POD-Based Deflation Method For Reservoir Simulation

Diaz Cortes, Gabriela

DOI
10.4233/uuid:0458fd29-920b-43cb-8c2b-e04be8db0dc7

Publication date
2019

Document Version
Final published version

Citation (APA)
Diaz Cortes, G. (2019). *POD-Based Deflation Method For Reservoir Simulation*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:0458fd29-920b-43cb-8c2b-e04be8db0dc7

POD-based deflation method for reservoir simulation

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
Maandag 18, Maart, 2019 om 10:00 uur

door

Gabriela Berenice DIAZ CORTES
Master of Science in Material Engineering, Universidad Nacional Autónoma de México, UNAM, México

geboren te Distrito Federal, México

Dit proefschrift is goedgekeurd door de promotoren Prof.dr.ir. C. Vuik and Prof.dr.ir. J.D. Jansen.

Samenstelling promotiecommissie bestaat uit:

Rector Magnificus	voorzitter
Prof.dr.ir. C. Vuik	Technische Universiteit Delft, promotor
Prof.dr.ir. J.D. Jansen	Technische Universiteit Delft, promotor

Onafhankelijke leden:

Dr. D. Pasetto	EPFL, Switzerland
ScD.dr.ir. L. Sheremetov	IMP, México
Dr. M. Ferronato	U-Padova, Italy
Dr. H. Hajibeygi	Technische Universiteit Delft
Prof.dr.ir. C.W. Oosterlee	Technische Universiteit Delft
Prof.dr.ir. L.J. Sluys	Technische Universiteit Delft, reservelid

POD-based deflation method for reservoir simulation
Dissertation at Delft University of Technology.

This research was funded by the Mexican Insitute of Petroleum (IMP), the 'Consejo Nacional de Ciencia y Tecnología (CONACYT)', and the 'Secretaría de Energía (SENER)' through the programs: Programa de Captación de Talento, Reclutamiento, Evaluación y Selección de Recursos Humanos (PCTRES) and 'Formación de Recursos Humanos Especializados para el Sector Hidrocarburos (CONACYT-SENER Hidrocarburos, 2015-2016, CUARTO PERIODO). This project was supervised by ScD.dr.ir. L. Sheremetov, IMP, México.
ISBN 978-94-6380-284-0

Printed by: ProefschriftMaken || www.proefschriftmaken.nl

Samenvatting

Simulaties van enkel-fase en twee-fasen stromingen door zeer heterogene poreuze media resulteren in grote slecht-geconditioneerde systemen van vergelijkingen, waaronder een gelineariseerd druksysteem dat bijzonder tijdrovend kan zijn om op te lossen. Daarom zijn uitgebreide inspanningen vereist om manieren te vinden om dit probleem effectief aan te pakken.

Iteratieve methoden zijn, samen met preconditioneringstechnieken, de meest gekozen technieken om deze problemen op te lossen. In de literatuur kunnen we ook Model Orde Reductie (MOR) en deflatiemethoden vinden, waarbij systeeminformatie opnieuw wordt gebruikt om een goede benadering sneller en goedkoper te vinden.

Onder de MOR-technieken is de Gepaste Orthogonale Ontbinding (POD) geïmplementeerd voor de versnelling van het gelineariseerde druksysteem. Deze techniek is gebaseerd op het verkrijgen van relevante systeeminformatie in een reeks vectoren om de oplossing van gerelateerde systemen te versnellen.

Wat deflatie betreft, is het nodig om een optimale selectie te vinden van deflatie- of projectievectoren die aan het systeem zijn gekoppeld om een iteratief oplossingsproces te versnellen. De gebruikelijke keuzes van deflatievectoren zijn echter ofwel duur om te verkrijgen, bijvoorbeeld eigenvectoren van de systeemmatrix, of probleemafhankelijk, bijvoorbeeld subdomeinvectoren.

In dit werk introduceren we een op POD gebaseerde deflatiemethode die de voordelen van beide methoden combineert. De dominante kenmerken van het systeem worden vastgelegd in een reeks POD-basisvectoren, die later als deflatievectoren worden gebruikt om de oplossing van lineaire systemen te versnellen. Sommige kenmerken van deze methodologie worden hieronder weergegeven

Snapshots-verzameling en POD-basisberekening. De POD-basis wordt verkregen uit een reeks snapshots die op effectieve wijze de dynamiek van het onderzochte systeem vastleggen. Om de snapshots te verkrijgen en dus de basis te berekenen, bestuderen we verschillende gevallen:

Recycling. De snapshots zijn een reeks oplossingen met iets andere parameters dan het oorspronkelijke systeem. Later wordt de POD-basis verkregen uit deze reeks snapshots. We implementeren deze aanpak voor onsamendrukbare enkel-fase problemen, en het wordt voornamelijk gebruikt om de toepasbaarheid van de methodologie te onderzoeken.

Bewegende venster. Deze aanpak is vooral geschikt voor tijdsvariërende problemen. Met deze benadering wordt de meest recent verkregen informatie, d.w.z.

de oplossing van de eerdere tijdstappen, gebruikt om het aankomende lineaire systeem op te lossen. Deze benadering is geïmplementeerd voor samendrukbare enkel-fase -en onsamendrukbare twee-fasen stromingsproblemen.

Training fase. Ook geschikt voor tijdsafhankelijke problemen, wordt met deze benadering een pre-simulatie uitgevoerd om de relevante systeemkenmerken in de POD-basis vast te leggen. Zodra de basis is verkregen, wordt deze gebruikt om problemen op te lossen met vergelijkbare parameters als de trainingssimulatie.

Naast de toepasbaarheid van deze aanpak bestuderen we verschillende manieren om de snapshots tijdens de pre-simulatie te verzamelen door de systeemconfiguratie te wijzigen en we testen deze met verschillende problemen.

We vergelijken de prestaties van een Deflated Preconditioned Conjugate Gradient-methode (DPCG) met de prestaties van de enkel voorgeconditioneerde versie van de Conjugate Gradient-methode (PCG). Alle onderzochte benaderingen versnellen de PCG-methode bij het gebruik van de leeggelopen versie DPCG. De beste prestaties van de DICG-methode verminderden het aantal iteraties tot slechts 8% van het aantal PCG-iteraties. In dit geval heeft de DPCG het werk (aantal iteraties maal het werk per iteratie) verminderd tot 11% van het PCG-werk, tijdens het iteratieproces.

Bovendien, als de door de POD-basis gegenereerde ruimte dicht bij de systeemoplossing ligt, bereikte DPCG een goede benadering ($\mathcal{O}(10^{-4})$) in één iteratie. Deze optimale prestatie werd verkregen wanneer de numerieke berekening van de momentopname werd uitgevoerd met een kleine fouttolerantie van de lineaire oplosser.

POD-basisvectoren als deflatievectoren. We bestuderen het gedrag van de deflatiemethode voor verschillende keuzes van deflatievectoren. We vergelijken de prestaties van de methode bij het gebruik van systeem-eigenvectoren, subdomeinvectoren, een lineair onafhankelijke (l.o.) reeks momentopnamen die de oplossingsruimte overspannen, en een POD-basis verkregen uit een reeks snapshots, die niet noodzakelijk de oplossingsruimte overspannen.

Uit deze vergelijking, wordt een betere prestatie van de deflatiemethode waargenomen bij gebruik van de gehele l.o. reeks en een POD-basis. Verder kan de informatie vervat in de l.o. reeks efficiënt worden vastgelegd in de POD-basis, d.w.z. de POD-basis is een goede benadering van de systeemoplossing.

Vergelijking van 2L-PCG methoden. De deflatiemethode die is geïmplementeerd voor de versnelling van de PCG-methode kan ook worden beschouwd als een Two-Level Preconditioned Conjugate Gradient methode (2L-PCG).
De prestaties van verschillende 2L-PCG methoden worden vergeleken met behulp van een POD-basis als deflatievectoren, samen met de op AMG gebaseerde preconditioners die zijn ontwikkeld door Passetto et al. [1].

