$ is a random vector. The approximated solution for Test Case 1 can be found in Figure 9.1.

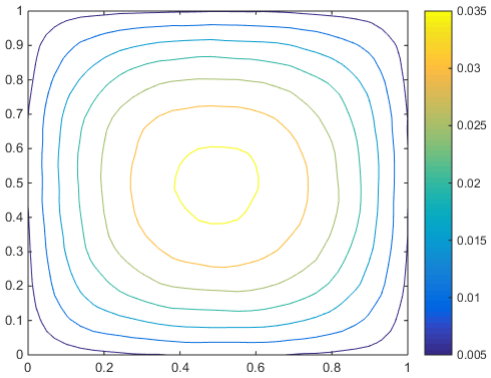


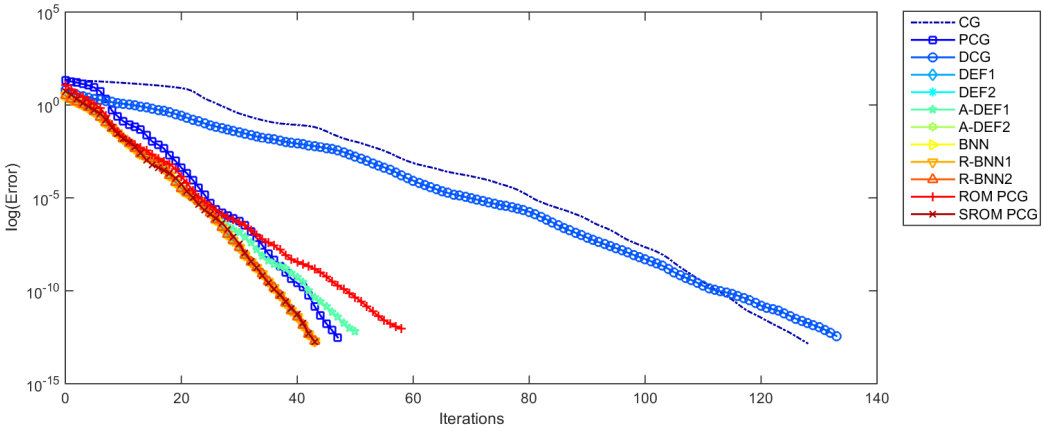
Figure 9.1: True solution of TC1: Laplace Equation

The exact solution is obtained using direct methods and will be denoted as the true solution $\mathbf{p}$, which will be used as reference for the solutions obtained by the different

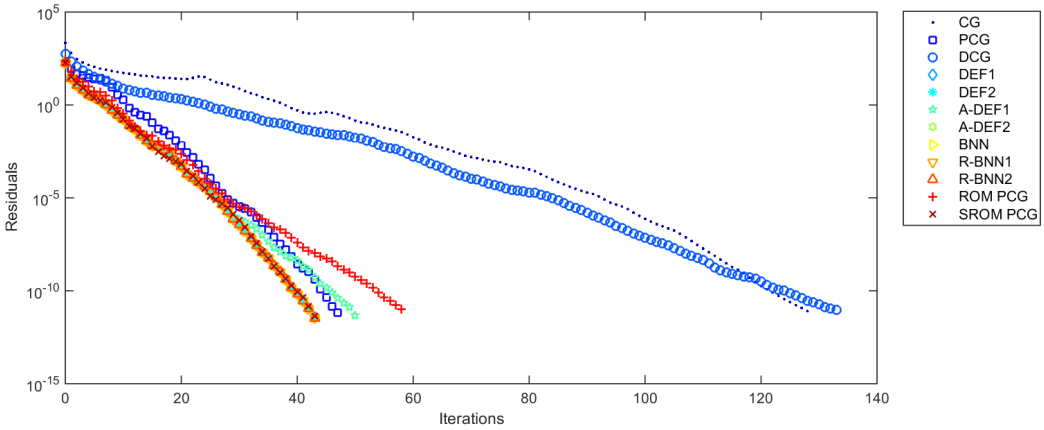
numerical methods. The system $\mathbf{Ap} = \mathbf{q}$ will be solved with the Two-Level PCG method. For this method, two choices of deflation vectors will be used: subdomain vectors and eigenvectors of the system $\mathbf{M}^{-1}\mathbf{A}$. The traditional preconditioner matrix $\mathbf{M}$ that is being used is the incomplete Cholesky with zero fill-in.

9.1.1. Subdomain as Deflation Vectors

The used stopping criterion is $\epsilon = 10^{-12}$ and the maximum number of iterations is 200. The domain is divided into 4 subdomains. Each subdomain vector corresponds with one subdomain. The results are presented in Figure 9.2 and the overview table can be found in Table 9.1.



(a) Relative error versus number of iterations



(b) Relative Residuals versus number of iterations

Figure 9.2: Visualization of the results for Test Case 1 using 4 subdomain vectors.

Table 9.1: Overview Table: Test Case 1 using 4 subdomain vectors as deflation vectors.

Method	Error	Residuals	# Iterations	Flops Iteration	Time s
CG	1.51e-13	8.03e-12	128	$19n$	0.00980
PCG	3.06e-13	6.82e-12	47	$30n$	0.00620
DCG	3.66e-13	9.51e-12	133	$35n$	0.01117
DEF1	2.01e-13	3.75e-12	43	$46n$	0.00598
DEF2	1.89e-13	3.47e-12	43	$46n$	0.00726
A-DEF1	6.67e-13	4.66e-12	50	$54n$	0.00723
A-DEF2	1.94e-13	3.47e-12	43	$62n$	0.00600
BNN	2.01e-13	3.47e-12	43	$78n$	0.00618
R-BNN1	1.95e-13	3.48e-12	43	$62n$	0.00593
R-BNN2	1.91e-13	3.45e-12	43	$46n$	0.00546
ROM	9.51e-13	9.82e-12	58	$56n$	0.00805
SROM	1.83e-13	4.43e-12	43	$64n$	0.00794

The number of iterations required to reach the stopping criterion is 128 for CG and 133 for DCG. Because of the high accuracy, $\epsilon = 10^{-12}$, DCG needs more iterations than CG. In general, as can be seen from both Figure 9.2 and Table 9.1, the number of iterations used for the deflation methods does not differ much from the PCG method. The number of iterations needed for when using the Two-Level preconditioners is at least 43 iterations (DEF1, DEF2, A-DEF2, BNN, R-BNN1, R-BNN2 and SROM) and the maximum number is 58 iterations (ROM). Furthermore, we increase the number of subdomain to show that the number of subdomain vectors does not lead to a large reduction in iterations. The result can be found in Table9.2.

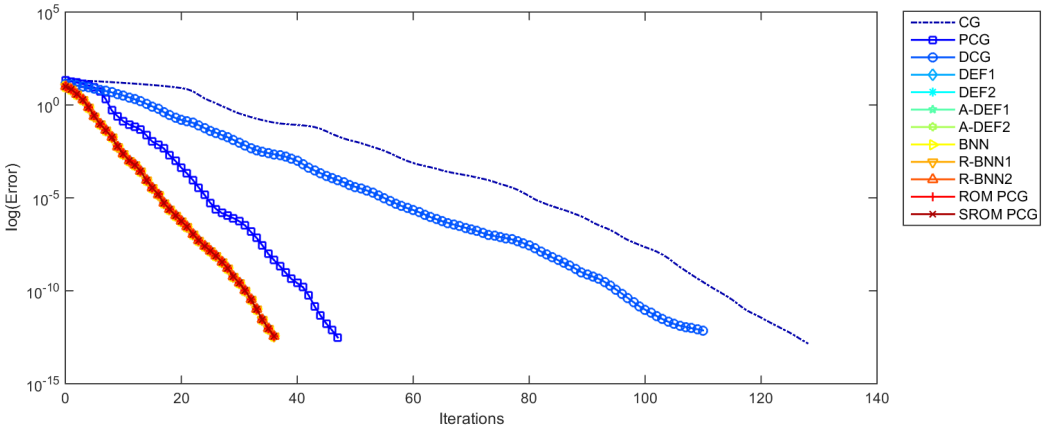
Table 9.2: Representation of the difference in flop counts using different numbers of subdomain vectors (m).

Method	$m = 4$			$m = 100$
	Flops		#Iteration	# Iteration
	Initial	Iteration		
CG	$12n$	$19n$	128	128
PCG	$28n$	$30n$	47	47
DCG	$184n$	$35n$	133	88
DEF1	$200n$	$46n$	43	36
DEF2	$200n$	$46n$	43	36
A-DEF1	$177n$	$54n$	50	43
A-DEF2	$217n$	$62n$	43	36
BNN	$201n$	$78n$	43	36
R-BNN1	$217n$	$62n$	43	36
R-BNN2	$201n$	$46n$	43	36
ROM	$151n$	$56n$	58	48
SROM	$175n$	$64n$	43	37

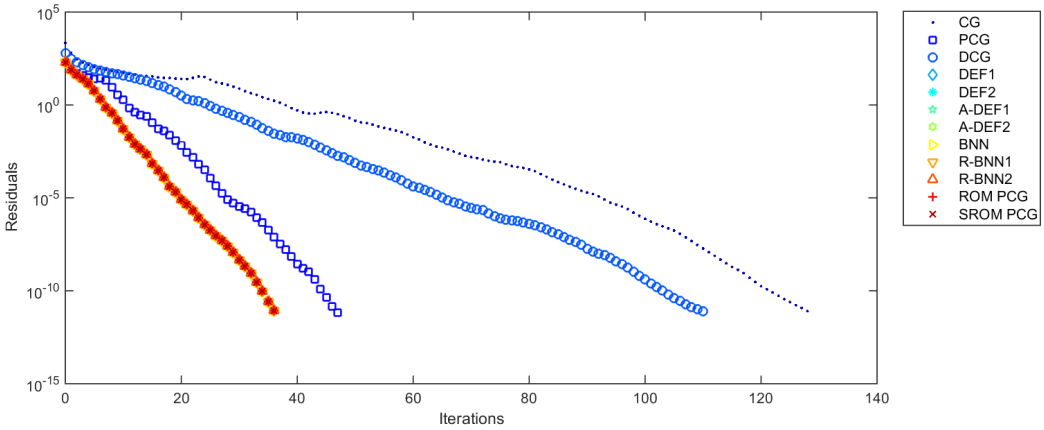
As can be seen in Table 9.2 that using 100 subdomain vectors is that the flops per iteration and initial flop will increase [25]. Therefore, adding more subdomain vectors will reduce the number of iterations, but also increase the number of flops.

9.1.2. Eigenvectors as Deflation Vectors

We repeat the process of the previous section using eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ as deflation vectors. The eigenvectors are chosen such that they corresponds to the smallest eigenvalues. The result can be found in Figure 9.3 and Table 9.3.



(a) Relative error versus number of iterations



(b) Relative Residuals versus number of iterations

Figure 9.3: Visualization of the results for Test Case 1 using 4 eigenvectors as deflation vectors.

Table 9.3: Overview Table: Test Case 1 using 4 eigenvectors as deflation vectors.

Method	Error	Residuals	# Iterations	Flops Iteration	Time s
CG	1.51e-13	8.03e-12	128	$19n$	0.00907
PCG	3.06e-13	6.82e-12	47	$30n$	0.00586
DCG	7.35e-13	8.18e-12	110	$35n$	0.00905
DEF1	3.56e-13	8.54e-12	36	$46n$	0.00497
DEF2	3.56e-13	8.57e-12	36	$46n$	0.00446
A-DEF1	3.61e-13	8.53e-12	36	$54n$	0.00570
A-DEF2	3.59e-13	8.56e-12	36	$62n$	0.00477
BNN	3.59e-13	8.52e-12	36	$78n$	0.00523
R-BNN1	3.56e-13	8.55e-12	36	$62n$	0.00509
R-BNN2	3.56e-13	8.57e-12	36	$46n$	0.00454
ROM	3.57e-13	8.52e-12	36	$56n$	0.00687
SROM	3.59e-13	8.54e-12	36	$64n$	0.00702

The advantage of using eigenvectors compared to using subdomain vectors is that the costs per iterations stays the same, but the number of iterations is lower for eigenvectors. The computation times are similar, because the problem is not large. We increase the number of eigenvectors to 100 and the result can be found in Table 9.4.

Table 9.4: Representation of the difference of flop counts using different number of eigenvectors

Method	$m = 4$			$m = 100$		
	Flops		# Iteration	Flops		# Iteration
	Initial	Iteration		Initial	Iteration	
CG	$12n$	$19n$	128	$12n$	$19n$	128
PCG	$28n$	$30n$	47	$28n$	$30n$	47
DCG	$184n$	$35n$	110	$61912n$	$419n$	42
DEF1	$200n$	$46n$	36	$61928n$	$430n$	14
DEF2	$200n$	$46n$	36	$61928n$	$430n$	14
A-DEF1	$177n$	$54n$	36	$61329n$	$630n$	14
A-DEF2	$217n$	$62n$	36	$62329n$	$830n$	14
BNN	$201n$	$78n$	36	$61929n$	$1230n$	14
R-BNN1	$217n$	$62n$	36	$62329n$	$830n$	14
R-BNN2	$201n$	$46n$	36	$61929n$	$430n$	14
ROM	$151n$	$56n$	36	$41239n$	$440n$	14
SROM	$175n$	$64n$	36	$22831n$	$832n$	14

We note that in Table 9.4, that using 100 eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$, is that the flops per iteration and initial flop costs are much higher. For example, BNN needs almost 16 times more flops per iteration and 310 times more flops to construct the initial matrices, while the number of iterations saved is 10. There is a noticeable difference for the DCG method. Still, the number of iterations is almost two times the number of the PCG method. Therefore, adding more eigenvectors as deflation vectors will reduce the number of iterations, but also increase the the flop costs.

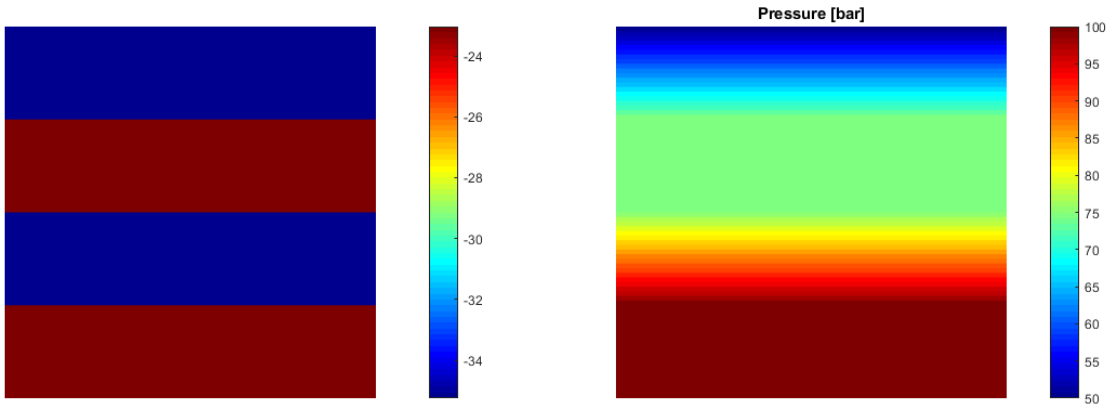
Remark

In Test Case 1, we used subdomain vectors and eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ as deflation vectors. If we only compare the Two-Level preconditioners. The maximum number of iterations needed to achieve convergence using subdomain vectors is 58 for the ROM method. The minimum number of iterations needed is 43 for DEF1, A-DEF2, BNN, R-BNN1, R-BNN2, and SROM. If eigenvectors are used, the maximum number of iterations is 36 for all methods. The eigenvectors are the best choice as deflation vectors for Test Case 1, assuming the eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ are already obtained. All methods needs the same number of iterations, the preferred method would be the one who needs the least number of flops. It follows that DEF1, DEF2, R-BNN2 and ROM would fit this criterion needing at most 56n flops for 4 deflation vectors and 440n flops for 100 deflation vectors per iteration.

For Test Case 2 and Test Case 3, we neglect the CG method and the DCG method in the numerical experiments. Only the PCG method is used as guideline while comparing the deflation techniques.

9.2. Test Case 2: Layered Problem

Test Case 2 is defined as a layered problem consisting of 4 layers with 2 different permeability values, where the order of the constract between them is $\mathcal{O}(10^6)$. The rock permeability field can be found in Figure 9.4a.