Deze methoden verminderen het aantal iteraties tot slechts $\sim 12\% - 22\%$ PCG iteraties, en het werk tijdens het iteratieproces tot $\sim 40\% - 80\%$. De DEF1-variant wordt in dit werk gebruikt onder de varianten die de beste prestaties bieden (12% van de PCG iteraties en $\sim 40\%$ van het PCG werk). Bovendien genereren de MOR-methoden hetzelfde spectrum van de voorgeconditioneerde matrix als sommige andere deflatievarianten (A-DEF1 / 2) en zijn hun operatoren equivalent. Ook kunnen hun prestaties worden verbeterd als een speciale startvector wordt gebruikt.

De 2L-PCG-methoden zijn onderverdeeld in Klasse 0 of Klasse 1, afhankelijk van of de kleinste eigenwaarden van het systeem respectievelijk op nul of één worden ingesteld voor elke willekeurige reeks deflatievectoren. De MOR-methode werd geclassificeerd als Klasse 1 en de symmetrische versie daarvan vertoonde alleen hetzelfde gedrag als een Klasse 1-operator als de eigenvectoren van de systeemmatrix als deflatievectoren werden gebruikt.

Toepasbaarheid van de POD-gebaseerde deflatiemethode. De toepasbaarheid van de POD-gebaseerde deflatiemethode is niet afhankelijk van de testcase. Hoewel het werd getest op reservoirsimulatie problemen, kan het worden geïmplementeerd voor elk probleem dat in tijd varieert.

Verder bestuderen we de toepasbaarheid ervan voor verschillende 2L-PCG-werkwijzen, maar het kan ook worden geïmplementeerd tezamen met vele andere lineaire oplossers, bijvoorbeeld multigrid, multilevel en domeinontledingstechnieken. De implementatie kan ook worden uitgebreid met verschillende preconditioners.

Summary

Simulation of single- and two-phase flow through highly heterogeneous porous media results in large ill-conditioned systems of equations. In particular, the linearized pressure system can be particularly time-consuming to solve. Therefore, extensive efforts to find ways to address this issue effectively are required.

Iterative methods, together with preconditioning techniques, are the most commonly chosen techniques to solve these problems. In the literature, we can also find Reduced Order Models (ROM) and deflation methods, where system information is reused to find a good approximation more quickly and less costly.

Among the ROM techniques, Proper Orthogonal Decomposition (POD) has been implemented for the acceleration of the linearized pressure system. This technique is based on the acquisition of relevant system information in a set of vectors to accelerate the solution of related systems.

Regarding deflation, it is required to find an optimal selection of deflation or projection vectors, associated with the system, to speed up an iterative solution process. However, the common choices of deflation vectors are either expensive to obtain, e.g., eigenvectors of the system matrix, or problem dependent, e.g., subdomain vectors.

In this work, we introduce a POD-based deflation method that combines the advantages of both methodologies. The dominant features of the system are captured in a set of POD basis vectors, which are used later as deflation vectors to accelerate the solution of linear systems.

The implementation of the proposed POD-based deflation method consists of three stages:

Snapshots collection. A set of solutions related to the system is obtained (more details are given below).

POD basis computation. A POD basis is obtained from the snapshots.

Solution of the linear system. The POD basis vectors are used as deflation vectors to solve problems with the deflation method.

Some features of the methodology followed throughout this work for these three stages are presented next.

Snapshots collection and POD basis computation. The POD basis is obtained from a set of snapshots that effectively capture the dynamics of the system under investigation. To obtain the snapshots, and, therefore, to compute the basis, we study various cases:

Recycling. The snapshots are a set of solutions with slightly different parameters than the original system.
Test cases: We implement this approach for incompressible single-phase problems, and it is mainly used to explore the applicability of the methodology.

Moving window. This approach is especially suited for time-varying problems. With this approach, the most recently-obtained information, i.e., the solutions of the linear systems obtained for the previous time steps, are used to solve the upcoming linear system.
Test cases: This approach is implemented for compressible single-phase, and incompressible two-phase flow problems.

Training phase. Also suited for transient problems, a pre-simulation is run with this approach, and all the solutions obtained during the pre-simulation are used as snapshots.

Test cases: This approach is implemented for incompressible two-phase flow problems to solve problems with similar parameters as the training simulation. We also study various ways to collect the snapshots during the pre-simulation.

Performance of the POD-based deflation method. We implement the POD-based deflation method for the acceleration of the Preconditioned Conjugate Gradient (PCG) method preconditioned with the Incomplete Cholesky factorization (ICCG). The performance of the Deflated Preconditioned Conjugate Gradient method (DICCG) is compared with the performance of the only preconditioned version of the method (ICCG).

Selection of deflation vectors. In this work, we show theoretically that if a linearly independent (*l.i.*) set of recycled vectors spanning the solution space is used as deflation vectors, the convergence of the deflation method is achieved in one iteration (see Lemma 4.1.3). This result is also illustrated with incompressible single-phase reservoir simulation problems containing large contrast in the permeability coefficients: an 'academic' layered problem with a contrast between layers of 10^1 and 10^6, and the SPE 10 benchmark with contrast in permeability coefficients of $\mathcal{O}(10^7)$.

If the set of deflation vectors is not linearly independent, but it contains a *l.i.* set of vectors that span the solution space, a *l.i.* set can be obtained by computing a POD basis from the non-*l.i.* set. This POD basis also leads to convergence in one iteration. This implies that the information contained in the *l.i.* set can be captured in the POD basis, i.e., the POD basis is a good approximation to the system solution.

For a problem with s sources, i.e., s elements in the right-hand side (rhs) different from zero, and Neumann boundary conditions, the *l.i.* set spanning the solution space consist on $s-1$ vectors. If non-homogeneous Dirichlet boundary conditions are selected and s sources are included, the *l.i.* set spanning the solution space consists on $s+1$ vectors, s related to the sources and one related to the boundary conditions.

If this set is reduced, i.e., it does not span the solution anymore, convergence is no longer achieved in one iteration. For these cases, the deflation method only reduces the number of ICCG iterations. Furthermore, the selection of the previously-mentioned POD-based as deflation vectors, achieves a more significant reductions in

the number of iterations. Also, more accelerations are achieved for problems with Neumann boundary conditions.

This behavior is illustrated with the SPE 10 problem ($\mathcal{O}(10^6)$ cells) with five sources and homogeneous Neumann; and four sources and non-homogeneous Dirichlet boundary conditions. For the first case, three POD basis vectors reduce the number of ICCG iterations to 33%, while using the same number of *l.i.* vectors reduce the number of ICCG iterations to 78%. For the second, four deflation vectors reduce this number to 83% with POD basis vectors and 94% with *l.i* vectors. Similar performance is obtained for the layered test cases.

We compare the performance of the deflation method when using the proposed *l.i.* set of snapshots spanning the solution space, and the POD basis described before, system eigenvectors, and subdomain vectors (five subdomains for the layered problem containing five layers). The best performance was achieved with the *l.i.* set and the POD basis requiring one DICCG iteration to converge. The other methods required from 21% to 96% of the number of ICCG, using the same number of deflation vectors as in the previous selections (*l.i.* set and POD basis vectors).

The DICCG method accelerates the ICCG method for all the studied approaches, including study cases with gravity and capillary pressure terms. The reductions to the number of ICCG iterations range from 10% to 54% with a reduction of work of 21% to 98% for an approximation with a residual of $\mathcal{O}(10^{-7})$. Moreover, the DICCG method achieved an approximation of $\mathcal{O}(10^{-4})$ after one iteration, for most of the cases.

Including capillary pressure terms increase the number of DICCG iterations up to 10% in the studied cases. However,nincreasing the Corey coefficient from 2 to 4 for the wetting phase does not change this number considerably, only 3% the number of ICCG iterations. For a set of experiments considering gravity terms, increasing the height of the reservoir results in a reduction of around 10% the number of ICCG iterations changing from a reservoir of size 1 [m] to one of 4 [m].

Comparison of 2L-PCG methods. The deflation method implemented for the acceleration of the PCG method can also be regarded as a Two-Level Preconditioned Conjugate Gradient method (2L-PCG).
The performance of various 2L-PCG methods is compared using a POD basis as deflation vectors, together with the AMG-based preconditioners developed by Passetto et al. [1].

These methods reduce the number of iterations to only $\sim 12\% - 22\%$ PCG iterations, and the work during the iteration process to $\sim 40\% - 80\%$. Being the DEF1 variant used throughout this work among the variants presenting the best performance (12% of PCG iterations and $\sim 40\%$ of PCG work). Furthermore, the ROM methods generate the same spectrum of the preconditioned matrix as some other deflation variants (A-DEF1/2), and their operators are equivalent. Moreover, the performance of these methods can be improved with an especial starting vector, different to the one suggested by Pasetto et al. [1].