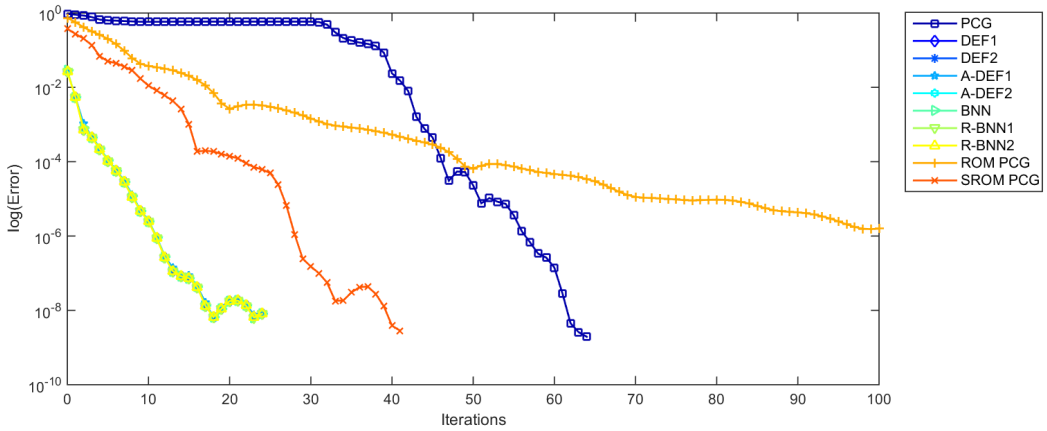
(a) Porous Media with 2 different layers (b) Solution of the pressure

Figure 9.4: Test Case 2: Layered problem

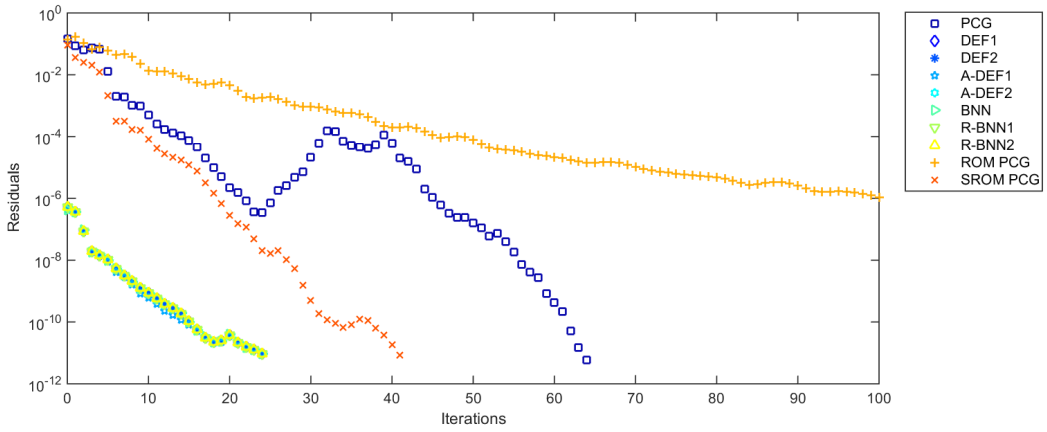
The model of Test case 2 is modelled using the software package MATLAB Reservoir Simulation Toolbox (MRST) found in [13]. The mesh of the problem is chosen where the x-axis and y-axis is equally divided in 40 . Therefore, the size of $\mathbf{A}$ is 1600×1600. The two different layers have permeabilities of 0.510^{-3} Darcy and 100 Darcy respectively. The condition number of $\mathbf{A}$ is $\kappa(\mathbf{A}) \approx \mathcal{O}(10^8)$. The boundary conditions are pressures on the boundary given as $p_{bc1} = 100$ bars and $p_{bc2} = 50$ bars. The approximated solution of the pressure field can be found in Figure 9.4b.

9.2.1. Subdomain vectors as deflation vectors

The porous media contains 4 layers and it is a natural choice to use 4 subdomain vectors. The result can be found in Figure 9.5 and Table 9.5.



(a) Relative error versus number of iterations



(b) Relative residuals versus number of iterations

Figure 9.5: Visualization of the results for Test Case 2 using 4 subdomain vectors as deflation vectors.

Table 9.5: Overview Table: Test case 2 using 4 subdomain vectors as deflation vectors.

Method	Error	Residuals	# Iterations	Flops Iteration	Time s
PCG	1.94e-09	5.84e-12	64	$30n$	0.01175
DEF1	8.12e-09	9.17e-12	24	$46n$	0.00462
DEF2	8.12e-09	9.17e-12	24	$46n$	0.00421
A-DEF1	8.13e-09	9.59e-12	24	$54n$	0.00505
A-DEF2	8.10e-09	9.17e-12	24	$62n$	0.00447
BNN	8.06e-09	9.32e-12	24	$78n$	0.00536
R-BNN1	8.11e-09	9.17e-12	24	$62n$	0.00486
R-BNN2	8.12e-09	9.17e-12	24	$46n$	0.00456
ROM	1.58e-06	1.10e-06	NC	$56n$	0.02070
SROM	2.84e-09	8.66e-12	41	$64n$	0.00862

9.2.2. Complexity

To test the dependence of the computation time on the size of the matrix, we enlarge the size by a factor of 4. In the previous case, the domain contains 4 distinct layers.

The choice for the deflation vectors using 4 subdomain vectors remains. The results are given in Table 9.6.

Table 9.6: Overview of the amount of flops and the needed computational time using 4 subdomain vectors. The size of the matrices are $\mathbf{A} \in \mathbb{R}^{1600\times1600}$ and $\mathbf{A} \in \mathbb{R}^{6400\times6400}$

Methods	n	Flops		Iterations	Time (s)		
		Initial	Iteration		Initial	Iteration	Total
PCG	1600	$28n$	$30n$	64	0.00013	0.00016	0.01045
	6400			NC	0.00141	0.00077	0.07860
DEF1	1600	$200n$	$46n$	24	0.00040	0.00017	0.00445
	6400			32	0.00335	0.00074	0.02693
DEF2	1600	$200n$	$46n$	24	0.00074	0.00015	0.00441
	6400			32	0.00234	0.00070	0.02473
A-DEF1	1600	$177n$	$54n$	24	0.00123	0.00015	0.00480
	6400			32	0.00592	0.00069	0.02784
A-DEF2	1600	$217n$	$62n$	24	0.00047	0.00017	0.00448
	6400			32	0.00298	0.00074	0.02653
BNN	1600	$201n$	$78n$	24	0.00045	0.00018	0.00465
	6400			32	0.00541	0.00083	0.03205
R-BNN1	1600	$217n$	$62n$	24	0.00039	0.00017	0.00444
	6400			32	0.00217	0.00075	0.02608
R-BNN2	1600	$201n$	$46n$	24	0.00037	0.00016	0.00426
	6400			32	0.00109	0.00062	0.02085
ROM	1600	$151n$	$56n$	NC	0.00122	0.00016	0.01754
	6400			NC	0.00582	0.00084	0.08962
SROM	1600	$175n$	$64n$	41	0.00130	0.00016	0.00789
	6400			75	0.00623	0.00076	0.06317

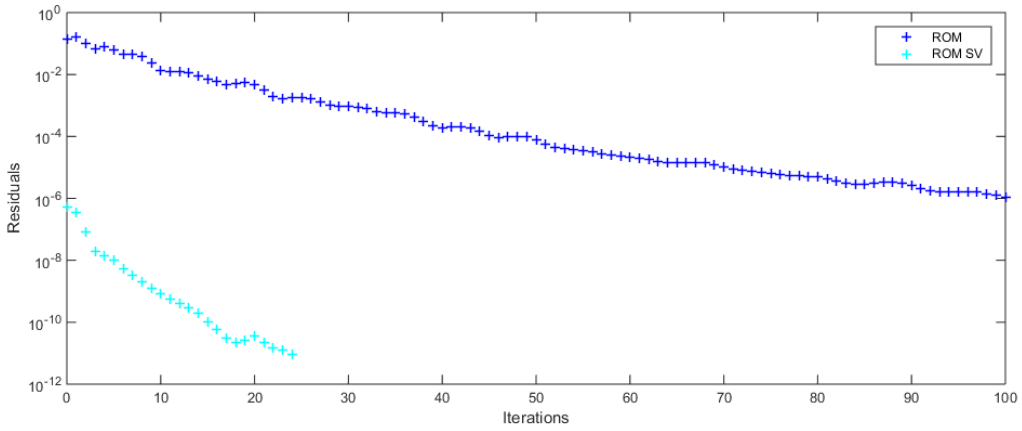
The studied matrix is enlarged by 4 and the number of iterations has increased by a factor of 1.5-2 for al methods that has reach convergence. For most methods, the amount of time is increased by a factor of 6. We observe two exceptions; The ROM method and the SROM method. The ROM method did not converge in both cases therefore the time only increased by 4. For the SROM method, the number of iterations has increased by a factor of 2. It follows that the time is increased by a factor of 8.

9.2.3. Special Starting Vector

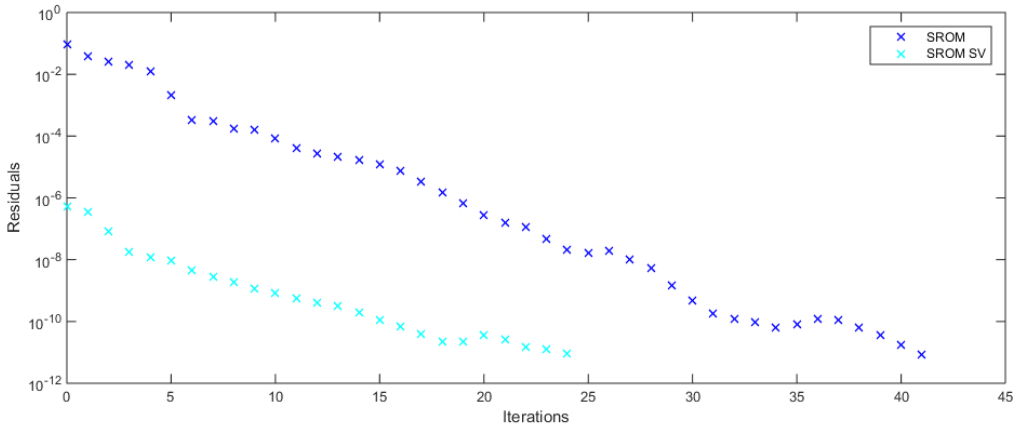
The chosen initial condition for Test Case 2 is a random vector. For some methods, the starting vector is the same as the initial condition or has been changed into the special starting vector using the formula

$$\mathbf{p}^0 = \mathbf{Q}\mathbf{q} + \mathbf{P}^T\overline{\mathbf{p}}, \tag{9.2}$$

where $\overline{\mathbf{p}}$ is the initial vector. This special starting vector will ensure convergence for all methods. In practice, the test cases are time dependent and the previous time-step is used as initial condition, see Section 9.3.4. If we use a random vector as initial condition, it could happen that some methods performs worse than PCG. To illustrate the difference, we use Test Case 2 with 4 subdomain vectors as deflation vectors. This will be applied to the ROM method and the SROM method. The differences are illustrated in Figure 9.6. We will denote the method with SV if the method uses the special starting vector.



(a) Relative Residuals of the ROM method



(b) Relative Residuals of the SROM method

Figure 9.6: The differences using different starting vectors applied to the ROM method and the SROM method

Table 9.7: Comparison of the ROM and SROM method using different starting vectors.

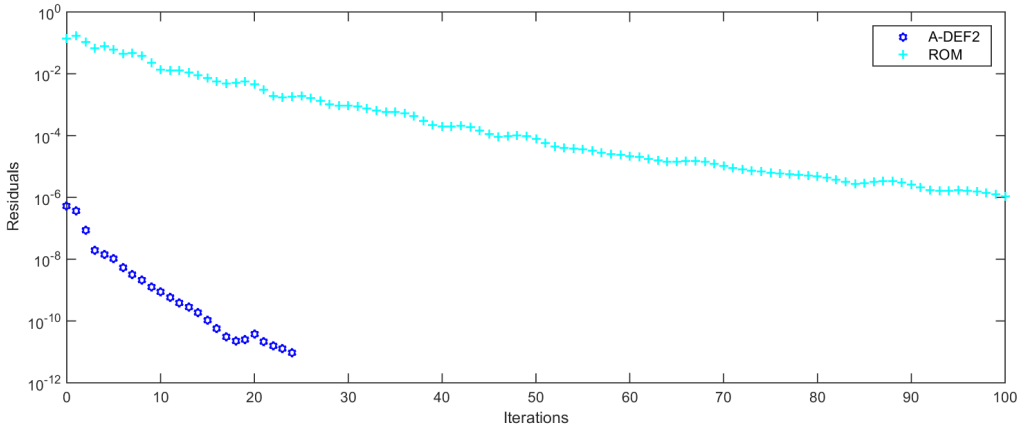
Method	Error	Residuals	Flops		# Iterations
			Initial	Iteration	
ROM	1.58e-06	1.10e-06	$151n$	$56n$	NC
ROM SV	8.12e-09	9.17e-12	$211n$	$56n$	24
SROM	2.84e-09	8.66e-12	$175n$	$64n$	41
SROM SV	7.45e-09	9.05e-12	$299n$	$64n$	24

The error, residual, flops and number of iterations are presented in Table 9.7. There is a noticeable difference between using a special starting for both the ROM method and the SROM method. If the special starting vector is used, the number of iterations decreases and with a small increases of the initial flops. We observe that using the special starting vectors guarantees convergence of the ROM method.

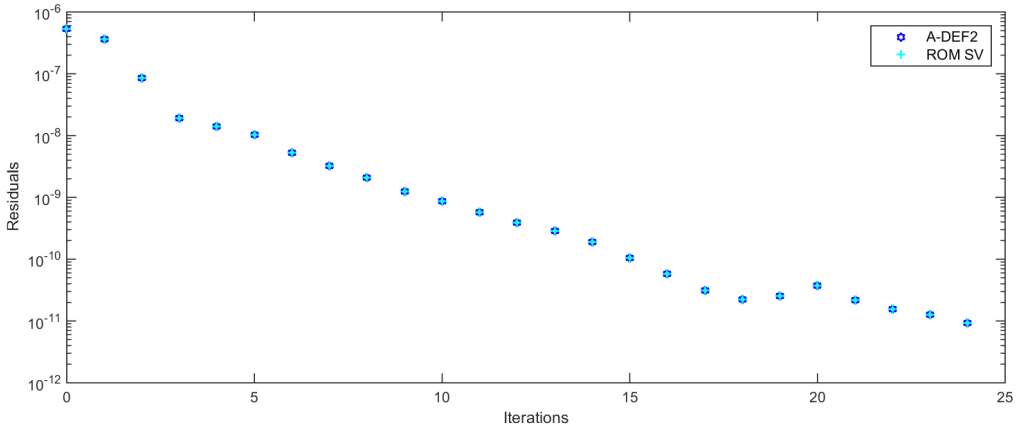
Comparison of the A-DEF2 method and the ROM method

In Section 7.3.1, we proved that the Two-Level preconditioner of A-DEF2 and ROM are the same in Lemma 7.3.2. In this experiments, we compare the A-DEF2 method with the ROM method using 4 subdomain vectors as deflation vectors. The A-DEF2

method always uses the special starting vector. We only change the starting vector of the ROM method and compare that with the A-DEF2. The result for ROM using different starting vector is given in Figure 9.7a and using special starting vector is given in Figure 9.7b.



(a) ROM uses initial condition as starting vector.



(b) ROM uses special starting vector.

Figure 9.7: Relative error versus number of iterations. Comparison of A-DEF2 and ROM.

Table 9.8: Comparison of A-DEF2 and ROM method using different starting vector

Without special starting vector

Method	Error	Residuals	Flops		# Iterations
			Initial	Iteration	
A-DEF2	8.10e-09	9.17e-12	$217n$	$62n$	24
ROM	1.58e-06	1.10e-06	$151n$	$56n$	NC

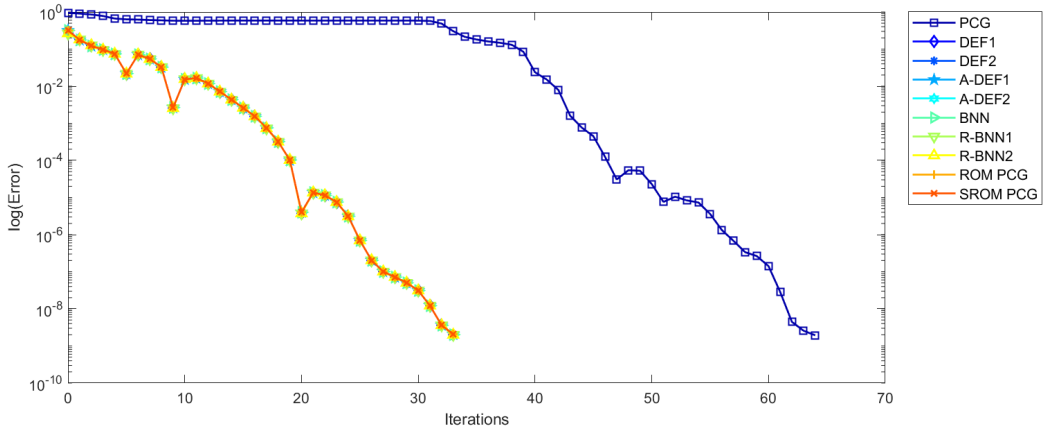
Using special starting vector

Method	Error	Residuals	Flops		# Iterations
			Initial	Iteration	
A-DEF2	8.10e-09	9.17e-12	$217n$	$62n$	24
ROM	8.10e-09	9.17e-12	$211n$	$56n$	24

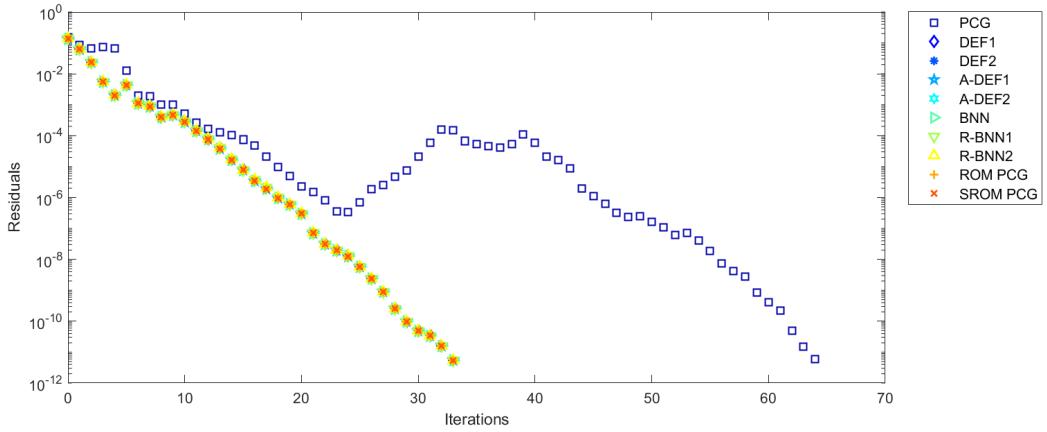
First, convergence is achieved for ROM when using the special starting vector. It can be seen from Table 9.8 that the result for A-DEF2 and ROM are the same. This verifies what we have proven in Section 7.3.1.

9.2.4. Eigenvectors as Deflation Vectors

As a second choice for the deflation vectors are eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$. We study a problem of size $\mathbf{A} \in \mathbb{R}^{1600\times1600}$. For this experiment we use 4 eigenvectors will be used as deflation vectors. The termination conditions are: $\epsilon = 10^{-11}$ and the maximum number of iterations is 100. The results are presented in Figure 9.8 and Table 9.9.



(a) Relative error versus number of iterations



(b) Relative Residuals versus number of iterations

Figure 9.8: Visualization of the results for Test Case 2 using 4 eigenvectors as deflation vectors.

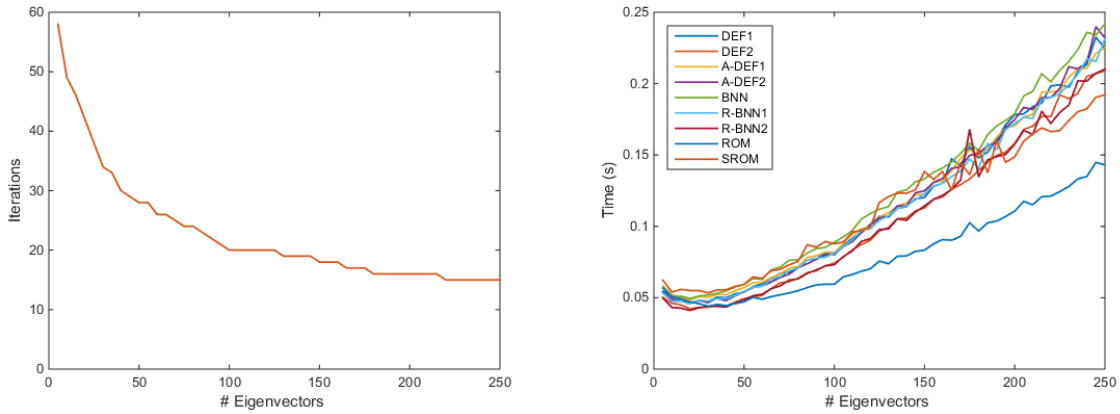
Table 9.9: Overview Table: Test Case 2 using 4 eigenvectors as deflation vectors.

Method	Error	Residuals	# Iterations	Flops Iteration	Time s
PCG	1.94e-09	5.84e-12	64	$30n$	0.01305
DEF1	2.00e-09	5.44e-12	33	$46n$	0.01251
DEF2	2.00e-09	5.44e-12	33	$46n$	0.01138
A-DEF1	2.06e-09	5.47e-12	33	$54n$	0.00970
A-DEF2	2.07e-09	5.44e-12	33	$62n$	0.00930
BNN	2.02e-09	5.47e-12	33	$78n$	0.00956
R-BNN1	2.00e-09	5.44e-12	33	$62n$	0.00957
R-BNN2	2.00e-09	5.44e-12	33	$46n$	0.00598
ROM	2.07e-09	5.47e-12	33	$56n$	0.00937
SROM	2.04e-09	5.47e-12	33	$64n$	0.01007

We observe that the number of iterations are the same for all methods. Using eigenvectors needed 33 iterations while using subdomain vectors needed 24 vectors for all method except ROM and SROM. This could happen due to the huge contrast between the eigenvalues.

Different number of eigenvectors

For this experiments, we increase the size of the matrix to $\mathbf{A} \in \mathbb{R}^{6400\times6400}$ and obtain the eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$. We vary the number of eigenvectors from 5 to 250. The termination condition remains the same: the stopping criterion is $\epsilon = 10^{-12}$ and the maximum number of iterations is 100. The result can be found in Figure 9.9.



(a) Number of eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$ versus Number of iterations

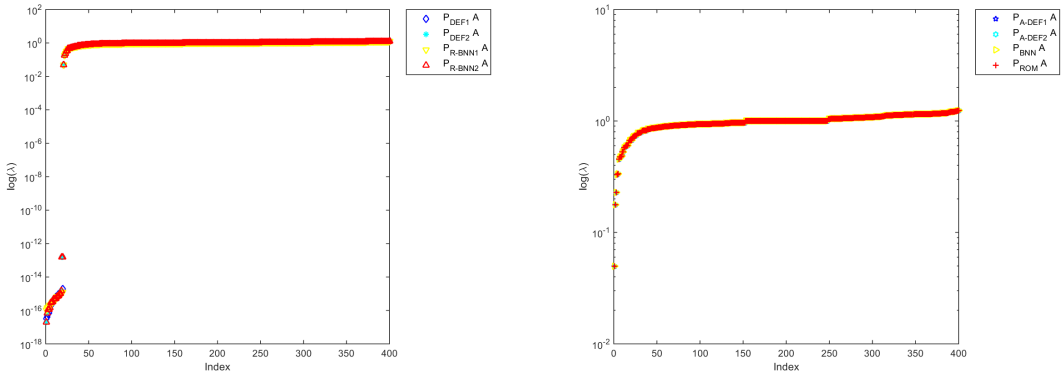
(b) Number of eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$ versus time (s)

Figure 9.9: Visualization of the results for Test Case 2 using different numbers of eigenvectors.

It follows from Figure 9.9a that if the number of eigenvectors are increased, the number of iterations decreases. However, increasing eigenvectors also increase the computation time and initial flops. The method that needs the most time is the BNN method and the one that needs the less is the DEF1 method. This can be found in Figure 9.9b and this result is expected from the within computational complexity in Chapter 7.

9.2.5. Spectra Analysis

In this section, we verify the Lemma's defined in Section 7. The parameters for this numerical experiments are: the size of the problem $\mathbf{A} \in \mathbb{R}^{400\times400}$, as traditional preconditioner $\mathbf{M}$ we choose the incomplete Cholesky with zero fill-in and we use 20 subdomain vectors as deflation vectors. The spectrum can be found in Figure 9.10.

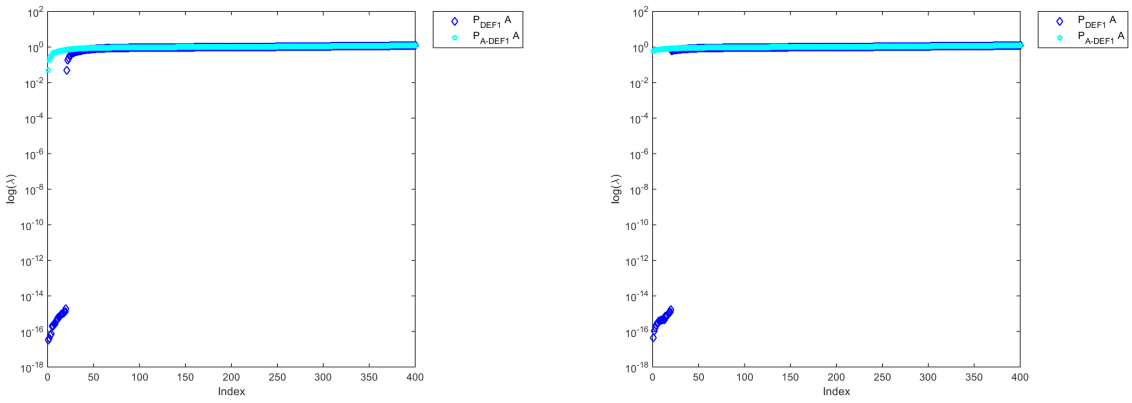


(a) The spectrum of DEF1, DEF2, R-BNN1, R-BNN2.

(b) The spectrum of Class A-DEF1, A-DEF2, BNN, ROM.

Figure 9.10: Visualization of the spectrum of different methods using 20 subdomain vectors as deflation vectors.

This validates Lemma 7.3.3 and Lemma 7.3.4. The methods can be sorted into two classes. The methods in one class sets the eigenvalues equal to zero and the methods of the other class sets the eigenvalues equal to 1. To investigate the spectrum, it is enough to choose a representative, DEF1 for the class that sets the eigenvalue equal to zero and A-DEF1 for the class that sets the eigenvalue equal to 1, and compare them. This result can be found in Figure 9.11.



(a) Comparison using 20 subdomain vectors as deflation vectors.

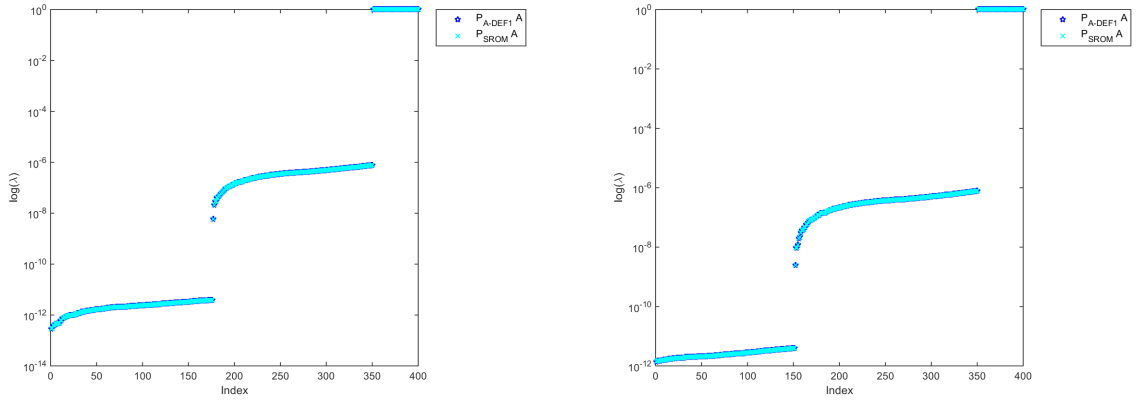
(b) Comparison using 20 eigenvectors of preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ as deflation vectors.

Figure 9.11: Visualization of the comparison between the two different classes using 20 eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ and subdomain vectors as deflation vectors.

We observe that some eigenvalues are set to 0 for one class and the eigenvalues are set to 1 for the other class. This validates Lemma 7.3.5.

Spectra Analysis of the SROM method

We suspect that the SROM method applied to the matrix $\mathbf{A}$ has the same spectrum as BNN, A-DEF1, A-DEF2 and ROM. We will verify this with two different types of deflation vectors: eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ and subdomain vectors. The number of deflation vectors for this experiment is equal to 50. The size of the matrix $\mathbf{A} \in \mathbb{R}^{400\times400}$ remains the same. First, we use $\mathbf{M} = \mathbf{I}$ as traditional preconditioner. The result can be found in Figure 9.12

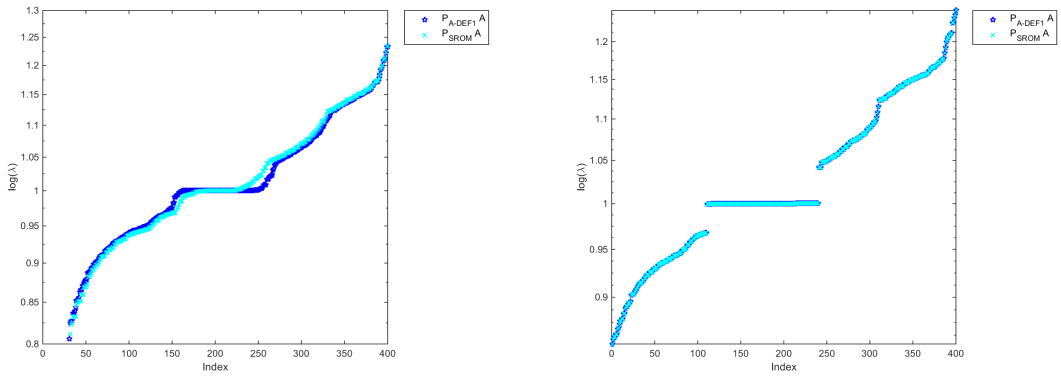


(a) Comparison using 50 subdomain vectors as deflation vectors.

(b) Comparison using 50 eigenvectors of preconditioned system $\mathbf{A}$ as deflation vectors.

Figure 9.12: Visualization of the spectrum of SROM applied to matrix $\mathbf{A}$ compared to A-DEF1 applied to the matrix $\mathbf{A}$. We use 50 eigenvectors of preconditioned system $\mathbf{A}$ and subdomain vectors as deflation vectors.

We repeat this using $\mathbf{M}$ to be incomplete Cholesky with zero fill-in as traditional preconditioner.



(a) Comparison using 50 subdomain vectors as deflation vectors.

(b) Comparison using 50 eigenvectors of preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ as deflation vectors.

Figure 9.13: Visualization of the spectrum of SROM applied to matrix $\mathbf{A}$ compared to A-DEF1 applied to the matrix $\mathbf{A}$. We use 50 eigenvectors of preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ and subdomain vectors as deflation vectors.

If we use $\mathbf{M} = \mathbf{I}$, it can be seen from Figure 9.12 that the result is what we would expect from theory. If we use $\mathbf{M}$ to be the incomplete Cholesky with zero fill-in. We observe from Figure 9.13a that using subdomain vectors, it does not align fully with the spectrum of A-DEF1. If eigenvectors of the preconditioned matrix is used, we

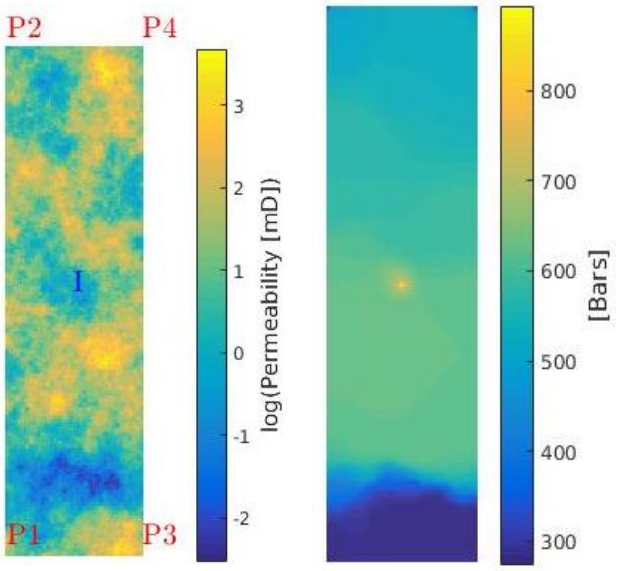
observe from Figure 9.13b that the result is as expected from theory.

Remark
For Test Case 2, it is logical to use 4 subdomain vectors because the porous media domain consists of 4 layers. The result showed that by using deflation methods decreases the number of iterations. The computation complexity increases linear with the size of the matrix. We have investigated the starting vector by using the initial vector and special starting vector. We have observed that the use of special starting vector improves the methods ROM and SROM. Therefore, we only use the special starting vector as starting vector. The advantages of using the special starting vector is that it ensure convergences for all methods. If the special starting vector is used, then the operators of A-DEF2 and ROM are the same.

For Test Case 2, eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ are used as deflation vectors. For all the methods, they need less iterations than using subdomain vectors. If the number of eigenvectors is increased, the computation time increases and the number of iterations hardly decreases. We compared the spectra of different methods applied to the matrix $\mathbf{A}$ to validate the theory. We observe that the theory and numerical experiments align.

9.3. Test Case 3: SPE10

In this section, we perform a two-phase flow simulation for the upper layer of SPE10 model, injecting water into the reservoir using an injection well I located the centre of the domain and producing oil through four production wells P1-P4, see Figure 9.14a. The permeability field of the full model is presented in Figure 9.14a and solution of the pressure in Figure 9.14b.



(a) Rock permeability of the upper layer.

(b) The pressure field of the upper layer.

Figure 9.14: Visualization of the results for Test Case 3 using 5 eigenvectors as deflation vectors.

We study the upper layer of the domain with 60×220 cells. It follows that the matrix

of the SPE10 model is large, $\mathbf{A} \in \mathbb{R}^{13200\times13200}$. The condition number of the original matrix is $\kappa(\mathbf{A}) \approx 10^8$. The termination criteria of this Test Case is chosen as follows: The stopping criterion is $\epsilon = 10^{-7}$ and the maximum number of iterations is 1500. We use eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ and POD basis vectors as deflation vectors. For all methods, the initial vector is a random vector and we use special starting vector as starting vector defined in Section 9.2.3 for ROM and SROM.

9.3.1. Eigenvectors as Deflation Vectors

It is known from Section 7 that the chosen eigenvectors set the corresponding eigenvalue equal to zero or one. The eigenvalues of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ can be found in Figure 9.15.