The 2L-PCG methods are divided into Class 0 or Class 1, depending on whether the smallest eigenvalues of the system are set to zero or one, respectively, for any arbitrary set of deflation vectors. The ROM method belongs to the Class 1 methods, and its symmetric version only showed the same behavior as a Class 1 operator if

eigenvectors of the system matrix are used as deflation vectors.

Applicability of the POD-based deflation method. The applicability of the POD-based deflation method does not depend on the test case. It is tested for reservoir simulation problems, but it can be implemented for any time-varying problem.

Furthermore, we study its applicability for various 2L-PCG methods, but it can also be implemented together with many other linear solvers, e.g., multigrid, multilevel, and domain decomposition techniques. The implementation can also be extended to include various preconditioners.

Chapter 1

Introduction

Simulation has become an important tool for analyzing and predicting system's behavior in many research and industrial areas, e.g., fluid dynamics, material science, seismology and weather forecasting.

In order to perform simulations of a physical phenomenon, it is necessary to construct a model that represents the essential features as realistically as possible. Once an accurate model is found, a mathematical representation is made, usually involving a system of differential equations.

These systems can be solved analytically on a limited number of cases; however, in most cases, a discretized numerical model is required. The physical and mathematical models can become more complex by, e.g., making a more detailed description of the phenomena or by studying the interaction with other systems.

As the complexity of the system increases, solving the discretized numerical models becomes more challenging, and ways of addressing this issue effectively are needed. In this work, we study the dynamics of fluids inside oil reservoirs, which, depending on the phenomena involved, can lead to very complicated systems.

1.1 Background and problem definition

The dynamics of fluids are studied not only as a separate phenomenon to understand a fluid's behavior but also as the interaction of the fluid with the surroundings. Underground flow simulations are performed to explore the fluid's dynamics inside a porous medium. In particular, for reservoirs containing hydrocarbons, predicting the performance under various exploitation schemes is essential for optimizing the recovery of the stored fluids.

The physical model used to simulate flow through porous media involves the principle of mass conservation and a constitutive equation that accounts for the conservation of momentum, known as Darcy's law [2–7]. These principles, for a fluid α, can be represented by the following equations:

$$\frac{\partial(\phi\rho_\alpha S_\alpha)}{\partial t} + \nabla\cdot(\rho_\alpha \mathbf{v}_\alpha) = \rho_\alpha q_\alpha, \qquad \mathbf{v}_\alpha = -\frac{\mathbf{K}_\alpha}{\mu_\alpha}(\nabla p_\alpha - \rho_\alpha g \nabla d), \tag{1.1}$$

where, ρ_α, μ_α and p_α are the density, viscosity, and pressure of the fluid; $\mathbf{K}_\alpha$ accounts for the permeability of a phase and ϕ is the porosity of the medium; g is the gravity constant; d is the depth of the reservoir and q_α are sources, usually, fluids injected into the reservoir. The saturation of a fluid phase, S_α, is the fraction of void space filled with a fluid α in the medium, where a zero saturation indicates that the phase is not present.

While studying the flow of a single fluid, we can simplify this model by, e.g., assuming no gravity terms, and constant porosity, permeability, and viscosity. Combining Equations (1.1) for the simplified case, leads to the following system:

$$-\nabla \cdot (\mathbf{K}_\alpha \nabla p_\alpha) = \mu_\alpha \rho_\alpha q_\alpha,$$

which is an elliptic equation that after discretization and given suitable boundary conditions, can be solved with direct or iterative methods [8, 9].

However, as the size of the problem increases, direct methods become slow, and the iterative methods come to be the only methods able to solve it. Furthermore, this is a simplified model that does not represent reality accurately. Consequently, the assumptions made have to be reconsidered and, depending on the study case, the complexity can increase remarkably.

In many cases, reservoir simulation involves large and highly heterogeneous problems, i.e., problems with significant variations on the permeability coefficients $\mathbf{K}_\alpha$ [1, 10–13], which lead to *ill-conditioned* matrices, requiring large computing times to find the solution. Furthermore, if we have a time-varying problem, we must compute a large number of simulations, which makes the solution of the problem very expensive.

When simulating two or more phases, the unknowns become the saturation and the pressure, where solving the pressure system is the most time-consuming part of the process.

In this work, we focus on the solution of the pressure equation. We begin with small incompressible single-phase problems and we increase the complexity by increasing the size and adding time dependence. Later, we study two-phase problems including gravity and capillary pressure terms.

1.2 Discretization

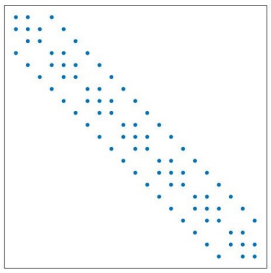


Figure 1.1: *Sparse matrix*

The model used to simulate flow through porous media results in a system of nonlinear equations that can depend on time and space (see Equation (1.1)). Under certain assumptions, the resulting system becomes linear and no pre-processing is required to solve it. Nonetheless, in many cases, some linearization processes are required, e.g., the Newton-Raphson method [14, 15].

Once the system is linearized, it is necessary to discretize it. For the spatial derivatives, finite differences, finite elements or isogeometric analysis methods, among others, can be used [14, 16, 17]. Time derivatives can be approximated with, e.g., the forward and backward Euler methods.

In the case of incompressible single-phase reservoir simulation, discretization of the resulting system (Equation (1.1)) leads to a linear system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1.2}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{x}$ the unknowns vector and $\mathbf{b}$ the right-hand side vector, which includes the sources and boundary conditions of the system.

For reservoir simulation problems, the system matrix is *sparse* [18], i.e., contains only a few non-zero diagonals (see Figure 1.1). The value of the entries depends on the permeability coefficients; moreover, the contrast in the coefficients is related to the complexity of the system.

A way to quantify the complexity is via the condition number κ[†], a quantity that depends on the eigenvalues of the system matrix. This number influences the performance of the solution methods. Hence, acceleration can be achieved by transformation of the linear system (1.2) into a new one faster to solve [9, 16, 18].

1.3 State-of-the-art of acceleration methods

The most widely used approach to speed up the solution of extreme problems is combining iterative solvers with preconditioning techniques. The latest techniques are implemented to speed up the convergence of the iterative methods by changing the system into another one with the same solution but smaller condition number [8, 9, 19, 20].

However, in some cases, a small set of extreme eigenvalues are responsible for the large condition number, and preconditioning techniques are no longer able to reduce it. Therefore, new techniques have to be developed, that together with the usual preconditioned iterative methods can find approximate solutions in a faster way.

Among others, Multigrid (MG) [21], Multilevel [22, 23], Multiscale [13], Domain Decomposition (DD) [24, 25], Proper Orthogonal Decomposition (POD), [26–30] and deflation techniques [16, 22, 23, 31–33], have been studied to accelerate iterative methods for large and ill-conditioned problems. Extensive literature exists, with new and innovative ways of approaching deflation and POD methodologies [1, 11, 12, 20, 26, 34–37].

Deflation techniques can be used to remove the influence of the extreme eigenvalues by creating a subspace, where they are no longer present, accelerating the solution process accordingly. To achieve an optimal performance, it is necessary to find deflation vectors that contain most of the system's variability. If a good selection is made, we can obtain a significant decrease in the total simulation time, with only a small increase in the required computing time per iteration.

Currently, the selection of the deflation vectors is mainly based on some standard approaches: approximated eigenvectors, recycling solutions [34, 38, 39], subdomain deflation vectors [32] and multigrid and multilevel-based deflation matrices [23, 25]. However, a good selection of deflation vectors is problem-dependent, which implies the need of finding good deflation vectors for each kind of problem.

POD methods are based on the collection of a series of snapshots, i.e., solutions of the system with slightly different characteristics, from which essential system in-

[†]For more details about the condition number see Section 2.3

formation will be captured on a basis. Acceleration with POD methods has been approached with diverse ways of collecting and recycling the information.

Some *state-of-the-art* POD acceleration strategies can be found in Astrid et al. [11], who propose the capture of system information in an offline phase for later reuse, accelerating problems with slightly modified parameters in a smaller subspace created with this basis. This approach is particularly useful for optimization or history-matching problems where very similar systems need to be solved.