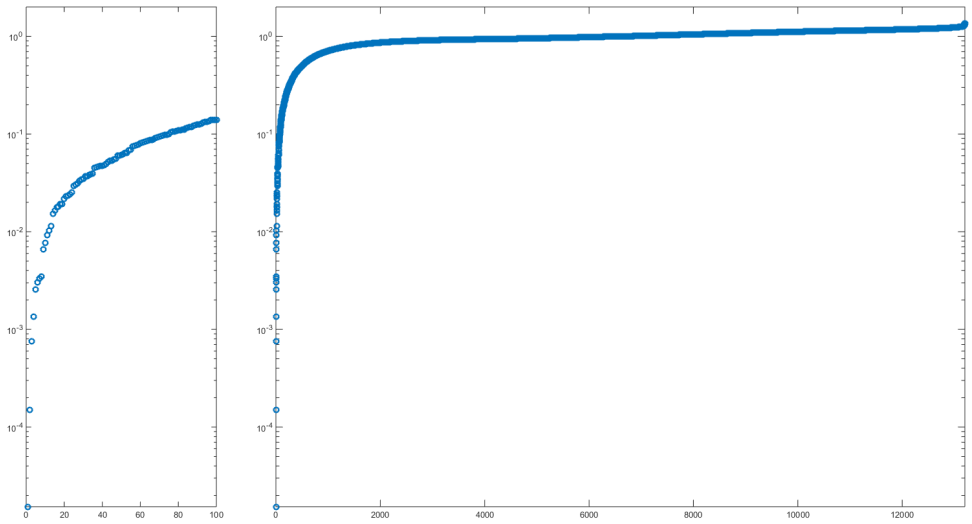
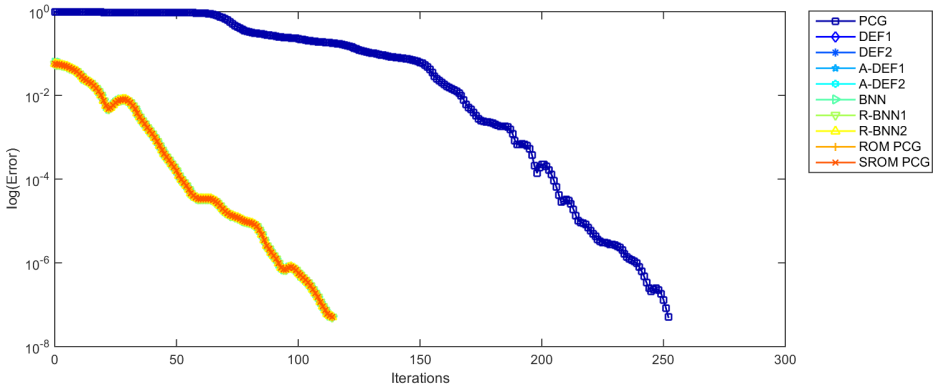
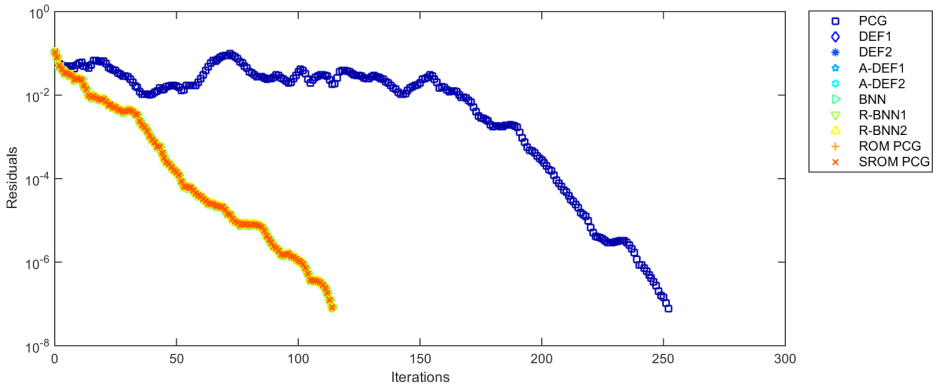


Figure 9.15: Eigenvalues of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$

It follows from Figure 9.15 that the extreme eigenvalues are the smallest eigenvalues. Therefore we use the eigenvectors corresponding to the smallest eigenvalues as deflation vectors. First we use 5 eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ and later we increase the number to 10. The result can be found in Figure 9.16.



(a) Relative error versus number of iterations



(b) Relative Residuals versus number of iterations

Figure 9.16: Visualization of the results for Test Case 3 using 5 eigenvectors as deflation vectors.

Table 9.10: Overview Table: Test Case 3 using 5 eigenvectors as deflation vectors.

Method	Error	Residuals	# Iterations	Flops Iteration	Time s
PCG	5.02e-08	7.68e-08	252	$30n$	0.31223
DEF1	5.16e-08	8.30e-08	114	$50n$	0.13650
DEF2	5.16e-08	8.30e-08	114	$50n$	0.12845
A-DEF1	5.19e-08	8.29e-08	114	$60n$	0.25791
A-DEF2	5.16e-08	8.30e-08	114	$70n$	0.13348
BNN	5.19e-08	8.29e-08	114	$90n$	0.15014
R-BNN1	5.16e-08	8.30e-08	114	$70n$	0.13692
R-BNN2	5.16e-08	8.30e-08	114	$50n$	0.13075
ROM	5.16e-08	8.30e-08	114	$60n$	0.25014
SROM	5.16e-08	8.30e-08	114	$72n$	0.26389

The PCG method needs 252 iterations to reach the stopping criterion. We observe using eigenvectors of the preconditioner system $\mathbf{M}^{-1}\mathbf{A}$ as deflation vectors, the number of iterations decreases. For all methods, 114 iterations are needed, which is half of the iterations needed compare to the PCG method. For some methods, around half of the computational time is needed compared to PCG, the methods are DEF1, DEF2, A-DEF2, BNN, R-BNN1 and R-BNN2. If we increase the number of eigenvectors to 10

and 20, less iterations are needed. The result can be found in Table 9.10.

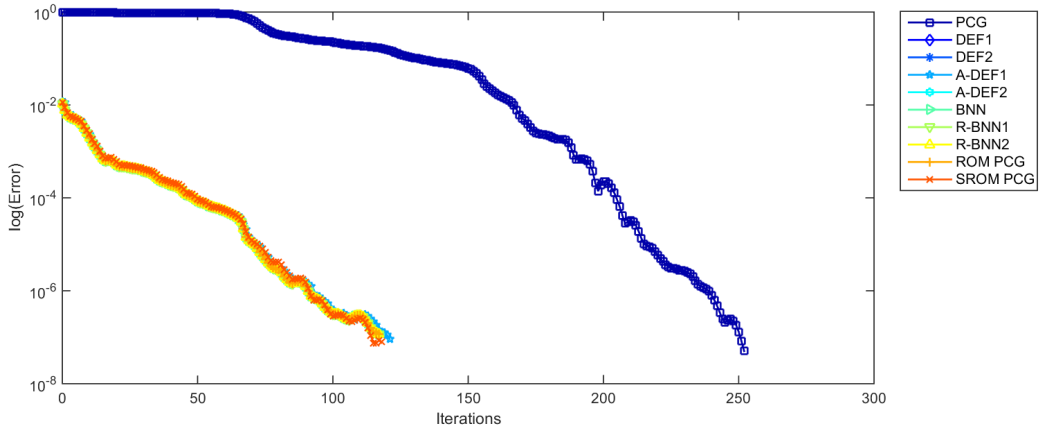
Table 9.11: Representation of the difference of flop counts using different number of eigenvectors

Method	m	Flops		# Iterations	Time		
		Initial	Iterations		Initial	Iteration	Total
PCG		$28n$	$30n$	252	0.00083	0.00124	0.31223
DEF1	5	$273n$	$50n$	114	0.00207	0.00118	0.13650
	10	$818n$	$70n$	76	0.00591	0.00136	0.10947
	20	$2808n$	$110n$	51	0.01062	0.00170	0.09747
DEF2	5	$273n$	$50n$	114	0.00218	0.00111	0.12845
	10	$818n$	$70n$	76	0.00650	0.00126	0.10239
	20	$2808n$	$110n$	51	0.01144	0.00141	0.08327
A-DEF1	5	$244n$	$60n$	114	0.12520	0.00116	0.25791
	10	$759n$	$90n$	76	0.13241	0.00139	0.23790
	20	$2689n$	$150n$	51	0.13303	0.00160	0.21467
A-DEF2	5	$294n$	$70n$	114	0.00218	0.00115	0.13348
	10	$859n$	$110n$	76	0.00549	0.00126	0.10123
	20	$2889n$	$190n$	51	0.01115	0.00277	0.15221
BNN	5	$274n$	$90n$	114	0.00214	0.00130	0.15014
	10	$819n$	$150n$	76	0.00575	0.00142	0.11392
	20	$2809n$	$270n$	51	0.00994	0.00186	0.10469
R-BNN1	5	$294n$	$70n$	114	0.00220	0.00118	0.13692
	10	$859n$	$110n$	76	0.00699	0.00135	0.10954
	20	$2889n$	$190n$	51	0.01039	0.00158	0.09098
R-BNN2	5	$274n$	$50n$	114	0.00209	0.00113	0.13075
	10	$819n$	$70n$	76	0.00595	0.00121	0.09797
	20	$2809n$	$110n$	51	0.01018	0.00140	0.08140
ROM	5	$284n$	$60n$	114	0.11541	0.00119	0.25014
	10	$829n$	$80n$	76	0.12029	0.00118	0.20792
	20	$2819n$	$120n$	51	0.12005	0.00140	0.18734
SROM	5	$414n$	$72n$	114	0.11956	0.00129	0.26389
	10	$1289n$	$112n$	76	0.12058	0.00134	0.21872
	20	$4539n$	$192n$	51	0.12570	0.00164	0.20243

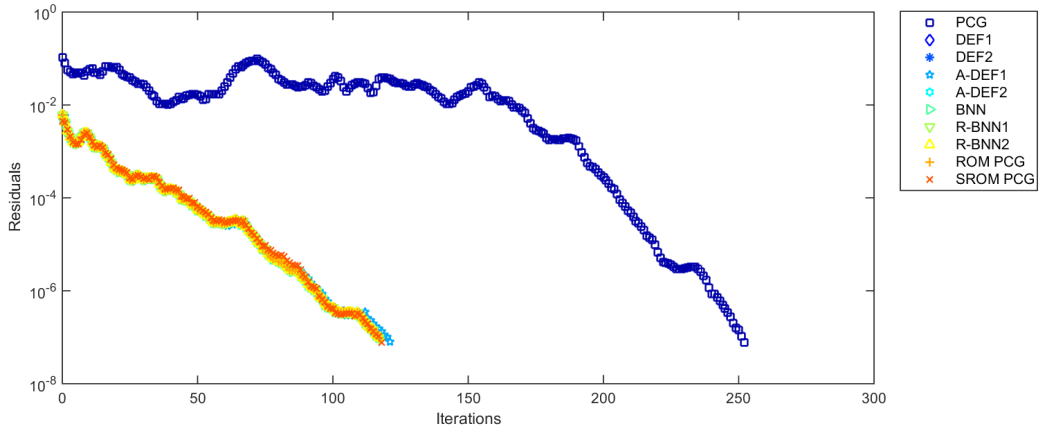
If the number of eigenvectors is increased, the computational complexity increases and the number of iterations decreases. The difference using 5 or 10 eigenvectors is more noticeable. The number of iterations dropped almost a third compared to a quarter from increasing 10 to 20 eigenvectors. In the end, the time to reach convergence decreases for all methods.

9.3.2. POD Basis vectors as Deflation Vectors

In this experiment, we use POD basis vectors as deflation vectors. The POD basis vectors are obtained as follows: The pressure at the injection well is maintained constant at $I = 1100$ bars. The pressure of the production wells P1-P4 is varied between 137.5 - 275 bars for every 2 time steps. The initial pressure is set to $P^0 = 500$ bars. Then, we run a simulation of 600 time steps, with every time step of 100 days, until the water reaches the production wells. See [5, 30, 35]. The solutions of this simulation are use to construct the POD basis vectors. We choose 5 and 10 POD basis vectors as deflation vectors in this experiment.



(a) Relative error versus number of iterations



(b) Relative Residuals versus number of iterations

Figure 9.17: Visualization of the results for Test Case 3 using 5 POD basis vectors.

Table 9.12: Overview Table: Test Case 3 using 5 POD basis vectors as deflation vectors.

Method	Error	Residuals	# Iterations	Flops Iteration	Time s
PCG	5.02e-08	7.68e-08	252	$30n$	0.30960
DEF1	1.15e-07	9.80e-08	117	$50n$	0.14375
DEF2	1.15e-07	9.80e-08	117	$50n$	0.15931
A-DEF1	9.14e-08	8.07e-08	121	$60n$	0.24939
A-DEF2	1.15e-07	9.80e-08	117	$70n$	0.14747
BNN	1.15e-07	9.75e-08	117	$90n$	0.15156
R-BNN1	1.15e-07	9.80e-08	117	$70n$	0.14885
R-BNN2	1.15e-07	9.80e-08	117	$50n$	0.12603
ROM	1.15e-07	9.80e-08	117	$60n$	0.24190
SROM	7.93e-08	7.74e-08	118	$72n$	0.25865

The number of POD vectors will be increased to 10.

Table 9.13: Representation of the difference in flop counts using different numbers of POD basis vectors.

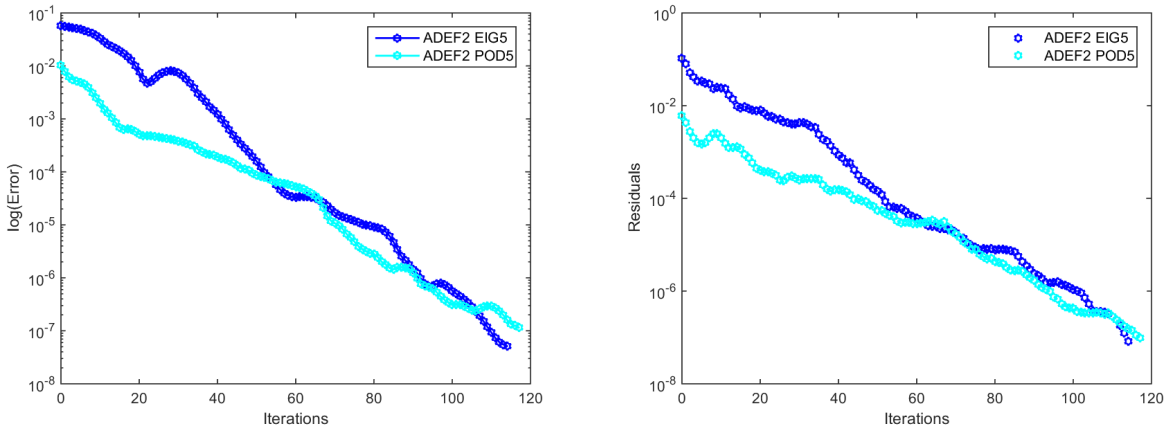
Method	m	Flops		# Iterations	Time		
		Initial	Iterations		Initial	Iteration	Total
PCG		$28n$	$30n$	252	0.00084	0.00123	0.30960
DEF1	5	$273n$	$50n$	117	0.00393	0.00120	0.14375
	10	$818n$	$70n$	101	0.00592	0.00126	0.13274
DEF2	5	$273n$	$50n$	117	0.00217	0.00134	0.15931
	10	$818n$	$70n$	101	0.00585	0.00120	0.12725
A-DEF1	5	$244n$	$60n$	121	0.11380	0.00112	0.24939
	10	$759n$	$90n$	104	0.11544	0.00122	0.24258
A-DEF2	5	$294n$	$70n$	117	0.00243	0.00124	0.14747
	10	$859n$	$110n$	101	0.00568	0.00139	0.14573
BNN	5	$274n$	$90n$	117	0.00309	0.00127	0.15156
	10	$819n$	$150n$	101	0.00874	0.00140	0.15021
R-BNN1	5	$294n$	$70n$	117	0.00251	0.00125	0.14885
	10	$859n$	$110n$	101	0.00590	0.00127	0.13412
R-BNN2	5	$274n$	$50n$	117	0.00212	0.00106	0.12603
	10	$819n$	$70n$	101	0.00551	0.00114	0.12092
ROM	5	$284n$	$60n$	117	0.11176	0.00112	0.24190
	10	$829n$	$80n$	101	0.11652	0.00121	0.23637
SROM	5	$414n$	$72n$	118	0.11538	0.00122	0.25865
	10	$1289n$	$112n$	102	0.11756	0.00138	0.25458

As can be seen from Figure 9.13, if the number of POD basis vectors is increased, the number of flops increases for the initialization and iteration. The computation time increases linear with the number of basis vectors. Note that fewer iterations are needed for each method. For the deflation methods (DEF1, DEF2, A-DEF1, A-DEF2), 20 iterations or less are needed before it reaches convergence and hence, the time remains fairly constant. We observe that the number of flops needed for the initialization doubles or triples, which matches with the fact that the times also increased with the similar constant. This also holds for the BNN methods (BNN, R-BNN1, R-BNN2). For ROM, the number of iterations is reduced a half and the time needed to reach

convergence also reduces a half.