Similarly, Markovinovic et al. [12] propose using the solution computed with POD methods to find a more accurate initial guess. Pasetto et al. [1] construct a preconditioner based on AMG, deflation, and POD for the acceleration of a Krylov-subspace iterative method.

Following the ideas of [1, 11, 12, 20], we propose the use of POD of many snapshots to capture the system's behavior and combine this technique with deflation to accelerate the convergence of a Krylov iterative method.

In this work, instead of computing the solution in a low dimensional subspace, the basis obtained with POD is proposed as an alternative choice for the deflation vectors to accelerate the convergence of the pressure solution in reservoir simulation. We refer to this methodology as POD-based deflation method, and we study its applicability and properties.

1.4 Proposal: POD-based deflation method

We present a method that combines deflation and POD techniques to further accelerate the solution of a linear system with iterative methods. We combine the main advantages of both methods: POD collects the most relevant system information on a basis; deflation reuses system information via a set of deflation vectors. With the proposed strategy, the POD basis is used as *subspace deflation* matrix within a deflation procedure.

For the collection of the snapshots, we use three different schemes. For the first one, we collect a set of linearly independent snapshots that capture the most relevant system information. These snapshots are obtained for a time independent problem by changing the system configuration, in particular, the right-hand sides.

The second and third approaches are for time-dependent problems. In the second one, the snapshots are captured $on-the-fly$, i.e., the solution of a small number of previous time steps are used as snapshots. Therefore, the system configuration is the same. It is referred to as *moving window* approach. The last one is named *training phase* approach. Here, the snapshots are solutions of a full simulation, i.e., all the time steps are computed. In this approach, the right-hand side of the system is varied randomly in an interval.

The main advantages of this method are:

- Problem independence: it does not depend on the case under study. We introduce this methodology for reservoir simulation examples, but it can be adapted to any time-varying problem.

- Linear solver independence: the acceleration was tested for the Preconditioned Conjugate Gradient (PCG) method. However, it can be implemented for mul-

tiple preconditioners and linear solvers, e.g., Krylov subspace linear solvers, Multilevel, Multigrid and Domain Decomposition methods.

1.5 Thesis outline

This thesis is divided as follows:

Chapter 2: Porous media flow. We begin by describing the problem under consideration. We introduce the physical and mathematical models describing single- and two-phase flow through porous media, i.e., we present the governing equations. Later, we describe the discretization methods required to approximate the solution of this problem numerically.

Chapter 3: Solution methods. We give an overview of the basic iterative methods together with Krylov subspace iterative methods. Additionally, we present the PCG method studied in this work, and we give more details about the acceleration techniques that are the basis of our methodology.

Chapter 4: POD-based deflation method. We introduce the methodology developed throughout this work, resulting from the combination of POD and deflation techniques. We start by analyzing the characteristics of the combined methods. We demonstrate some Lemmas that give us more insight into the behavior and applicability of the method. Finally, we study the spectral performance of the linear system while using this method.

Chapters 5, 6 & 7: Numerical experiments. These chapters are dedicated to the numerical experiments that illustrate the analysis performed in Chapter 4 and to study the performance of the POD-based deflation methodology.

Chapter 5 is devoted to an incompressible single-phase problem, Chapter 6 explores the compressible single-phase case, and Chapter 7 studies the two-phase incompressible case.

Chapter 8: Comparison of 2L-PCG using deflation techniques. In this chapter, we introduce the deflation method as a two-level preconditioned conjugate gradient method (2L-PCG). We present a theoretical and a numerical comparison of the approach presented by Pasetto et al. [1] and a series of 2L-PCG methods, including the one studied throughout this work.

Conclusions. Finally, we present the general conclusions and we give some recommendations for the application of the method.

Chapter 2

Porous media flow: Mathematical model and discretization schemes

Throughout this thesis, we study the acceleration of linear systems of equations resulting during reservoir simulation. When studying flow through large and highly-heterogeneous reservoirs, the solution of the system becomes difficult with standard methods. Nowadays, there is a pursuit of efficient linear solvers to tackle these problems; however, not all the methods work for all the problems. Therefore, it is important to know the properties of the system under consideration to select the most suitable solution methodology, exploiting in this way, the system and solver properties.

To better understand the complexity of the tackled problems throughout this work, in this chapter we give more insight into the systems under investigation. We begin with presenting the principles that govern the dynamics of single- and two-phase flow through porous media, together with the boundary conditions and sources that make a complete system description. Finally, we introduce spatial and temporal discretization schemes that give rise to the linear system studied all over this work.

2.1 Reservoir simulation

Petroleum reservoirs are layers of sedimentary rock, which vary concerning grain size, mineral and clay contents. Reservoir simulation is a way to analyze and predict the fluid behavior inside a reservoir through the analysis of a model, which can be a geological or a mathematical model.

The geological model describes the reservoir, i.e., the rock formation, for which a set of petrophysical properties are defined. The main reservoir properties are the *rock porosity* (ϕ) defined as the fraction of void space inside the rock, and the *rock permeability* ($\mathbf{K}$) that determines the rock's ability to transmit fluids through the reservoir. The rock permeability, in general, is a tensor where each entry (K_{ij}) rep-

resents the flow rate in one direction, i, caused by the pressure drop in the same, or in a perpendicular direction, j.

For the mathematical modeling, we describe the flow through porous media making use of the principle of mass conservation and Darcy's law, corresponding to momentum conservation. The mass balance equation for a fluid phase α is given by:

$$\frac{\partial(\phi\rho_\alpha S_\alpha)}{\partial t} + \nabla\cdot(\rho_\alpha \mathbf{v}_\alpha) = \rho_\alpha q_\alpha, \tag{2.1}$$

and the Darcy's law reads:

$$\mathbf{v}_\alpha = -\frac{\mathbf{K}_\alpha}{\mu_\alpha}\cdot(\nabla p_\alpha - \rho_\alpha g\Delta d), \tag{2.2}$$

where, ρ_α, μ_α and p_α are the density, viscosity, and pressure of the fluid phase α; g is the gravity constant, d is the depth of the reservoir and q_α are sources, usually, fluids injected and/or produced. The saturation of a phase, S_α, is the fraction of void space filled with that phase in the medium, where a zero saturation indicates that the phase is not present. Fluids inside a reservoir are usually filling the empty space completely, this property is expressed by the following relation:

$$\sum_\alpha S_\alpha = 1. \tag{2.3}$$

If our system consists of more than one fluid phase, the permeability of each fluid phase, α, will be affected by the presence of the other phases. Therefore, the effective permeability $\mathbf{K}_\alpha$ has to be used instead of the absolute permeability $\mathbf{K}$. The absolute and effective permeabilities are related via the saturation-dependent relative permeability ($k_{r\alpha}$):

$$\mathbf{K}_\alpha = k_{r\alpha}(S_\alpha)\mathbf{K}.$$

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

$$k_{r\alpha} = (\hat{S}_\alpha)^{n_\alpha} k^0_\alpha, \qquad k_{r_\beta} = (1-\hat{S}_\alpha)^{n_\beta} k^0_\beta, \tag{2.4}$$

where $\hat{S}_\alpha$ is the effective saturation. The Corey coeficients ($n_{\alpha,\beta} > 1$), and the endpoint relative permeabilities ($k^0_{\alpha,\beta}$) are fitting parameters.

Darcy's law can be rewritten as:

$$\mathbf{v}_\alpha = -\lambda_\alpha(S_\alpha)\cdot(\nabla p_\alpha - \rho_\alpha g\Delta d), \tag{2.5}$$

where we introduce the phase mobilities:

$$\lambda_\alpha(S_\alpha) = \frac{k_{r\alpha}(S_\alpha)\mathbf{K}}{\mu_\alpha}. \tag{2.6}$$

The fluid density and the rock porosity can be pressure dependent, i.e., $\rho_\alpha = \rho_\alpha(p)$ and $\phi = \phi(p)$. These dependencies can be expressed as:

$$c_r = \frac{1}{\phi}\frac{d\phi}{dp}, \qquad c_f = \frac{1}{\rho_\alpha}\frac{d\rho_\alpha}{dp}, \tag{2.7}$$

where c_r is the *rock compressibility*, and c_f is the *fluid compressibility*. To describe the system completely, we need to define boundary conditions and, if present, include source terms. The boundary conditions can be prescribed pressures (Dirichlet conditions), fixed flow rates (Neumann conditions) or a combination of these (Robin conditions). For reservoirs, the source terms are wells where fluids are injected or extracted at constant surface flow rate or constant bottom-hole pressure (bhp).