9.3.3. Difference using Eigenvectors of the Preconditioned Matrix or POD basis vectors as Deflation Vectors

The noticeable difference after one iteration step is the order of the residuals. The order of the residual using eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ is 10^{-1} while using POD basis vectors the order of the residual is 10^{-2}. We illustrate the difference using the A-DEF2 method and 5 deflation vector each. The result can be found in Figure 9.18 and Table 9.14.



(a) Relative error versus number of iterations (b) Relative Residuals versus number of iterations

Figure 9.18: Visualization of the results for Test Case 3 using 5 eigenvectors and 5 POD vectors applied to A-DEF2 method.

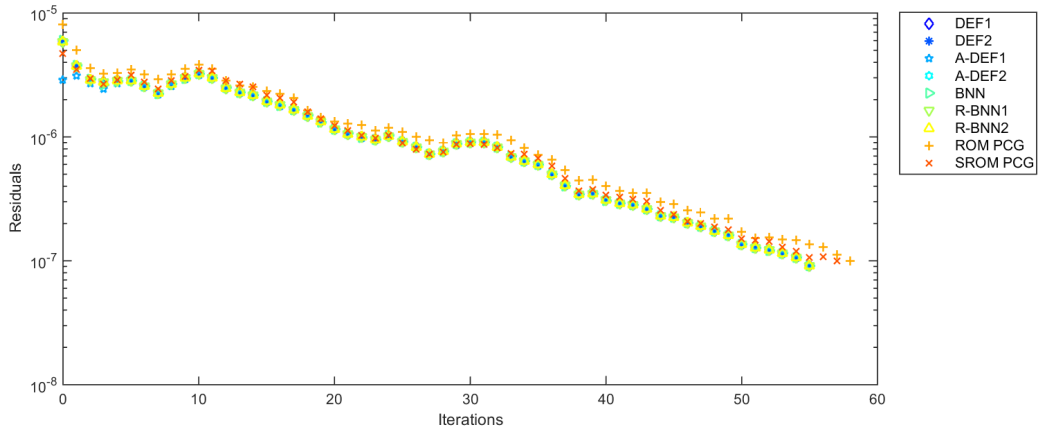
Table 9.14: Test Case 3: The difference using 5 eigenvectors and 5 POD vectors applied to A-DEF2 method.

Deflation vector	m	Error	Residuals	# Iterations	Flops Iteration	Time s
Eigenvectors	5	5.16e-08	8.30e-08	114	$70n$	0.14465
POD basis vectors	5	1.15e-07	9.80e-08	117	$70n$	0.14961

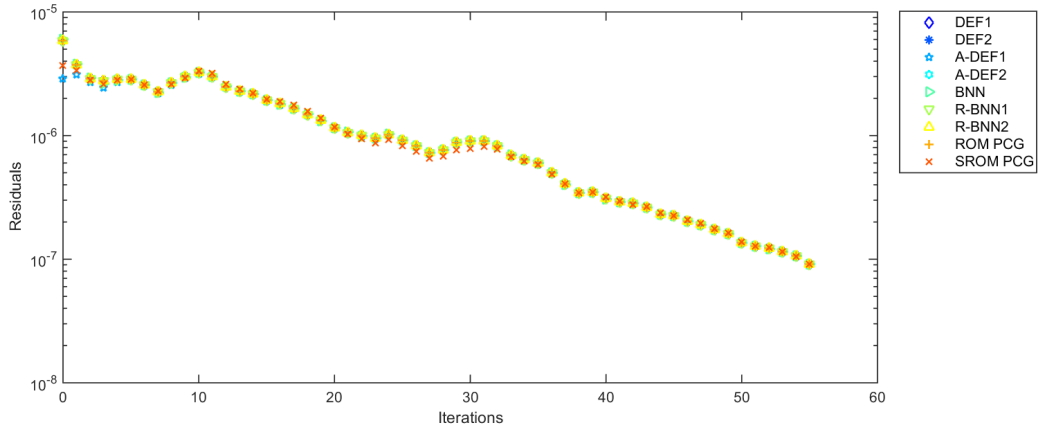
We have proved that using eigenvectors of the preconditioned matrix would be the best choice since we can manipulate the extreme eigenvalues to zero. The advantages of using POD basis vectors is the start solution in the beginning. The order of the residual starts at $\mathcal{O}(10^{-2})$, while using eigenvectors, it starts at $\mathcal{O}(10^{-1})$. It is due to the fact that POD basis vectors already contain information of the previous time-steps. Since using eigenvectors removes extreme eigenvalues in the methods, the convergence is steeper than using POD basis vectors. Note that 10^{-8} is a high accuracy. If the accuracy is lowered, using POD basis vectors will need less iterations than using eigenvectors.

9.3.4. Initial Vector

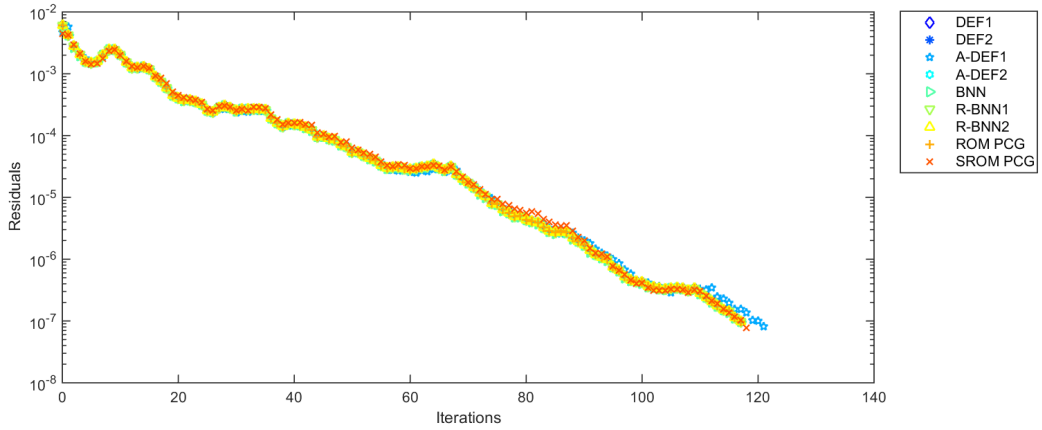
The initial vector used in the previous numerical experiments are random vectors. In practise, it is more realistic to reuse the solution of the previous time-step as initial vector. In this numerical experiment we use 5 POD basis vectors as deflation vectors and the same termination conditions. The result can be found in Figure 9.19.



(a) The initial vector is the previous time-step and the starting vector is the same as the initial vector.



(b) The initial vector is the previous time-step and the starting vector is the special starting vector.



(c) The initial vector is a random vector and the starting vector is the special starting vector.

Figure 9.19: Visualization of the relative residual for Test Case 3 using a random vector and previous time-step as initial vector.

There are a few advantages using the previous time-step compared using a random vector. The first observation is that the number of iterations is half of the number needed for the random vector. The other observation is that the order of the residual starts at $\mathcal{O}(10^{-5})$ instead of $\mathcal{O}(10^{-2})$. Finally, there is hardly any difference between

using a special starting vector or initial vector as starting vector. The number of iterations stays below 60.

Remark

For Test Case 3, we use eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ and POD basis vectors as deflation vectors. Comparing these two types of deflation vectors, eigenvectors are a better option because less iterations are needed and the residuals reach the stopping criterion faster. The advantages of using POD basis vectors is a smaller residual obtained after one iteration and they are relative cheap to obtain compared to eigenvectors. For both cases, increasing the number of deflation vectors will decrease the number of iterations, but the computation time remains the same. We have compared the performance of the methods using the previous time-step as initial condition compared to using a random vector. The residual has an order of $\mathcal{O}(10^{-5})$ for using the previous time-step as initial vector. The accuracy is higher than only using the special starting vector. The difference in magnitude is of order $\mathcal{O}(10^{3})$. This results hold for all studied methods and need less iterations to converge.

9.4. Concluding Remarks

In this section, we have tried several choices for deflation vectors. It can be concluded that the number of iterations of using the eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ is lower compared to using subdomain or POD basis vectors. In general, increasing the number of deflation vectors, decreases the number of iterations and increases the computation time.

From the studied methods, we observe that the BNN method needs the most number of flops per iteration and computation time. The method who needs the least number of flops and computation time is the DEF1 method. The other methods are DEF2, A-DEF1, R-BNN2 and ROM, they need around the same number of flops per iteration.

There are several ways to improve the methods. In this chapter we observe that using the right deflation vector can make a difference. In Test Case 2, using special starting vector instead of initial vector improves the methods ROM and SROM. In Test Case 3, we observe that using POD basis vectors wins a magnitude of $\mathcal{O}(10^{1})$ compared to using eigenvectors of the preconditioned system. For the same test case, we changed the initial condition to the previous time-step. After one iteration, the difference in magnitude is an order of $\mathcal{O}(10^{3})$ compared to using a random vector.

10

Conclusion

In this research, we focussed on the iterative solvers used for water simulation. We have presented a mathematical model for the one-phase and two-phase flow through porous media. Then, we linearized this model using the Newton-Raphson method to obtain a linear system. This results in a system of linear equations that is solved with Two-Level Preconditioned Conjugate Gradient method. For this method, we need to specify deflation vectors. The choices for deflation vectors considered in this report: subdomain vectors, eigenvectors and POD basis vectors. The Two-Level preconditioners we compared in this thesis report are: Deflation method (DEF1, DEF1), Adapted Deflation method (A-DEF1, A-DEF2), Reduced Order Model methods (ROM, SROM) and Balancing Neumann Neumann (BNN, R-BNN1, R-BNN2). These are compared theoretically and three test cases are used for the numerical experiments.

From the theoretical results, we have observed the computational complexity, memory storage and condition number of the various Two-Level PCG methods. The cheapest method per iteration is DEF1, while the most expensive one is BNN. For the memory storage, the cheapest method is DEF1, DEF2, R-BNN2, and ROM. The method that takes the most memory storage is BNN. We have also proven that A-DEF2 and ROM have the same Two-Level preconditioners. Then, we have compared the spectra of the methods. DEF1, DEF2, R-BNN1, and R-BNN2 have the same spectrum transforming some eigenvalues equal to 0. A-DEF1, A-DEF2, BNN, ROM, and possibly SROM have the same spectrum.

In the numerical experiments, we have tested the computational complexity and spectra using various deflation vectors to test the theory. We increased the number of deflation vectors and it makes a small difference in performance. We have observed that using the starting vectors improved the performance of ROM and SROM and using the previous time step as initial condition improved the performance for all methods by three orders of magnitude. If we use POD basis vectors instead of eigenvectors we can win an order of magnitude after one iteration. In general, the number of iterations for all methods is similar.

From the results we see that the performance and the memory storage of the methods are similar. However, the cheapest methods per iteration resulted DEF1, DEF2, R-BNN2, and ROM.

Future research

Deflation methods are fairly new compared to the Conjugate Gradient method and more research is needed for full understanding them. The recommendations for future research can be formulated as follows:

- Another criterion that this report did not consider is the accuracy of the methods. The stopping criterion was fixed for every test case. If we change the accuracy, it is possible that not every method will reach the convergence criterion.
- The only spectrum we did not prove for all deflation vectors is the SROM method. We have seen that the use of eigenvectors gives the result we expect from theory and numerical experiments. More research is needed to fully understand the spectrum of this method.
- The adapted deflation methods are an improvement to the deflation methods. In general, we could improve the methods by setting specific eigenvalues to the largest eigenvalue of the preconditioned system instead to 1. This could improve the performance of the deflation methods.
- In this report, we did not focus on the use of the POD basis as deflation vectors. More research on this could improve the methods.

A

Nomenclature

The list of notations defined in Section 3 is given in this Appendix.

Table A.1: Notation

Symbol	Quantity	SI Unit
ρ	Fluid density	kg/m^3
ϕ	Rock porosity	
q	Source term	
$\vec{v}$	Darcy's velocity	m/d
p	Pressure	Pa
K	Rock permeability	Darcy (D)
μ	Fluid viscosity	Pa
g	Gravity	m/s^2
d	reservoir depth	m
c_l	Liquid compressibility	Pa^{-1}
c_r	Rock compressibility	Pa^{-1}

B

Compressible Model

The compressible model is, unlike the incompressible model, time-dependent. For this model, it is more complex to derive the discretization to solve the problem numerically. This means that it would take more computational time to obtain a solution of the flow problem. Therefore, only the constant compressible model will be explained. More information can be found in [8, 13].

B.1. Constant Compressibility

Assume that the fluid density and rock porosity are constant compressible, i.e. $c_l, c_r \in \mathbb{R}$. Then, the total compressibility is also constant. Therefore, the fluid density and porosity are linearly dependent on the pressure. The initial condition for the pressure is defined as $p\big|_{t=0} = p_0$, without loss of generality let $p_0 = 0$. Now, the initial conditions for rock porosity and fluid density are:

$$\rho\Big|_{p=p_0} = \rho_0 \quad \text{and} \quad \varphi\Big|_{p=p_0} = \varphi_0. \tag{B.1}$$

Inserting the initial conditions in Equation (3.5) and Equation (3.4) gives:

$$\varphi = \varphi_0 e^{c_r p} \quad \text{and} \quad \rho = \rho_0 e^{c_l p}. \tag{B.2}$$

For small values of the fluid compressibility, the fluid density can be written as

$$\rho \approx \rho_0 (1 + c_l p) \tag{B.3}$$

using linearization. If the rock porosity is pressure independent, Equation (3.8) can be written as

$$\varphi \frac{\partial \rho(p)}{\partial t} - \nabla\left(\rho(p)\frac{\mathbf{K}}{\mu}(\nabla p - \rho(p) g \nabla d)\right) = \rho(p) q. \tag{B.4}$$

Assuming isotropic permeability, constant depth and fluid with constant velocity reduces to

$$\varphi \frac{\partial \rho(p)}{\partial t} - \frac{\rho_0}{\mu}\nabla(\mathbf{K}\nabla p) - \frac{\rho_0 c_l}{\mu}\nabla(p\mathbf{K}\nabla p) = \rho(p) q. \tag{B.5}$$

If the fluid compressibility is sufficiently small, in the sense that $c_l \nabla(p\mathbf{K}\nabla p) \ll \nabla(\mathbf{K}\nabla p)$, the term $c_l \nabla(p\mathbf{K}\nabla p)$ can be neglected. Finally, the result is

$$\varphi \frac{\partial \rho(p)}{\partial t} - \frac{\rho_0}{\mu}\nabla(\mathbf{K}\nabla p) = \rho(p) q. \tag{B.6}$$

B.2. Discretization of the Compressible Model

The compressible model is time dependent while the incompressible model is not. Therefore, Equation (B.6) contains the time derivative as a term. The spatial part of Equation (B.6) can be discretized in the same manner as in the incompressible case, and written in the form:

$$\varphi \frac{\partial \rho(\mathbf{p})}{\partial t} + \mathbf{T}\mathbf{p} = \bar{\mathbf{q}}(\mathbf{p}), \tag{B.7}$$

where the source term is defined as $\bar{\mathbf{q}}(\mathbf{p}) := \rho(\mathbf{p})\mathbf{q}$ and $\mathbf{T}$ is the transmissibility matrix. In this case, Euler Backwards will be used to perform numerical time integration and the equation will be rewritten as

$$\mathbf{V}\frac{\rho(\mathbf{p}^{k+1}) - \rho(\mathbf{p}^{k})}{\Delta t^{k}} + \mathbf{T}\mathbf{p}^{k+1} = \bar{\mathbf{q}}(\mathbf{p}^{k+1}), \tag{B.8}$$

where $\Delta t^{k} = t^{k+1} - t^{k}$ and $\mathbf{v}$ is the accumulation matrix defined as

$$\mathbf{v} = \Delta x \Delta y \Delta z \mathbf{I}_{n}, \tag{B.9}$$

where $\mathbf{I}_{n} \in \mathbb{R}^{n \times n}$ is the identity matrix.

C

Computational Complexity

In this Chapter, the floating-point operations (flops) and memory storage of each method are given. First, the algorithm is given and then we give the number of flops for each operation. The assumptions are:

Assumptions

- $\mathbf{A} \in \mathbb{R}^{N\times N}$ is sparse with s nonzero elements per row.
- $\mathbf{M} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$ is the incomplete Cholesky preconditioner with zero fill-in of $\mathbf{A}$.flops
- $\mathbf{Z} \in \mathbb{R}^{N\times M}$ is a full matrix.
- $\mathbf{E} \in \mathbb{R}^{M\times M}$ is a full matrix.
- $\mathbf{E} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$ is the Cholesky decomposition of $\mathbf{E}$.

We construct the following matrices:

$$\mathbf{V} = \mathbf{A}\mathbf{Z}, \quad \mathbf{V} \in \mathbb{R}^{N\times M} \tag{C.1}$$
$$\mathbf{W} = \mathbf{Z}\mathbf{E}^{-1}, \quad \mathbf{W} \in \mathbb{R}^{N\times M} \tag{C.2}$$
$$\mathbf{B} = \mathbf{A}\mathbf{V}, \quad \mathbf{B} \in \mathbb{R}^{N\times M}. \tag{C.3}$$
$$\mathbf{H} = \mathbf{M}^{-1}\mathbf{V}, \quad \mathbf{H} \in \mathbb{R}^{N\times M} \tag{C.4}$$

For each iteration, these matrices are computed. It would be cheaper to compute them before each iteration and store the matrices. For the deflation methods we do not compute the Cholesky decomposition of $\mathbf{E}$. For these method, we construct the inverse of $\mathbf{E}$ and this will be used to compute $\mathbf{W}, \mathbf{B}$. The basic operations are given in Table C.1.