In this work, we make use of the Peaceman's model to describe the pressure change due to the presence of wells. Taking into account gravity forces, this model is given by:

$$q = -J_{well}(p - p_{bh} - \rho g \Delta d_w), \tag{2.8}$$

where Δd_w is the vertical distance from the cell to the surface, J_{well} is known as the well index, p is the pressure in the reservoir and p_{bh} is the pressure inside the well.

2.1.1 Single-phase flow

If the system consists of only one phase, combining Darcy's law and the mass balance equation we obtain:

$$\frac{\partial(\phi\rho)}{\partial t} - \nabla \cdot [\rho\lambda \cdot (\nabla p - \rho g \Delta d)] = \rho q. \tag{2.9}$$

Incompressible fluid. Given an incompressible rock ($\frac{d\phi}{dt} = 0$), when the fluid is incompressible, i.e., the fluid's density is constant and does not depend on the pressure, Equation 2.9 reduces to an elliptic equation for the pressure:

$$-\nabla \cdot [\lambda \cdot (\nabla p - \rho g \Delta d)] = q, \tag{2.10}$$

where $\lambda = \mathbf{K}/\mu$.

Slightly compressible fluid. For this case, the accumulation term cannot be neglected, instead, making use of Equation (2.7) and assuming an incompressible rock ($\frac{d\phi}{dt} = 0$), it can be expressed as follows:

$$\frac{\partial(\rho\phi)}{\partial t} = \rho\frac{\partial\phi}{\partial p}\frac{\partial p}{\partial t} + \phi\frac{\partial\rho}{\partial p}\frac{\partial p}{\partial t} = \phi\rho\left[\frac{1}{\phi}\frac{\partial\phi}{\partial p} + \frac{1}{\rho}\frac{\partial\rho}{\partial p}\right]\frac{\partial p}{\partial t} =$$

$$= \phi\rho\left[c_r + c_f\right]\frac{\partial p}{\partial t} = \phi\rho c_t\frac{\partial p}{\partial t},$$

where $c_t = c_r + c_f$ denotes the total compressibility. Assuming small spatial variations for the density, i.e., $\nabla \cdot \rho = 0$, Equation (2.9) reads:

$$\phi c_t\frac{\partial p}{\partial t} - \nabla \cdot [\lambda \cdot (\nabla p - \rho g \Delta d)] = q. \tag{2.11}$$

Compressible fluid. For a compressible fluid, the density depends on the pressure, $\rho(p)$, in Equation (2.9):

$$\phi \frac{\partial \rho(p)}{\partial t} - \nabla \cdot [\rho(p)\lambda \cdot (\nabla p - \rho(p) g \Delta d)] = \rho q. \tag{2.12}$$

The compressible system, Equation (2.12), is non-linear and some linearization strategies are required to solve it numerically. Through this work, we use the Newton-Raphson (NR) linearization method.

2.1.2 Two-phase flow

For simulation of two-phase flow through a porous medium, we assume that the phases are separated, i.e., they are immiscible, and there is no mass transfer between them. We consider one of the fluids as the wetting phase (w), which is more attracted to the mineral particles than the other phase, known as the non-wetting phase (nw). In the case of a water-oil system, water is considered the wetting phase.

The surface tension and the curvature of the interface between the fluids causes a difference in pressure among phases known as the capillary pressure (p_c) that depends on the saturation as follows:

$$p_c(S_w) = p_{nw} - p_w. \tag{2.13}$$

The pressure of the non-wetting fluid is higher than the pressure of the wetting fluid; therefore, the capillary pressure is always a positive quantity. The relation between the capillary pressure and the saturation is an empirical model based on experiments and it depends on the porosity and permeability of the medium.

As mentioned before, the governing equations of flow through porous media are the Darcy's law, Equation (2.2), and principle of mass conservation, Equation (2.1), combining these equations we obtain a parabolic equation for pressures and saturations:

$$\frac{\partial (\phi \rho_\alpha S_\alpha)}{\partial t} - \nabla \cdot (\rho_\alpha \lambda_\alpha \cdot (\nabla p_\alpha - \rho_\alpha g \Delta d)) = \rho_\alpha q_\alpha. \tag{2.14}$$

For this system, the pressure and saturation can be decoupled via, e.g., the fractional flow formulation. For an immiscible, incompressible flow, the pressure equation becomes elliptic and the transport equation becomes hyperbolic, and they can be solved separate in a sequential procedure. In the next section we describe this formulation in more detail.

Fractional flow formulation. In the case of incompressible flow, the porosity (ϕ) and the densities (ρ_α) do not depend on the pressure. Therefore, Equation (2.14) reduces to:

$$\phi \frac{\partial S_\alpha}{\partial t} - \nabla \cdot (\lambda_\alpha (\nabla p_\alpha - \rho_\alpha g \Delta d)) = q_\alpha. \tag{2.15}$$

Taking a two-phase system with a wetting (w) and a non wetting phase (nw), the

resulting governing equations are:

$$\begin{aligned}\phi\frac{\partial S_w}{\partial t} + \nabla\cdot\mathbf{v}_w =& \phi\frac{\partial S_w}{\partial t} + \nabla\cdot(\lambda_w(\nabla p_w - \rho_w g\Delta d)) = q_w,\\ \phi\frac{\partial S_{nw}}{\partial t} + \nabla\cdot\mathbf{v}_{nw} =& \phi\frac{\partial S_{nw}}{\partial t} + \nabla\cdot(\lambda_{nw}(\nabla p_{nw} - \rho_{nw} g\Delta d)) = q_{nw}.\end{aligned} \tag{2.16}$$

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

$$\begin{aligned}\mathbf{v} = \mathbf{v}_w + \mathbf{v}_{nw} &= -\lambda_{nw}\nabla p_{nw} - \lambda_w\nabla p_w + (\lambda_{nw}\rho_{nw} + \lambda_w\rho_w)g\Delta d\\ &= -(\lambda_{nw} + \lambda_w)\nabla p_{nw} + \lambda_w\nabla p_c + (\lambda_{nw}\rho_{nw} + \lambda_w\rho_w)g\Delta d.\end{aligned} \tag{2.17}$$

If we add the two continuity equations, system (2.16), and use the relationship $S_{nw} + S_w = 1$, we obtain:

$$\phi\frac{\partial(S_w + S_{nw})}{\partial t} + \nabla\cdot(\mathbf{v}_w + \mathbf{v}_{nw}) = \phi\frac{\partial(S_w + S_{nw})}{\partial t} + \nabla\cdot\mathbf{v} = q, \tag{2.18}$$

where $q = q_{nw} + q_w$ is the total source term. Defining the total mobility as $\lambda = \lambda_{nw} + \lambda_w$, and using Darcy's law, Equation (2.18) becomes:

$$-\nabla\cdot(\lambda\nabla p_{nw}) = q - \nabla[\lambda_w\nabla p_c + (\lambda_{nw}\rho_{nw} + \lambda_w\rho_w)g\Delta d], \tag{2.19}$$

which is an equation for the pressure of the non wetting phase. This equation depends on the saturation via the capillary pressure p_c and the total mobility λ. Multiplying each phase velocity by the relative mobility of the other phase and subtracting the result, together with Equation (2.17), we get:

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

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

$$\mathbf{v}_w = \frac{\lambda_w}{\lambda}\mathbf{v} + \frac{\lambda_w\lambda_{nw}}{\lambda}[\nabla p_c + (\rho_w - \rho_{nw})g\Delta d]. \tag{2.20}$$

We introduce the fractional flow function,

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

which, together with the previously computed velocity $\mathbf{v}_w$, transforms the transport Equation (2.1) into:

$$\phi\frac{\partial S_w}{\partial t} + \nabla\cdot[f_w(\mathbf{v} + \lambda_{nw}\Delta\rho g\Delta d)] + \nabla\cdot(f_w\lambda_{nw}\nabla p_c) = q_w, \tag{2.21}$$

where $\Delta\rho = \rho_w - \rho_{nw}$.

With this approach, the system is expressed in terms of the non wetting phase pressure, Equation (2.19), and the saturation of the wetting phase, Equation (2.21).

For the pressure equation, the coupling to saturation is present via the phase mobilities, Equation (2.6), and the derivative of the capillary function.

Solution of the two-phase flow using the fractional flow formulation results in a sequential procedure where, first, the pressure of the non-wetting phase for the current time step (p_{nw}^t) is obtained from Equation (2.12), where the current mobility ($\lambda^t(S^{t-1})$, see Equation (2.6)) is computed using the saturation of the previous time step. The recently obtained pressure is then used to compute the total velocity ($\mathbf{v}^t(p_{nw}^t)$, see Equation (2.17)). And finally, the saturation of the wetting phase is also updated with the total velocity ($S_w^t(\mathbf{v}^t(p_{nw}^t))$, see Equation (2.21)). The procedure is summarized in Algorithm 1.

Algorithm 1 Sequential procedure for incompressible two-phase flow simulation (no gravity and capillary pressure terms included).

% Time integration
for $t = 0, ...,$ steps
  % Compute the pressure of the non-wetting phase during the current time step, Equation (2.12),
    $-\nabla \cdot (\lambda^t \nabla p^t) = q^t$.
  % Compute the total velocity, Equation (2.17),
    $\mathbf{v}^t = -\lambda^t \nabla p^t$.
  % Update the saturation for the current time step, Equation (2.21),
    $\phi \frac{\partial S_w^t}{\partial t} + \nabla \cdot (f_w^t \mathbf{v}^t) = q_w^t$.
end for

2.2 Discretization methods

In Section 2.1, we presented the equations that describe single and two-phase flow through porous media. The resulting elliptic pressure systems, Equation (2.10) and Equation (2.19), consisting of spatial derivatives can be approximated using, e.g., finite differences, finite volumes, or finite elements schemes.

For the discretization of temporal derivatives, Equation (2.12), and the transport Equation (2.21), commonly used discretization schemes are backward and forward Euler. In this section, we describe the discretization mentioned above methods typically used to model flow through porous media. Throughout this work, the matrices and right-hand sides of the studied linear systems are obtained with the Matlab Reservoir Simulation toolbox (MRST, [7]); therefore, the implementation of the discretization schemes is carried out by the software.

2.2.1 Spatial discretization

As mentioned before, flow through a porous medium is described using the mass conservation principle and the Darcy's law. To solve the resulting system of equations, they have to be discretized. In this section we explore the spatial discretization schemes for the elliptic equation appearing from simulation of an incompressible fluid without gravity terms given by Equation (2.10) for one phase, and Equation (2.19)

for two phases. Assuming no gravity and no capillary pressure terms present, the Equation (2.10) transforms into:

$$-\nabla \cdot (\lambda \cdot \nabla p) = q, \quad \text{where} \quad \mathbf{v} = -\lambda \cdot \nabla p. \tag{2.22}$$

Next, we present the discretization of the Equation (2.22) using finite differences and finite volumes schemes, a similar procedure can be used for the discretization of Equation (2.19).

Finite differences

In this section we present the approximation of Equation (2.22) using a cell-centered finite difference scheme. For a 3D model, taking a mesh with a uniform grid size Δx, Δy, Δz, where (i, j, l) is the center of the cell in the position i for the $x-$direction, j for the $y-$direction, and l for the $z-$direction (x_i, y_j, z_l), and $p_{i,j,l} = p(x_i, y_j, z_l)$ is the pressure at this point. Using the harmonic average,

$$\lambda_{i-\frac{1}{2},j,l} = \frac{2\lambda_{i-1,j,l}\lambda_{i,j,l}}{\lambda_{i-1,j,l} + \lambda_{i,j,l}},$$

for the mobility at the interface between cells $(i-1, j, l)$ and (i, j, l), the derivative in the $x-$direction becomes (see, e.g. [2, 5, 6, 33]):

$$\begin{aligned} &\frac{\partial}{\partial x}\left(\lambda \frac{\partial p}{\partial x}\right) = \frac{\Delta}{\Delta x}\left(\lambda \frac{\Delta p}{\Delta x}\right) + \mathscr{O}(\Delta x^2) \\ &= \frac{\lambda_{i+\frac{1}{2},j,l}(p_{i+1,j,l} - p_{i,j,l}) - \lambda_{i-\frac{1}{2},j,l}(p_{i,j,l} - p_{i-1,j,l})}{(\Delta x)^2} + \mathscr{O}(\Delta x^2). \end{aligned} \tag{2.23}$$

Discretizing the other directions in a similar way, Equation (2.22), together with boundary conditions, can be written as:

$$\mathbf{T}\mathbf{p} = \mathbf{q}, \tag{2.24}$$

where $\mathbf{T}$ is known as the transmissibility matrix [6], with elements

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

and $\mathbf{q}$ is the right-hand side, that can be boundary conditions or sources. The stencil of this matrix is given by

$$-p_{i-1}T_{i-1/2} + p_{i-1}(T_{i-1/2} + T_{i+1/2}) - p_{i+1}T_{i+1/2},$$

thus, it is a symmetric matrix.

Finite volumes

Finite volumes methods are derived from conservation of physical quantities over cell volumes, averaging the functions in a set of volumes. Taking a cell Ω_i as control volume, Equation (2.22) for the x direction can be rewritten as:

$$\int_{\Omega_i} \nabla \cdot \mathbf{v} \; d\mathbf{x} = \int_{\partial\Omega_i} \mathbf{v} \cdot \mathbf{n} \; ds = \int_{\Omega_i} q \; d\mathbf{x}. \tag{2.25}$$

The half-faces, $\Gamma_{i,k} = \partial\Omega_i \cup \partial\Omega_k$, are the interface between cells associated with a normal vector $\mathbf{n}_{i,k}$. For each interior half-face $\Gamma_{i,k}$ it exist a twin half-face $\Gamma_{k,i}$ with identical area but opposite normal vector $\mathbf{n}_{k,i} = -\mathbf{n}_{i,k}$. Integrating Equation (2.25) over the cell face, the flux across the interface between cells i and k can be written as:

$$\mathbf{v}_{i,k} = \int_{\Gamma_{i,k}} \mathbf{v} \cdot \mathbf{n} ds \approx \mathbf{A}_{i,k}\mathbf{v}(\mathbf{x}_{i,k}) \cdot \mathbf{n}_{i,k} = -\mathbf{A}_{i,k}(\lambda \cdot \nabla p)\mathbf{x}_{i,k} \cdot \mathbf{n}_{i,k}, \tag{2.26}$$

where $\mathbf{x}_{i,k}$ is the center of $\Gamma_{i,k}$. Taking the pressure at the interface as $\pi_{i,k}$ and the averaged pressure inside the cell i as p_i, Equation (2.26) becomes:

$$\mathbf{v}_{i,k} \approx \mathbf{A}_{i,k}\lambda_i \frac{(p_i - \pi_{i,k})\mathbf{c}_{i,k}}{|\mathbf{c}_{i,k}|^2} \cdot \mathbf{n}_{i,k} = T_{i,k}(p_i - \pi_{i,k}), \tag{2.27}$$

where $\mathbf{c}_{i,k}$ is the vector joining the center of the cell i with $\mathbf{x}_{i,k}$, and $T_{i,j}$ is the half-transmissibility of the cell i. Imposing continuity of fluxes across the faces, $v_{i,k} = -v_{k,i} = v_{ik}$ and continuity of face pressures $\pi_{i,k} = \pi_{k,i} = \pi_{ik}$, Equation (2.27) can be rewritten as:

$$T_{i,k}^{-1}\mathbf{v}_{ik} = p_i - \pi_{ik}, \qquad -T_{k,i}^{-1}\mathbf{v}_{ik} = p_k - \pi_{ik}.$$

Adding up these equations we obtain:

$$\mathbf{v}_{ik} = [T_{i,k}^{-1} + T_{k,i}^{-1}]^{-1}(p_i - p_k) = T_{ik}(p_i - p_k).$$

Equation (2.2.1) is known as Two-point flux approximation (TPFA) and T_{ik} is the transmissibility associated with the cells i and k. As in the previous discretization scheme, Equation (2.22) can be rewritten as:

$$\mathbf{T}\mathbf{p} = \mathbf{q}, \tag{2.28}$$

with the transmissibility matrix elements previously defined. In MRST, finite volumes is used as spatial discretization scheme.