Table C.1: Basic operations with the number of flops

Operation	Flops
$\mathbf{x}+\mathbf{y}$	N
$\langle \mathbf{x}, \mathbf{y} \rangle$	$2N-1$
$\mathbf{Ax}$	$(2s-1)N$
Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
Solve $\mathbf{Mx}=\mathbf{y}$	$(2s+1)N$
Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
Construct $\mathbf{E}=\mathbf{Z}^{\mathsf{T}}\mathbf{V}$	$(2N-1)M^2$
Construct $\mathbf{C}$	$\frac{1}{3}M^3$
Solve $\mathbf{Ex}=\mathbf{y}$	$2M^2$
Construct $\mathbf{W}=\mathbf{ZE}^{-1}$	$(2M-1)MN$
Construct $\mathbf{B}=\mathbf{VE}^{-1}$	$(2M-1)MN$
Construct $\mathbf{H}=\mathbf{M}^{-1}\mathbf{V}$	$(2s+1)MN$

The frequently used operations are $\mathbf{Qx}=\mathbf{y}$ and $\mathbf{Px}=\mathbf{y}$. The number of flops of these operations are given in Table C.2.

Table C.2: The number of flops of $\mathbf{Qx}=\mathbf{y}$, $\mathbf{Px}=\mathbf{y}$ and $\mathbf{P}^{\mathsf{T}}\mathbf{x}=\mathbf{y}$.

Operation	Flops
$\mathbf{Qx}=\mathbf{y}$	$(4M-1)N-M$
$\mathbf{Px}=\mathbf{y}$	$4MN-M$
$\mathbf{P}^{\mathsf{T}}\mathbf{x}=\mathbf{y}$	$4MN-M$

Proof. The calculation shown in Table C.2 will be proved. First, compute $\mathbf{Qx}=\mathbf{y}$ where $\mathbf{Q}$ is given by:

$$\mathbf{Q}=\mathbf{ZE}^{-1}\mathbf{Z}^{\mathsf{T}}=\mathbf{WZ}^{\mathsf{T}}. \tag{C.5}$$

The number of flops is calculated as follows:

Table C.3: The number of flops of $\mathbf{Qx}=\mathbf{y}$

$\mathbf{Qx}=\mathbf{y}$	
$\mathbf{y}_1=\mathbf{Z}^{\mathsf{T}}\mathbf{x}$	$2MN-M$
$\mathbf{y}=\mathbf{Wy}_1$	$(2M-1)N$
Total	$(4M-1)N-M$

The process is repeated for $\mathbf{Px}=\mathbf{y}$ and $\mathbf{P}^{\mathsf{T}}\mathbf{x}=\mathbf{y}$. The deflation matrix $\mathbf{P}$ and its transpose $\mathbf{P}^{\mathsf{T}}$ is given by:

$$\mathbf{P}=\mathbf{I}-\mathbf{AQ}=\mathbf{I}-\mathbf{BZ}^{\mathsf{T}},\quad \mathbf{P}^{\mathsf{T}}=\mathbf{I}-\mathbf{QA}=\mathbf{I}-\mathbf{ZB}^{\mathsf{T}}. \tag{C.6}$$

Table C.4: The number of flops of $\mathbf{Px}=\mathbf{y}$ and $\mathbf{P}^{\mathsf{T}}\mathbf{x}=\mathbf{y}$

$\mathbf{Px}=\mathbf{y}$		$\mathbf{P}^{\mathsf{T}}\mathbf{x}=\mathbf{y}$	
$\mathbf{y}_1=\mathbf{Z}^{\mathsf{T}}\mathbf{x}$	$2MN-M$	$\mathbf{y}_1=\mathbf{B}^{\mathsf{T}}\mathbf{x}$	$(2M-1)N$
$\mathbf{y}=\mathbf{x}-\mathbf{By}_1$	$2MN$	$\mathbf{y}=\mathbf{x}-\mathbf{Zy}_1$	$(2M+1)N-M$
Total	$4MN-M$	Total	$4MN-M$

□

C.1. Conjugate Gradient

The algorithm, the number of flops per iteration and memory storage of the Conjugate Gradient method are given in this section.

Algorithm

Algorithm 7 Conjugate Gradient

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$ and $\mathbf{p}^0 = \mathbf{r}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}{\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \rangle}$
        $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
        $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
        $\beta^k = \frac{\langle \mathbf{r}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}$
        $\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta^k \mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{x}^{k+1}$

Computational Efficiency

Table C.5: The number of flops and memory storage of the Conjugate Gradient method

	Operation	Flops
2.	$\mathbf{x}^0 = \overline{\mathbf{x}}$	
	$\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$	$2sN$
	$\mathbf{p}^0 = \mathbf{r}^0$	
	$\gamma^0 = \langle \mathbf{r}^0, \mathbf{r}^0 \rangle$	$2N - 1$
5a.	$\mathbf{w}^k = \mathbf{A}\mathbf{p}^k$	$(2s-1)N$
5.	$\alpha^k = \frac{\gamma^k}{\langle \mathbf{w}^k, \mathbf{p}^k \rangle}$	$2N$
6.	$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$	$2N$
7.	$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{w}^k$	$2N$
8a.	$\gamma^{k+1} = \langle \mathbf{r}^{k+1}, \mathbf{r}^{k+1} \rangle$	$2N - 1$
8.	$\beta^k = \frac{\gamma^{k+1}}{\gamma^k}$	1
9.	$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta^k \mathbf{p}^k$	$2N$
10.	$\mathbf{x}_{it} = \mathbf{x}^{k+1}$	
Total	Initial	$(2s+2)N - 1$
	Iteration	$(2s+9)N$

Memory positions		
Scalars	α^k	1
	β^k	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
Matrices	$\mathbf{A}$	sN
	Total	$(5+s)N + 4$

C.2. Preconditioned Conjugate Gradient

The algorithm, the number of flops per iteration and memory storage of the Preconditioned Conjugate Gradient method are given in this section. The algorithm is given by:

Algorithm 8 Preconditioned Conjugate Gradient

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\mathbf{z}^0 = \mathbf{M}^{-1}\mathbf{r}^0$ and $\mathbf{p}^0 = \mathbf{z}^0$
for $k = 0, \ldots$ **do**
  while $\mathbf{r}^k > \varepsilon$ **do**
    $\mathbf{z}^{k+1} = \mathbf{M}^{-1}\mathbf{r}^k$
    $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{z}^k \rangle}{\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \rangle}$
    $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
    $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
    $\beta^k = \frac{\langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{z}^k, \mathbf{r}^k \rangle}$
    $\mathbf{p}^{k+1} = \mathbf{z}^{k+1} + \beta^k \mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{x}^{k+1}$

Computational Efficiency

Table C.6: The number of flops and memory storage of the Preconditioned Conjugate Gradient method

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
2.	$\mathbf{x}^0 = \overline{\mathbf{x}}$	
	$\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$	$2sN$
	$\mathbf{z}^0 = \mathbf{M}^{-1}\mathbf{r}^0$	$(2s+1)N$
	$\mathbf{p}^0 = \mathbf{z}^0$	
	$\gamma^0 = \langle \mathbf{z}^0, \mathbf{r}^0 \rangle$	$2N-1$
5a.	$\mathbf{w}^k = \mathbf{A}\mathbf{p}^k$	$(2s-1)N$
5.	$\mathbf{z}^{k+1} = \mathbf{M}^{-1}\mathbf{r}^k$	$(2s+1)N$
6.	$\alpha^k = \frac{\gamma^k}{\langle \mathbf{w}^k, \mathbf{p}^k \rangle}$	$2N$
7.	$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$	$2N$
8.	$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{w}^k$	$2N$
9a.	$\gamma^{k+1} = \langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \rangle$	$2N-1$
9.	$\beta^k = \frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1} = \mathbf{z}^{k+1} + \beta^k \mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it} = \mathbf{x}^{k+1}$	
Total	Initial	$\frac{1}{2}(11s+1)N - 1$
	Iteration	$(4s+10)N$

Memory positions		
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\mathbf{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
Total		$\frac{1}{2}(3s+13)N + 4$

C.3. Deflation Method

The algorithm, the number of flops per iteration and memory storage of the Deflated Conjugate Gradient method are given in this section.

Algorithm

Algorithm 9 Deflated Conjugated Gradient

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0, \hat{\mathbf{r}}^0 = \mathbf{P}\mathbf{r}^0$ and $\mathbf{p}^0 = \hat{\mathbf{r}}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\alpha^k = \frac{\langle \hat{\mathbf{r}}^k, \hat{\mathbf{r}}^k \rangle}{\langle \mathbf{p}^k, \mathbf{PAp}^k \rangle}$
        $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \alpha^k \mathbf{p}^k$
        $\hat{\mathbf{r}}^{k+1} = \hat{\mathbf{r}}^k - \alpha^k \mathbf{PAp}^k$
        $\beta^k = \frac{\langle \hat{\mathbf{r}}^{k+1}, \hat{\mathbf{r}}^{k+1} \rangle}{\langle \hat{\mathbf{r}}^k, \hat{\mathbf{r}}^k \rangle}$
        $\mathbf{p}^{k+1} = \hat{\mathbf{r}}^{k+1} + \beta^k \mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top \mathbf{x}^{k+1}$

Computational Efficiency

Table C.7: The number of flops and memory storage of the Deflated Conjugate Gradient method

	Operation	Flops
1.	Construct $\mathbf{V} = \mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E} = \mathbf{Z}^{\mathsf{T}}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W} = \mathbf{ZE}^{-1}$	$(2M-1)MN$
	Construct $\mathbf{B} = \mathbf{VE}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0 = \overline{\mathbf{x}}$	
	$\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0$	$2sN$
	$\hat{\mathbf{r}}^0 = \mathbf{Pr}^0$	$4MN - M$
	$\mathbf{p}^0 = \hat{\mathbf{r}}^0$	
	$\gamma^0 = \langle \hat{\mathbf{r}}^0, \hat{\mathbf{r}}^0 \rangle$	$2N-1$
5a.	$\mathbf{w}^k = \mathbf{PAp}^k$	$(4M+2s-1)N - M$
5.	$\alpha^k = \frac{\gamma^k}{\langle \mathbf{p}^k, \mathbf{w}^k \rangle}$	$2N$
6.	$\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \alpha^k \mathbf{p}^k$	$2N$
7.	$\hat{\mathbf{r}}^{k+1} = \hat{\mathbf{r}}^k - \alpha^k \mathbf{w}^k$	$2N$
8a.	$\gamma^{k+1} = \langle \hat{\mathbf{r}}^{k+1}, \hat{\mathbf{r}}^{k+1} \rangle$	$2N-1$
8.	$\beta^k = \frac{\gamma^{k+1}}{\gamma^k}$	1
9.	$\mathbf{p}^{k+1} = \hat{\mathbf{r}}^{k+1} + \beta^k \mathbf{p}^k$	$2N$
10.	$\mathbf{x}_{it} = \mathbf{Qb} + \mathbf{P}^{\mathsf{T}}\hat{\mathbf{x}}^{k+1}$	$8MN - 2M$
Total	Initial	$(6M+2s+9)MN + (2s+2)N + \frac{1}{3}M^3 - M^2 - 3M - 1$
	Iteration	$(4M+2s+9)N - M$

Memory positions		
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{W}$	MN
	$\mathbf{B}$	MN
	Total	$(4M+s+6)N + 4$

C.4. Deflation Variant

The algorithm, the number of flops per iteration and memory storage of the Deflated Preconditioned Conjugate Gradient method are given in this section.

Algorithm

Algorithm 10 Deflation Variant 1 (DEF1)

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\hat{\mathbf{r}}^0 = \mathbf{P}\mathbf{r}^0$, $\boldsymbol{z}^0 = \mathbf{M}^{-1}\hat{\mathbf{r}}^0$ and $\mathbf{p}^0 = \boldsymbol{z}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\boldsymbol{z}^{k+1} = \mathbf{M}^{-1}\hat{\mathbf{r}}^k$
        $\alpha^k = \frac{\left\langle \hat{\mathbf{r}}^k, \boldsymbol{z}^k \right\rangle}{\left\langle \mathbf{p}^k, \mathbf{P}\mathbf{A}\mathbf{p}^k \right\rangle}$
        $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \alpha^k \mathbf{p}^k$
        $\hat{\mathbf{r}}^{k+1} = \hat{\mathbf{r}}^k - \alpha^k \mathbf{P}\mathbf{A}\mathbf{p}^k$
        $\beta^k = \frac{\left\langle \hat{\mathbf{r}}^{k+1}, \boldsymbol{z}^{k+1} \right\rangle}{\left\langle \hat{\mathbf{r}}^k, \boldsymbol{z}^k \right\rangle}$
        $\mathbf{p}^{k+1} = \boldsymbol{z}^{k+1} + \beta^k \mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top \mathbf{x}^{k+1}$

Algorithm 11 Deflation Variant 2 (DEF2)

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\boldsymbol{z}^0 = \mathbf{M}^{-1}\mathbf{r}^0$ and $\mathbf{p}^0 = \mathbf{P}^\top \boldsymbol{z}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\boldsymbol{z}^{k+1} = \mathbf{M}^{-1}\mathbf{r}^k$
        $\alpha^k = \frac{\left\langle \mathbf{r}^k, \boldsymbol{z}^k \right\rangle}{\left\langle \mathbf{p}^k, \mathbf{A}\mathbf{p}^k \right\rangle}$
        $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
        $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
        $\beta^k = \frac{\left\langle \mathbf{r}^{k+1}, \boldsymbol{z}^{k+1} \right\rangle}{\left\langle \mathbf{r}^k, \boldsymbol{z}^k \right\rangle}$
        $\mathbf{p}^{k+1} = \boldsymbol{z}^{k+1} + \beta^k \mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{x}^{k+1}$

Computational Efficiency

Table C.8: The number of flops and memory storage of the DEF1 method method

	DEF1	
	Operation	**Flops**
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E}=\mathbf{Z}^{\mathsf{T}}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W}=\mathbf{ZE}^{-1}$	$(2M-1)MN$
	Construct $\mathbf{B}=\mathbf{VE}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0=\overline{\mathbf{x}}$	
	$\mathbf{r}^0=\mathbf{b}-\mathbf{A}\mathbf{x}^0$	$2sN$
	$\hat{\mathbf{r}}^0=\mathbf{P}\mathbf{r}^0$	$4MN-M$
	$\boldsymbol{z}^0=\mathbf{M}^{-1}\hat{\mathbf{r}}^0$	$(2s+1)N$
	$\mathbf{p}^0=\boldsymbol{z}^0$	
	$\gamma^0=\langle\hat{\mathbf{r}}^0,\boldsymbol{z}^0\rangle$	$2N-1$
5.	$\boldsymbol{z}^{k+1}=\mathbf{M}^{-1}\hat{\mathbf{r}}^k$	$(2s+1)N$
6a.	$\mathbf{w}^k=\mathbf{P}\mathbf{A}\mathbf{p}^k$	$(4M+2s-1)N-M$
6.	$\alpha^k=\frac{\gamma^k}{\langle\mathbf{p}^k,\mathbf{w}^k\rangle}$	$2N$
7.	$\hat{\mathbf{x}}^{k+1}=\hat{\mathbf{x}}^k+\alpha^k\mathbf{p}^k$	$2N$
8.	$\hat{\mathbf{r}}^{k+1}=\hat{\mathbf{r}}^k-\alpha^k\mathbf{w}^k$	$2N$
9a.	$\gamma^0=\langle\hat{\mathbf{r}}^0,\boldsymbol{z}^0\rangle$	$2N-1$
9.	$\beta^k=\frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1}=\boldsymbol{z}^{k+1}+\beta^k\mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it}=\mathbf{Q}\mathbf{b}+\mathbf{P}^{\mathsf{T}}\mathbf{x}^{k+1}$	$8MN-2M$
Total	Initial	$(6M+2s+9)MN+\frac{1}{2}(11s+1)N+\frac{1}{3}M^3-M^2-3M-1$
	Iteration	$(4M+4s+10)N-M$

	Memory positions	
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\boldsymbol{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{W}$	MN
	$\mathbf{B}$	MN
	Total	$\frac{1}{2}(8M+3s+13)N+4$

Table C.9: The number of flops and memory storage of the DEF2 method method

DEF2

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E}=\mathbf{Z}^{\mathsf{T}}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W}=\mathbf{ZE}^{-1}$	$(2M-1)MN$
	Construct $\mathbf{B}=\mathbf{VE}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0=\mathbf{Qb}+\mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}$	$8MN-2M$
	$\mathbf{r}^0=\mathbf{b}-\mathbf{Ax}^0$	$2sN$
	$\mathbf{z}^0=\mathbf{M}^{-1}\mathbf{r}^0$	$(2s+1)N$
	$\mathbf{p}^0=\mathbf{P}^{\mathsf{T}}\mathbf{z}^0$	$4MN-M$
	$\gamma^0=\langle\mathbf{r}^0,\mathbf{z}^0\rangle$	$2N-1$
5a.	$\mathbf{w}^k=\mathbf{Ap}^k$	$(2s-1)N$
5.	$\alpha^k=\frac{\gamma^k}{\langle\mathbf{p}^k,\mathbf{w}^k\rangle}$	$2N$
6.	$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha^k\mathbf{p}^k$	$2N$
7.	$\mathbf{r}^{k+1}=\mathbf{r}^k-\alpha^k\mathbf{w}^k$	$2N$
8.	$\mathbf{z}^{k+1}=\mathbf{M}^{-1}\mathbf{r}^{k+1}$	$(2s+1)N$
9a.	$\gamma^{k+1}=\langle\mathbf{r}^{k+1},\mathbf{z}^{k+1}\rangle$	$2N-1$
9.	$\beta^k=\frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1}=\mathbf{P}^{\mathsf{T}}\mathbf{z}^{k+1}+\beta^k\mathbf{p}^k$	$(4M+2)N-M$
11.	$\mathbf{x}_{it}=\mathbf{x}^{k+1}$	
Total	Initial	$(6M+2s+9)MN+\frac{1}{2}(11s+1)N+\frac{1}{3}M^3-M^2-3M-1$
	Iteration	$(4M+4s+10)N-M$

	Memory positions	
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\mathbf{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{B}$	MN
Total		$\frac{1}{2}(8M+3s+13)N+4$

C.5. Adapted Deflation Variant

The number of flops of the operator $\mathcal{P}_{\text{A-DEF1}}$ and $\mathcal{P}_{\text{A-DEF2}}$ are given first. Then, we replace the preconditioner in the preconditioner algorithm.

$$\mathcal{P}_{\text{A-DEF1}} = \mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}, \quad \mathcal{P}_{\text{A-DEF2}} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1} + \mathbf{Q} \tag{C.7}$$

The number of flops of these operators are given by:

Table C.10: The number of flops of Adapted Deflation Variant

$\mathbf{x} = (\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{y}$		$\mathbf{x} = (\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1} + \mathbf{Q})\mathbf{y}$	
$\mathbf{x}_1 = \mathbf{Z}^{\mathsf{T}}\mathbf{y}$	$2MN - M$	$\mathbf{x}_1 = \mathbf{Q}\mathbf{y}$	$(4M-1)N - M$
$\mathbf{x}_2 = \mathbf{W}\mathbf{x}_1$	$(2M-1)N$	$\mathbf{x}_2 = \mathbf{M}^{-1}\mathbf{y}$	$(2s+1)N$
$\mathbf{x}_3 = \mathbf{y} - \mathbf{B}\mathbf{x}_1$	$2MN$	$\mathbf{x}_3 = \mathbf{P}^{\mathsf{T}}\mathbf{x}_2$	$4MN - M$
$\mathbf{x}_4 = \mathbf{M}^{-1}\mathbf{x}_3$	$(2s+1)N$	$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_3$	N
$\mathbf{x} = \mathbf{x}_4 + \mathbf{x}_2$	N		
Total	$(6M + 2s + 1)N - M$	Total	$(8M + 2s + 1)N - 2M$

Algorithm

Algorithm 12 Adapted Deflation Variant 1 (A-DEF1)

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\mathbf{z}^0 = (\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{r}^0$ and $\mathbf{p}^0 = \mathbf{z}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\mathbf{z}^{k+1} = (\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{r}^k$
        $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{z}^k \rangle}{\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \rangle}$
        $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
        $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
        $\beta^k = \frac{\langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{z}^k, \mathbf{r}^k \rangle}$
        $\mathbf{p}^{k+1} = \mathbf{z}^{k+1} + \beta^k \mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{x}^{k+1}$

Algorithm 13 Adapted Deflation Variant 2 (A-DEF2)

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\boldsymbol{z}^0 = (\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{Q})\mathbf{r}^0$ and $\mathbf{p}^0 = \boldsymbol{z}^0$
for $k = 0, \ldots$ **do**
  while $\mathbf{r}^k > \varepsilon$ **do**
    $\boldsymbol{z}^{k+1} = (\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{Q})\mathbf{r}^k$
    $\alpha^k = \frac{\langle \mathbf{r}^k, \boldsymbol{z}^k \rangle}{\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \rangle}$
    $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{p}^k$
    $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k\mathbf{A}\mathbf{p}^k$
    $\beta^k = \frac{\langle \boldsymbol{z}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \boldsymbol{z}^k, \mathbf{r}^k \rangle}$
    $\mathbf{p}^{k+1} = \boldsymbol{z}^{k+1} + \beta^k\mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{x}^{k+1}$

Computational Efficiency

Table C.11: The number of flops and memory storage of the A-DEF1 method

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V} = \mathbf{A}\mathbf{Z}$	$(2s-1)MN$
	Construct $\mathbf{E} = \mathbf{Z}^\top\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W} = \mathbf{Z}\mathbf{E}^{-1}$	$(2M-1)MN$
	Construct $\mathbf{B} = \mathbf{V}\mathbf{E}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0 = \overline{\mathbf{x}}$	
	$\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$	$2sN$
	$\boldsymbol{z}^0 = (\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{r}^0$	$(6M+2s+1)N - M$
	$\mathbf{p}^0 = \boldsymbol{z}^0$	
	$\gamma^0 = \langle \boldsymbol{z}^0, \mathbf{r}^0 \rangle$	$2N-1$
5a.	$\mathbf{w}^k = \mathbf{A}\mathbf{p}^k$	$(2s-1)N$
5.	$\boldsymbol{z}^{k+1} = (\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q})\mathbf{r}^k$	$(6M+2s+1)N - M$
6.	$\alpha^k = \frac{\gamma^k}{\langle \mathbf{w}^k, \mathbf{p}^k \rangle}$	$2N$
7.	$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{p}^k$	$2N$
8.	$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k\mathbf{w}^k$	$2N$
9a.	$\gamma^{k+1} = \langle \boldsymbol{z}^{k+1}, \mathbf{r}^{k+1} \rangle$	$2N-1$
9.	$\beta^k = \frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1} = \boldsymbol{z}^{k+1} + \beta^k\mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it} = \mathbf{x}^{k+1}$	
Total	Initial	$(6M+2s+3)MN + \frac{1}{2}(11s+3)N + \frac{1}{3}M^3 - M^2 - M - 1$
	Iteration	$(6M+4s+10)N - M$

	Memory positions	
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\boldsymbol{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN

Table C.12: The number of flops and memory storage of the A-DEF2 method

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E}=\mathbf{Z}^{\mathsf{T}}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W}=\mathbf{ZE}^{-1}$	$(2M-1)MN$
	Construct $\mathbf{B}=\mathbf{VE}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0=\mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}+\mathbf{Qb}$	$8MN-2M$
	$\mathbf{r}^0=\mathbf{b}-\mathbf{Ax}^0$	$2sN$
	$\boldsymbol{z}^0=(\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}+\mathbf{Q})\mathbf{r}^0$	$(8M+2s+1)N-2M$
	$\mathbf{p}^0=\boldsymbol{z}^0$	
	$\gamma^0=\langle \boldsymbol{z}^0,\mathbf{r}^0\rangle$	$2N-1$
5a.	$\mathbf{w}^k=\mathbf{Ap}^k$	$(2s-1)N$
5.	$\boldsymbol{z}^{k+1}=(\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}+\mathbf{Q})\mathbf{r}^k$	$(8M+2s+1)N-2M$
6.	$\alpha^k=\frac{\gamma^k}{\langle \mathbf{w}^k,\mathbf{p}^k\rangle}$	$2N$
7.	$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha^k\mathbf{p}^k$	$2N$
8.	$\mathbf{r}^{k+1}=\mathbf{r}^k-\alpha^k\mathbf{w}^k$	$2N$
9a.	$\gamma^{k+1}=\langle \boldsymbol{z}^{k+1},\mathbf{r}^{k+1}\rangle$	$2N-1$
9.	$\beta^k=\frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1}=\boldsymbol{z}^{k+1}+\beta^k\mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it}=\mathbf{x}^{k+1}$	
Total	Initial	$(6M+2s+13)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-4M-1$
	Iteration	$(8M+4s+10)N-2M$

		Memory positions
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\boldsymbol{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{W}$	MN
	$\mathbf{B}$	MN
	Total	$\frac{1}{2}(8M+3s+13)N+4$

C.6. Reduced BNN

The number of flops of the operator $\mathcal{P}_{\text{R-BNN1}}$ and $\mathcal{P}_{\text{R-BNN2}}$ are given first. Then, we replace the preconditioner in the preconditioner algorithm.

$$\mathcal{P}_{\text{R-BNN1}} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{P}, \quad \mathcal{P}_{\text{R-BNN2}} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1} \tag{C.8}$$

The number of flops of this operator is given by:

Table C.13: The number of flops of $\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{P}\mathbf{y}$ and $\mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{y}$

$\mathbf{x} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{P}\mathbf{y}$	
$\mathbf{x}_1 = \mathbf{P}\mathbf{y}$	$4MN - M$
$\mathbf{x}_2 = \mathbf{M}^{-1}\mathbf{x}_1$	$(2s+1)N$
$\mathbf{x} = \mathbf{P}^{\mathsf{T}}\mathbf{x}_2$	$4MN - M$
Total	$(8M + 2s + 1)N - 2M$

$\mathbf{x} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{-1}\mathbf{y}$	
$\mathbf{x}_1 = \mathbf{M}^{-1}\mathbf{y}$	$(2+1)sN$
$\mathbf{x} = \mathbf{P}^{\mathsf{T}}\mathbf{x}_1$	$4MN - M$
Total	$(4M + 2s + 1)N - M$

Algorithm

The algorithm is the same as the PCG method. The only difference is the starting vector being changed before the iteration. The operator $\mathcal{P}$ is given and is different depending using $\mathcal{P}_{\text{R-BNN1}}$ or $\mathcal{P}_{\text{R-BNN2}}$.

Algorithm 14 Reduced BNN

Initial: $\overline{\mathbf{x}}$, ε
Compute: $\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^{\mathsf{T}}\overline{\mathbf{x}}$, $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$, $\mathbf{z}^0 = \mathcal{P}\mathbf{r}^0$ and $\mathbf{p}^0 = \mathbf{z}^0$
for $k = 0, \ldots$ **do**
    while $\mathbf{r}^k > \varepsilon$ **do**
        $\mathbf{z}^{k+1} = \mathcal{P}\mathbf{r}^k$
        $\alpha^k = \dfrac{\left\langle \mathbf{r}^k, \mathbf{z}^k \right\rangle}{\left\langle \mathbf{A}\mathbf{p}^k, \mathbf{p}^k \right\rangle}$
        $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
        $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
        $\beta^k = \dfrac{\left\langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \right\rangle}{\left\langle \mathbf{z}^k, \mathbf{r}^k \right\rangle}$
        $\mathbf{p}^{k+1} = \mathbf{z}^{k+1} + \beta^k \mathbf{p}^k$
$\mathbf{x}_{it} = \mathbf{x}^{k+1}$

Computational Efficiency

Table C.14: The number of flops and memory storage of the R-BNN1 method

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E}=\mathbf{Z}^{\top}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W}=\mathbf{ZE}^{-1}$	$(2M-1)MN$
	Construct $\mathbf{B}=\mathbf{VE}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0=\mathbf{P}^{\top}\overline{\mathbf{x}}+\mathbf{Qb}$	$8MN-2M$
	$\mathbf{r}^0=\mathbf{b}-\mathbf{Ax}^0$	$2sN$
	$\boldsymbol{z}^0=\mathbf{P}^{\top}\mathbf{M}^{-1}\mathbf{Pr}^0$	$(8M+2s+1)N-2M$
	$\mathbf{p}^0=\boldsymbol{z}^0$	
	$\gamma^0=\langle\boldsymbol{z}^0,\mathbf{r}^0\rangle$	$2N-1$
5a.	$\mathbf{w}^k=\mathbf{Ap}^k$	$(2s-1)N$
5.	$\boldsymbol{z}^{k+1}=\mathbf{P}^{\top}\mathbf{M}^{-1}\mathbf{Pr}^k$	$(8M+2s+1)N-2M$
6.	$\alpha^k=\frac{\gamma^k}{\langle\mathbf{w}^k,\mathbf{p}^k\rangle}$	$2N$
7.	$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha^k\mathbf{p}^k$	$2N$
8.	$\mathbf{r}^{k+1}=\mathbf{r}^k-\alpha^k\mathbf{w}^k$	$2N$
9a.	$\gamma^{k+1}=\langle\boldsymbol{z}^{k+1},\mathbf{r}^{k+1}\rangle$	$2N-1$
9.	$\beta^k=\frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1}=\boldsymbol{z}^{k+1}+\beta^k\mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it}=\mathbf{x}^{k+1}$	
Total	Initial	$(6M+2s+13)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-4M-1$
	Iteration	$(8M+4s+10)N-2M$

	Memory positions	
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\boldsymbol{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{W}$	MN
	$\mathbf{B}$	MN
	Total	$\frac{1}{2}(8M+3s+13)N+4$

Table C.15: The number of flops of the R-BNN2 method

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E}=\mathbf{Z}^{\mathrm{T}}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W}=\mathbf{ZE}^{-1}$	$(2M-1)MN$
	Construct $\mathbf{B}=\mathbf{VE}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0=\mathbf{P}^{\mathrm{T}}\overline{\mathbf{x}}+\mathbf{Qb}$	$8MN-2M$
	$\mathbf{r}^0=\mathbf{b}-\mathbf{Ax}^0$	$2sN$
	$\boldsymbol{z}^0=\mathbf{P}^{\mathrm{T}}\mathbf{M}^{-1}\mathbf{r}^0$	$(4M+2s+1)N-M$
	$\mathbf{p}^0=\boldsymbol{z}^0$	
	$\gamma^0=\langle\boldsymbol{z}^0,\mathbf{r}^0\rangle$	$2N-1$
5a.	$\mathbf{w}^k=\mathbf{Ap}^k$	$(2s-1)N$
5.	$\boldsymbol{z}^{k+1}=\mathbf{P}^{\mathrm{T}}\mathbf{M}^{-1}\mathbf{r}^k$	$(4M+2s+1)N-M$
6.	$\alpha^k=\frac{\gamma^k}{\langle\mathbf{w}^k,\mathbf{p}^k\rangle}$	$2N$
7.	$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha^k\mathbf{p}^k$	$2N$
8.	$\mathbf{r}^{k+1}=\mathbf{r}^k-\alpha^k\mathbf{w}^k$	$2N$
9a.	$\gamma^{k+1}=\langle\boldsymbol{z}^{k+1},\mathbf{r}^{k+1}\rangle$	$2N-1$
9.	$\beta^k=\frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1}=\boldsymbol{z}^{k+1}+\beta^k\mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it}=\mathbf{x}^{k+1}$	
Total	Initial	$(6M+2s+9)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-4M-1$
	Iteration	$(4M+4s+10)N-M$

		Memory positions
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\boldsymbol{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{W}$	MN
	$\mathbf{B}$	MN
	Total	$\frac{1}{2}(8M+3s+13)N+4$

C.7. ROM-based Preconditioner

The algorithm, the number of flops per iteration and memory storage of the Preconditioned Conjugate Gradient method are given in this section. The preconditioner used in this section is different than the one used in the PCG method. Therefore, we refer to section C.2 for the algorithm.

Algorithm

The operation steps is the same as the PCG method. The only difference is replacing the preconditioner at step 5. The ROM-based preconditioner is given by:

$$\mathcal{P}_{\text{ROM}} = \mathbf{M}^{-1} + \mathbf{Q}(\mathbf{I} - \mathbf{A}\mathbf{M}^{-1}) \tag{C.9}$$

Table C.16: The number of flops of constructing ROM-based preconditioner

$\mathbf{x} = \mathcal{P}_{\text{ROM}}\mathbf{y}$	
$\overline{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{y}$	$(2s+1)N$
$\mathbf{r} = \mathbf{y} - \mathbf{A}\overline{\mathbf{x}}$	$2sN$
$\overline{\mathbf{r}} = \mathbf{Z}^{\top}\mathbf{r}$	$2MN - M$
$\mathbf{e} = \mathbf{W}\overline{\mathbf{r}}$	$(2M-1)N$
$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{e}$	N
Total	$(4M + 4s + 1)N - M$

Computational Efficiency

Table C.17: The number of flops of the Preconditioned Conjugate Gradient method using ROM-based preconditioner

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E}=\mathbf{Z}^{\mathsf{T}}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{W}=\mathbf{ZE}^{-1}$	$(2M-1)MN$
2.	$\mathbf{x}^0=\overline{\mathbf{x}}$	
	$\mathbf{r}^0=\mathbf{b}-\mathbf{Ax}^0$	$2sN$
	$\boldsymbol{z}^0=\mathcal{P}_{\text{ROM}}\mathbf{r}^0$	$(4M+4s+1)N-M$
	$\mathbf{p}^0=\boldsymbol{z}^0$	
	$\gamma^0=\langle \boldsymbol{z}^0,\mathbf{r}^0\rangle$	$2N-1$
5a.	$\mathbf{w}^k=\mathbf{Ap}^k$	$(2s-1)N$
5.	$\boldsymbol{z}^{k+1}=\mathcal{P}_{\text{ROM}}\mathbf{r}^k$	$(4M+4s+1)N-M$
6.	$\alpha^k=\frac{\gamma^k}{\langle \mathbf{w}^k,\mathbf{p}^k\rangle}$	$2N$
7.	$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha^k\mathbf{p}^k$	$2N$
8.	$\mathbf{r}^{k+1}=\mathbf{r}^k-\alpha^k\mathbf{w}^k$	$2N$
9a.	$\gamma^{k+1}=\langle \boldsymbol{z}^{k+1},\mathbf{r}^{k+1}\rangle$	$2N-1$
9.	$\beta^k=\frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1}=\boldsymbol{z}^{k+1}+\beta^k\mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it}=\mathbf{x}^{k+1}$	
Total	Initial	$(4M+2s+2)MN+\frac{1}{2}(15s+3)N+\frac{1}{3}M^3-M^2-M-1$
	Iteration	$(4M+6s+10)N-M$

	Memory positions	
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\boldsymbol{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{W}$	MN
	Total	$\frac{1}{2}(6M+3s+13)N+4$

C.8. SROM-based Preconditioner

The algorithm, the number of flops per iteration and memory storage of the Preconditioned Conjugate Gradient method are given in this section. The preconditioner used in this section is different than the one used in the PCG method. Therefore, we refer to section C.2 for the algorithm.