2.2.2 Temporal discretization

In this section we present the backward Euler temporal discretization scheme that is implemented for the solution of compressible single-phase and incompressible two-phase examples.

Slightly compressible fluid. We can discretize the temporal part of Equation (2.11) using backward Euler discretization as follows:

$$\frac{p^{n+1} - p^n}{\Delta t} - \frac{1}{c_t\phi}\nabla \cdot \left(\lambda \cdot \nabla p^{n+1}\right) = q^{n+1}.$$

For this case, we have a transmissibility matrix resulting from the spatial discretization for the current time step and a term resulting from the temporal discretization; thus, the system to solve is:

$$\left(c_t\phi\mathbf{I} + \Delta t\mathbf{T}\right)\mathbf{p}^{n+1} = \mathbf{q}^{n+1} + c_t\phi\mathbf{p}^n. \tag{2.29}$$

Compressible fluid. For a compressible fluid flowing in a medium with constant porosity, the temporal discretization is given by (neglecting gravity terms):

$$\phi\frac{\rho(p)^{n+1}-\rho(p)^n}{\Delta t}-\nabla\cdot[\rho(p)\lambda\cdot\nabla p]^{n+1}=(q\rho(p))^{n+1}.$$

After spatial discretization, we obtain a matrix that depends on the pressure at the past and current time step, the resulting system will be:

$$\mathbf{F}(\mathbf{p}^{n+1};\mathbf{p}^n)=\mathbf{V}(\mathbf{p}^{n+1})-\mathbf{V}(\mathbf{p}^n)+\frac{\Delta t}{\phi}(\mathbf{T}(\mathbf{p}^{n+1})+\mathbf{q}^{n+1})=0. \tag{2.30}$$

This is a non linear system that can be linearized with, e.g., Newton-Raphson method described below.

Newton-Raphson (NR) method. Given a time dependent vectorial function $\mathbf{F}(\mathbf{p})$, the zeros of this function can be found using an approximation with Taylor series as follows:

$$\mathbf{F}(\mathbf{p};\mathbf{p}^0)=\mathbf{F}(\mathbf{p}^0)+(\mathbf{p}-\mathbf{p}^0)\mathbf{J}(\mathbf{p}^0)+...=\mathbf{0},$$

where $\mathbf{J}(\mathbf{p})=\frac{\partial\mathbf{F}(\mathbf{p};\mathbf{p}^0)}{\partial\mathbf{p}^k}$ is the Jacobian matrix, $\mathbf{p}^0$ is an initial guess, and $\mathbf{F}(\mathbf{p}^0)$ is the function evaluated in $\mathbf{p}^0$. Considering only the first terms, the difference between the old and new values $\delta\mathbf{p}$ is given by:

$$\delta\mathbf{p}=\mathbf{p}-\mathbf{p}^0=-\mathbf{J}^{-1}(\mathbf{p}^0)\mathbf{F}(\mathbf{p}^0).$$

With the NR method, this approximation is made recursively until $\delta\mathbf{p}$ is smaller than a given tolerance ϵ. Therefore, the linear system for the $(k+1)-th$ NR iteration becomes:

$$\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1}=-\mathbf{F}(\mathbf{p}^{k+1};\mathbf{p}^0). \tag{2.31}$$

This system is solved for $\delta\mathbf{p}^{k+1}$, and the value of $\mathbf{p}^{k+1}$ is updated with the previously computed solution $\mathbf{p}^k$,

$$\mathbf{p}^{k+1}=\mathbf{p}^k+\delta\mathbf{p}^{k+1} \tag{2.32}$$

Solution procedure for a compressible fluid. The procedure simulate flow of a compressible fluid consists of three stages:

i During the first stage, we solve Equation (2.30) for each time step.

ii Because of the nonlinearity, we use an iterative NR procedure that involves linearization at each time step, i.e. we perform a series of iterations to make $\delta\mathbf{p}=0$.

iii Solve the resulting linear system, Equation (2.31).

A summary of this procedure is presented in Algorithm 2.

Algorithm 2 Solution procedure for a compressible fluid.

```
% Time integration
for t = 0, ..., steps
    % NR iteration, finding F(p^{n+1}; p^n) = 0 for each time step.
    while δp > ϵ,    δp = −J(p^n)^{−1} F(p^n).
        % Linear iteration over i,
        while ||r_{i+1}|| > ϵ,    r_{i+1} = b(p_k) − J(p_k) δp_{k+1}
            Solve J(p_k) δp_{k+1} = b(p_k) for δp_{k+1} with an iterative method.
        end while
    end while
end for
```

For this problem, it is also necessary to specify the initial conditions, which are the pressure values of the reservoir at the beginning of the simulation.

Well model The sources are given by injection through boundary or wells, for the studied cases. In the case of wells, we used the Peaceman model introduced in Equation 2.8, for a cell (i, j, l) containing the well, this model can be discretized as follows:

$$q_{(i,j,l)} = J_{(i,j,l)}(p_{(i,j,l)} - p_{bh(i,j,l)}), \tag{2.33}$$

where $p_{(i,j,l)}$ is the reservoir pressure in the cell, and $p_{bh(i,j,l)}$ is a prescribed pressure inside the well.

Once the system is discretized, in the case of non linear systems, after linearization, the governing equations become a linear system containing a matrix that varies depending on the problem, boundary conditions and sources.

Neumann boundary conditions. Special attention is required when the system presents homogeneous Newmann boundary conditions, the following Lemma presents a required compatible condition such that Equation (2.22) has solution.

Lemma 2.2.1. Considering the elliptic Equation (2.22), let $\lambda = 1$ with homogeneous Neumann boundary conditions $\frac{\partial \mathbf{p}}{\partial \mathbf{n}} = 0$, then $\sum_{i=1}^{n} q_i = 0$.

Proof. Equation (2.22) can be rewritten as:

$$-\nabla \mathbf{p} = \mathbf{q},$$

if we integrate over all the domain and use Gauss divergence theorem we get

$$-\int_{\Omega} \nabla \mathbf{p} \quad d\mathbf{x} = -\int_{\partial\Omega} \frac{\partial \mathbf{p}}{\partial \mathbf{n}} \quad d\mathbf{s} = \int_{\Omega} \mathbf{q} \quad d\mathbf{x}$$

$$\Rightarrow \int_{\Omega} \mathbf{q} \quad d\mathbf{x} = \mathbf{0}, \qquad \text{using bc, } \frac{\partial \mathbf{p}}{\partial \mathbf{n}} = 0.$$

Or in discrete form $\sum_{i}^{n} q_i = 0$ □

Lemma 2.2.1 implies that solving an elliptic problem with homogeneous Neumann boundary conditions we have to select the sources in a way such that $\sum_{i}^{n} q_i = 0$.

Concluding remarks. In this chapter, we introduced the governing equations for simulation through porous media for single- and two-phase flow. We also presented some discretization and linearization schemes, that lead to a linear system. A summary of the resulting linear systems is presented in Table 2.1. In this work we study the acceleration of the solution of these systems.

Single phase		Two phases
Incompressible	Compressible	Incompressible
$\mathbf{T}\mathbf{p} = \mathbf{q}$	$\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1} = \mathbf{b}(\mathbf{p}^k)$	$\mathbf{T}(\mathbf{S}^n)\mathbf{p}^n = \mathbf{q}^n$

Table 2.1: *Linear systems.*

The stencil of the $\mathbf{T}$ matrix is given by:

$$-p_{i-1}T_{i-1/2} + p_{i-1}(T_{i-1/2} + T_{i+1/2}) - p_{i+1}T_{i+1/2},$$

similarly for $\mathbf{T}(S)$; thus, they are symmetric matrices.

This work is focused on the acceleration of the solution of these systems with iterative methods, in particular, with the Conjugate Gradient method. A brief introduction to these methods is presented in next chapter, together with some acceleration techniques. Furthermore, we will introduce the Proper Orthogonal Decomposition method, used for the development of the POD-based deflation method, further explained in Chapter 4.

Chapter 3

Solution methods

In the previous chapter, we developed the theory of reservoir simulation, and we discussed how some discretization and linearization techniques lead to a linear system. In this thesis, we study oil reservoirs, usually exhibiting a high contrast in permeability coefficients, resulting in an ill-conditioned system expensive to solve. These systems are commonly solved with iterative methods.