Algorithm

The operation steps is the same as PCG method. The only difference is replacing the preconditioner at step 5. The symmetric ROM-based preconditioner is given by:

$$\mathcal{P}_{\text{SROM}} = \mathbf{M}^{-1} + \mathbf{Q} - \frac{1}{2}(\mathbf{QAM}^{-1} + \mathbf{M}^{-1}\mathbf{AQ}) \tag{C.10}$$

Table C.18: The number of flops of constructing SROM-based preconditioner

$\mathbf{x} = \mathcal{P}_{\text{SROM}}\mathbf{y}$	
$\overline{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{y}$	$(2s+1)N$
$\mathbf{r}_1 = \mathbf{Z}^\top\mathbf{y}$	$2MN - M$
$\mathbf{r}_2 = \mathbf{r}_1 - \frac{1}{2}\mathbf{H}^\top\mathbf{y}$	$2MN + M$
$\overline{\mathbf{e}}_1 = \mathbf{E}^{-1}\mathbf{r}_1$	$2M^2$
$\overline{\mathbf{e}}_2 = \mathbf{E}^{-1}\mathbf{r}_2$	$2M^2$
$\mathbf{e}_1 = \frac{1}{2}\mathbf{H}\overline{\mathbf{e}}_1$	$2MN$
$\mathbf{e}_2 = \mathbf{Z}\overline{\mathbf{e}_2}$	$2MN - N$
$\mathbf{x} = \overline{\mathbf{x}} - \mathbf{e}_1 + \mathbf{e}_2$	$3N$
Total	$(8M + 2s + 3)N + 4M^2$

Computational Efficiency

Table C.19: The number of flops of the Preconditioned Conjugate Gradient method using SROM-based preconditioner

	Operation	Flops
1.	Construct $\mathbf{L}$	$\frac{1}{2}(3s-3)N$
	Construct $\mathbf{V}=\mathbf{AZ}$	$(2s-1)MN$
	Construct $\mathbf{E}=\mathbf{Z}^{\top}\mathbf{V}$	$(2N-1)M^2$
	Construct $\mathbf{C}$	$\frac{1}{3}M^3$
	Construct $\mathbf{H}=\mathbf{M}^{-1}\mathbf{V}$	$(2s+1)MN$
2.	$\mathbf{x}^0=\overline{\mathbf{x}}$	
	$\mathbf{r}^0=\mathbf{b}-\mathbf{A}\mathbf{x}^0$	$2sN$
	$\mathbf{z}^0=\mathcal{P}_{\text{SROM}}\mathbf{r}^0$	$(8M+2s+3)N+4M^2$
	$\mathbf{p}^0=\mathbf{z}^0$	
	$\gamma^0=\langle\mathbf{z}^0,\mathbf{r}^0\rangle$	$2N-1$
5a.	$\mathbf{w}^k=\mathbf{A}\mathbf{p}^k$	$(2s-1)N$
5.	$\mathbf{z}^{k+1}=\mathcal{P}_{\text{SROM}}\mathbf{r}^k$	$(8M+2s+3)N+4M^2$
6.	$\alpha^k=\frac{\gamma^k}{\langle\mathbf{w}^k,\mathbf{p}^k\rangle}$	$2N$
7.	$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha^k\mathbf{p}^k$	$2N$
8.	$\mathbf{r}^{k+1}=\mathbf{r}^k-\alpha^k\mathbf{w}^k$	$2N$
9a.	$\gamma^{k+1}=\langle\mathbf{z}^{k+1},\mathbf{r}^{k+1}\rangle$	$2N-1$
9.	$\beta^k=\frac{\gamma^{k+1}}{\gamma^k}$	1
10.	$\mathbf{p}^{k+1}=\mathbf{z}^{k+1}+\beta^k\mathbf{p}^k$	$2N$
11.	$\mathbf{x}_{it}=\mathbf{x}^{k+1}$	
Total	Initial	$(2M+4s+8)MN+\frac{1}{2}(11s+7)N+\frac{1}{3}M^3+3M^2-1$
	Iteration	$(8M+4s+12)N+4M^2$

		Memory positions
Scalars	α	1
	β	1
	γ^k	1
	γ^{k+1}	1
Vectors	$\mathbf{x}^k$	N
	$\mathbf{b}$	N
	$\mathbf{p}^k$	N
	$\mathbf{r}^k$	N
	$\mathbf{w}^k$	N
	$\mathbf{z}^k$	N
Matrices	$\mathbf{A}$	sN
	$\mathbf{L}$	$\frac{1}{2}(s+1)N$
	$\mathbf{C}$	$\frac{1}{2}(M+1)M$
	$\mathbf{Z}$	MN
	$\mathbf{V}$	MN
	$\mathbf{H}$	MN
Total		$\frac{1}{2}(6M+3s+13)N+\frac{1}{2}(M+1)M+4$

C.9. Conclusion

The overview of the number of flops can be found in Table C.20 and memory storage can be found in C.21.

Table C.20: Overview table of the number of flops of the above discussed methods.

Methods	**Flops**	
	Initial	Iterations
CG	$(2s+2)N-1$	$(2s+9)N$
PCG	$\frac{1}{2}(11s+1)N-1$	$(4s+10)N$
DCG	$(6M+2s+9)MN+(2s+2)N+\frac{1}{3}M^3-M^2-3M-1$	$(4M+2s+9)N-M$
DEF1	$(6M+2s+9)MN+\frac{1}{2}(11s+1)N+\frac{1}{3}M^3-M^2-3M-1$	$(4M+4s+10)N-M$
DEF2	$(6M+2s+9)MN+\frac{1}{2}(11s+1)N+\frac{1}{3}M^3-M^2-3M-1$	$(4M+4s+10)N-M$
A-DEF1	$(6M+2s+3)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-M-1$	$(6M+4s+10)N-M$
A-DEF2	$(6M+2s+13)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-4M-1$	$(8M+4s+10)N-2M$
BNN	$(6M+2s+9)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-3M-1$	$(12M+4s+10)N-3M$
R-BNN1	$(6M+2s+13)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-4M-1$	$(8M+4s+10)N-2M$
R-BNN2	$(6M+2s+9)MN+\frac{1}{2}(11s+3)N+\frac{1}{3}M^3-M^2-4M-1$	$(4M+4s+10)N-M$
ROM	$(4M+2s+2)MN+\frac{1}{2}(15s+3)N+\frac{1}{3}M^3-M^2-M-1$	$(4M+6s+10)N-M$
SROM	$(2M+4s+8)MN+\frac{1}{2}(11s+7)N+\frac{1}{3}M^3+3M^2-1$	$(8M+4s+12)N+4M^2$

Table C.21: Overview table of the memory storage of the above discussed methods.

Methods	**Memory positions**
CG	$(5+s)N+4$
PCG	$\frac{1}{2}(3s+13)N+4$
DCG	$(4M+s+6)N+4$
DEF1	$\frac{1}{2}(8M+3s+13)N+4$
DEF2	$\frac{1}{2}(8M+3s+13)N+4$
A-DEF1	$\frac{1}{2}(8M+3s+13)N+4$
A-DEF2	$\frac{1}{2}(8M+3s+13)N+4$
BNN	$\frac{1}{2}(8M+3s+13)N+4$
R-BNN1	$\frac{1}{2}(8M+3s+13)N+4$
R-BNN2	$\frac{1}{2}(8M+3s+13)N+4$
ROM	$\frac{1}{2}(6M+3s+13)N+4$
SROM	$\frac{1}{2}(6M+3s+13)N+\frac{1}{2}(M+1)M+4$

Focusing on number of iterations of each method, the flops in term of order N is given in Table C.22. Clearly, the cheapest method per iteration of the deflation methods is the DEF1 method, the DEF2 method, the ROM method and the R-BNN2 method using $\mathcal{O}(4MN)$ flops per iteration.
It has been proven that the A-DEF2 method and the ROM method. The only difference lies in the way of implementing the operators. The value depends on the size of the deflation-subspace matrix $\mathbf{Z} \in \mathbb{R}^{N\times M}$ and sparsity s.
The most expensive method is the BNN method. Then, the R-BNN1 method and the SROM method are expensive with using $\mathcal{O}(8MN)$ flopsper iteration.

Table C.22: Overview table of the iteration the number of flops of the above discussed methods of order N.

Methods	**Flops**
CG	$(2s+9)N$
PCG	$(4s+10)N$
DCG	$(4M+2s+9)N$
DEF1	$(4M+4s+10)N$
DEF2	$(4M+4s+10)N$
A-DEF1	$(6M+4s+10)N$
A-DEF2	$(8M+4s+10)N$
BNN	$(12M+4s+10)N$
R-BNN1	$(8M+4s+10)N$
R-BNN2	$(4M+4s+10)N$
ROM	$(4M+6s+10)N$
SROM	$(8M+4s+12)N$

Bibliography

[1] Jonsthovel TB A, van Gijzen Martin B, Vuik Cornelis, and Scarpas. On the Use of Rigid Body Modes in the Deflated Preconditioned Conjugate Gradient Method. *SIAM Journal on Scientific Computing*, 35(1):B207–B225, 2013. ISSN 1064-8275. doi: 10.1137/100803651. URL http://epubs.siam.org/doi/10.1137/100803651.

[2] Jönsthövel TB A, Van Gijzen MB, MacLachlan S, Vuik C, and Scarpas. Comparison of the deflated preconditioned conjugate gradient method and algebraic multigrid for composite materials. *Computational Mechanics*, 50(3):321–333, 2011. ISSN 0178-7675. doi: 10.1007/s00466-011-0661-y. URL http://link.springer.com/10.1007/s00466-011-0661-y.

[3] Khalid Aziz and Antoni Settati. *Petroleum Reservoir Simulation*. Chapman & Hall, 1979.

[4] V. Forstall and K. Carlberg R. Tuminaro. Krylov-subspace recycling via the {POD}-augmented conjugate-gradient algorithm. *arXiv:1512.05820 [math]*, 37(3):1–32, 2015. ISSN 0895-4798. doi: 10.1137/16M1057693. URL http://arxiv.org/abs/1512.05820.

[5] G.B. Diaz Cortes, C. Vuik and J.D. Jansen. POD-based deflation techniques for the solution of two-phase flow problems in large and highly heterogeneous porous media. Technical Report 18-1, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2018.

[6] A.W. Heemink and P.T.M. Vermeulen 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. ISSN 03091708. doi: 10.1016/j.advwatres.2003.09.008. URL http://linkinghub.elsevier.com/retrieve/pii/S0309170803001490.

[7] T Heijn, R Markovinović, J D Jansen, and et al. Generation of Low-Order Reservoir Models Using System-Theoretical Concepts. In *SPE Journal*, volume 9, pages 3–5. Society of Petroleum Engineers, 2004. ISBN 1111111111. doi: 10.2118/88361-PA.

[8] Jan Dirk Jansen. A systems description of flow through porous media. *Springer*, (April): 130, 2013. doi: 10.1007/978-3-319-00260-6.The. URL http://link.springer.com/content/pdf/10.1007/978-3-319-00260-6.pdf.

[9] Jocelyne and Y. Saad M. Yeung J. Erhel F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926, 2000. ISSN 1064-8275. doi: 10.1137/s1064829598339761.

[10] K. Kahl and H. Rittich. The Deflated Conjugate Gradient Method: Convergence, Perturbation and Accuracy. *Linear Algebra and its Applications*, 515:111–129, 2012. URL http://arxiv.org/abs/1209.1963.

[11] L Yu Kolotilina. Preconditioning of systems of linear algebraic equations by means of twofold deflation. I. *Theory. J. Math. Sci*, 89:1652–1689, 1998.

[12] 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. ISSN 0036-1429. doi: 10.1137/S0036142900382612. URL http://epubs.siam.org/doi/10.1137/S0036142900382612.

[13] Knut-Andreas. Lie. *An introduction to reservoir simulation using MATLAB - user guide for the Matlab Reservoir Simulation Toolbox (MRST)*. SINTEF ICT, 2014.

[14] 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.

[15] Jan Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(3):233–241. doi: 10.1002/cnm.1640090307. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640090307.

[16] Jan MANDEL. Hybrid Domain Decomposition with Unstructured Subdomains. *Contemporary Mathematics*, 157, 1994.

[17] Jan Mandel and Marian Brezina. Balancing domain decomposition for problems with large jumps in coefficients. *Mathematics of Computation of the American Mathematical Society*, 65(216):1387–1401, 1996.

[18] R. Markovinović and J. van Doren J.D. Jansen. Reduced-order optimal control of water flooding using proper orthogonal decomposition. *Computational Geosciences*, 10(1):137–158, 2006. ISSN 1420-0597. doi: 10.1007/s10596-005-9014-2. URL http://link.springer.com/10.1007/s10596-005-9014-2.

[19] R Nabben and C Vuik. A Comparison of Deflation and Coarse Grid Correction Applied to Porous Media Flow. *SIAM Journal on Numerical Analysis*, 42(0):1631–1647, 2004. ISSN 0036-1429. doi: 10.1137/S0036142903430451.

[20] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. URL https://epubs.siam.org/doi/pdf/10.1137/0724027.

[21] Damiano Pasetto, Massimiliano Ferronato, and Mario Putti. A reduced order model-based preconditioner for the efficient solution of transient diffusion equations. *International Journal for Numerical Methods in Engineering*, 109(8):1159–1179, 2017. ISSN 10970207. doi: 10.1002/nme.5320. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.5320#.

[22] D.W. Peaceman. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation(includes associated paper 6988). *Society of Petroleum Engineers Journal*, 18(03):183–194, 1978. ISSN 0197-7520. doi: 10.2118/6893-PA. URL http://www.onepetro.org/doi/10.2118/6893-PA.

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

[24] Jonathan Richard Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical Report CS-94-125, 1994. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.418&rep=rep1&type=pdf%5Cnhttp://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

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

[26] J M Tang, R Nabben, C Vuik, and Y A Erlangga. REPORT 07-04 THEORETICAL AND NUMERICAL COMPARISON OF VARIOUS PROJECTION METHODS DERIVED FROM DEFLATION, DOMAIN DECOMPOSITION AND MULTIGRID METHODS. 2007. ISSN 1389-6520.

[27] F J Vermolen, C Vuik, and A Segal. Deflation in preconditioned conjugate gradient methods for Finite Element Problems. pages 1–22, 2002. doi: 10.1007/978-3-642-18560-1{_}7.

[28] J.C. Vink and P. Astrid G. Papaioannou J.D. Jansen. Pressure Preconditioning Using Proper Orthogonal Decomposition. In *SPE Reservoir Simulation ...*, number February, pages 21–23, 2011. ISBN 9781617823862. URL http://www.onepetro.org/mslib/servlet/onepetropreview?id=SPE-141922-MS.

[29] C. Vuik and G.B. Diaz Cortes J.D. Jansen. Physics-based pre-conditioners for large-scale subsurface flow simulation. Technical report, Delft University of Technology, Department of Applied Mathematics, 2016.

[30] C. Vuik and G.B. Diaz Cortes J.D. Jansen. On POD-based Deflation Vectors for DPCG applied to porous media problems. Technical Report 17-1, Delft University of Technology, Delft Institute of Applied Mathematics, 2018. URL http://engineering.purdue.edu/~mark/puthesis.

[31] C Vuik and D J P Lahaye. Scientific Computing (wi4201). 2015.

[32] C. Vuik and J.M. Tang R. Nabben Y. Erlangga. Theoretical and Numerical Comparison of Various Projection Methods derived from Deflation, Domain Decomposition and Multigrid Methods 1. *Journal of scientific computing*, 31429621(3):1–30, 2007.

[33] 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(1):385–403, 1999. ISSN 00219991. doi: 10.1006/jcph.1999.6255.

[34] C Vuik, A Segal, J.A Meijerink, and G.T Wijma. The Construction of Projection Vectors for a Deflated ICCG Method Applied to Problems with Extreme Contrasts in the Coefficients. *Journal of Computational Physics*, 172(2):426–450, 9 2001. ISSN 0021-9991. doi: 10.1006/JCPH.2001.6795. URL https://www.sciencedirect.com/science/article/pii/S0021999101967956.

[35] K. Willcox and P. Astrid S. Weiland T. Backx. Missing Point Estimation in Models Described by Proper Orthogonal Decomposition. *Autom. Control. IEEE Trans.*, 53(10):2237–2251, 2008. ISSN 0018-9286. doi: 10.1109/TAC.2008.2006102.