However, as the contrast increases and the systems become larger, the efficiency of the solver decreases and acceleration techniques are required. The objective of this work is to develop an efficient methodology to accelerate the solution of ill-conditioned linear systems. We introduce and develop an approach based on two state-of-the-art methods: deflation and Proper Orthogonal Decomposition (POD), more details about the methodology are discussed in Chapter 4.

This chapter is devoted to exploring the state-of-the-art iterative methods implemented to accelerate the solution of linear systems. We start by giving some theory and properties of the system under investigation. Later, we introduce the basic iterative methods, followed by the Krylov-subspace methods. For the latter kind, we put particular attention to the Conjugate Gradient method (CG), especially suited to solve linear systems containing Symmetric Positive Definite (SPD) matrices that appear in reservoir simulation.

Additionally, we present the customary stopping criteria used for iterative methods. Then, we explore the deflation methodology, along with some of their properties and common uses. Finally, we introduce the POD method.

3.1 Background

This work is focused on the solution of linear systems of the form:

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

with $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^n$. System 3.1 is considered *consistent* if it has either one or infinitely many solutions, otherwise, it is *inconsistent*. The structure of

the system matrix, $\mathbf{A}$, and right-hand side, $\mathbf{b}$, are important to determine if a system has a solution and to select an optimal solution method.

Sometimes, the system matrix $\mathbf{A}$ contains a large number of zero entries and is referred to as a *sparse* matrix. In some cases, a sparse matrix can be structured, i.e., the non-zero entries form a regular pattern. The structure may consist of a small number of diagonals or blocks with non-zero values. Discretization via finite differences on a structured grid results in a structured sparse matrix containing 3, 5, or 7 non-zero diagonals, depending on the problem's dimension.

The problems addressed in this work (see Table 2.1) consist of sparse SPD matrices, i.e., $\mathbf{A}^T = \mathbf{A}$, and $(\mathbf{Ax}, \mathbf{x}) > 0$, $\forall \mathbf{x} \in \mathbb{R}^n$, with $\mathbf{x} \neq 0$.

In this section, we give some definitions and properties of the system matrix and its respective right-hand side for a better understanding of the behavior of the iterative method.

Definition 3.1.1. The *eigenvalues*, λ_i, of an SPD matrix $\mathbf{A}$ are the n roots of the characteristic polynomial $p(\lambda) = det(\lambda \mathbf{I} - \mathbf{A})$.

The set of all the eigenvalues is known as the *spectrum*, $\sigma(\mathbf{A}) = \{\lambda_{min}, ..., \lambda_{max}\}$ and for an SPD matrix it contains only real positive *eigenvalues*, $\lambda_i \in \mathbb{R}^n$. The nonzero vectors $\mathbf{v} \in \mathbb{R}^n$, satisfying

$$\mathbf{Av} = \lambda \mathbf{v},$$

are known as the *eigenvectors* of $\mathbf{A}$.

Lemma 3.1.1. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ be arbitrary matrices. Now, the following equalities hold:

1. $\sigma(\mathbf{AB}) = \sigma(\mathbf{BA})$,
2. $\sigma(\mathbf{A} + \alpha \mathbf{I}) = \sigma(\mathbf{A}) + \alpha \sigma(\mathbf{I})$, where $\alpha \in \mathbb{R}$,
3. $\sigma(\mathbf{A}) = \sigma(\mathbf{A}^\top)$.

Proof. See [40]. □

Definition 3.1.2. The *null space* of $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the set of all solutions to the homogeneous equation $\mathbf{Ax}_n = \mathbf{0}$, or

$$\mathcal{N}(\mathbf{A}) := \{\mathbf{x}_n \in \mathbb{R}^n | \mathbf{Ax}_n = \mathbf{0}\}.$$

Definition 3.1.3. The *column space* $\mathcal{R}(\mathbf{A})$, or range of $\mathbf{A}$, is the *span* or set of all possible linear combinations of its column vectors, if $\mathbf{a}_i \in \mathbb{R}^n$ are the columns of $\mathbf{A}$,

$$\mathcal{R}(\mathbf{A}) = span\{\mathbf{a}_1, \mathbf{a}_2 ..., \mathbf{a}_n\},$$

or

$$\mathcal{R}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n, \mathbf{a}_i \in \mathbb{R}^n, \alpha_i \in \mathbb{R} | \mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{a}_i\}.$$

Definition 3.1.4. Given a set of n vectors $\{\mathbf{a}_i\}$, this set is *linearly independent* (l.i.) if the only vector, $\mathbf{c} = [c_1, ..., c_n]$, for which $\sum_{i=1}^{n} c_i \mathbf{a}_i = \mathbf{0}$, is the zero vector, $\mathbf{c} = \mathbf{0}$.

Definition 3.1.5. The *rank* or dimension of $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the number of linearly independent columns, and

$$rank(\mathcal{N}(\mathbf{A})) + rank(\mathbf{A}) = n.$$

Lemma 3.1.2. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be the system matrix of (3.1), if $\mathbf{x}_n \in \mathcal{N}(\mathbf{A})$, $\mathbf{x}_p \in \mathcal{R}(\mathbf{A})$, and $\mathbf{A}\mathbf{x}_p = \mathbf{b}$, then

$$\mathbf{x} = \mathbf{x}_n + \mathbf{x}_p, \tag{3.2}$$

is a solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Proof. Multiplying Equation (3.2) by $\mathbf{A}$ we get

$$\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}_n + \mathbf{A}\mathbf{x}_p = \mathbf{0} + \mathbf{b} = \mathbf{b}.$$

Then $\mathbf{x}$ is a solution. □

Lemma 3.1.3.

i. The linear system (3.1) is consistent if $\mathbf{b} \in \mathcal{R}(\mathbf{A})$, and

ii. the solution is unique iff $\mathcal{N}(\mathbf{A}) = \{\mathbf{0}\}$.

Proof.

i. As the system (3.1) is consistent, we can find a solution $\mathbf{x}_p = [x_1, ...x_n]$ such that $\mathbf{A}\mathbf{x}_p = \mathbf{b}$, let $\mathbf{a}_i$ be the columns of $\mathbf{A}$. Then, we can re-write the linear system as:

$$\mathbf{A}\mathbf{x}_p = \sum_i \mathbf{a}_i\mathbf{x}_i = \mathbf{b},$$

then, $\mathbf{b} \in \mathcal{R}(\mathbf{A})$.

ii.

$$\begin{aligned} & \mathbf{x}_p \text{ is unique} \\ \Leftrightarrow \quad & \mathbf{x} = \mathbf{x}_p \text{ from Lemma 3.1.2} \\ \Leftrightarrow \quad & \nexists \mathbf{x}_n \in \mathcal{N}(\mathbf{A}) \\ \Leftrightarrow \quad & \mathcal{N}(\mathbf{A}) = \{\mathbf{0}\} \end{aligned}$$

□

LU decomposition method. Once we know that the system (3.1) has a solution, it can be found with, e.g., direct methods. LU decomposition is a direct method where the matrix $\mathbf{A}$ is decomposed into a lower triangular $\mathbf{L}$, and an upper triangular matrix $\mathbf{U}$. The system is thus transformed into:

$$\mathbf{A}\mathbf{x} = \mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b}.$$

The solution of the decomposed system is obtained by forward substitution of the subsystem $\mathbf{b} = \mathbf{L}\mathbf{y}$, followed by back substitution of $\mathbf{y} = \mathbf{U}\mathbf{x}$.

Cholesky decomposition method. Symmetric matrices can be decomposed as $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. This decomposition is called Cholesky. This implies that the system (3.1) can be rewritten as:

$$\mathbf{A}\mathbf{x} = \mathbf{L}\mathbf{L}^T\mathbf{x} = \mathbf{b}. \tag{3.3}$$

This system can be solved as $\mathbf{L}\mathbf{U}$ with forward and backward substitution.

The difficulties of solving system (3.1) appear when the matrix $\mathbf{A}$ presents a large condition number $\kappa(\mathbf{A})$, given by:

$$\kappa(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})}. \tag{3.4}$$

If κ is small, direct methods can be used; nonetheless, if the smallest, λ_{min}, and largest, λ_{max}, eigenvalues are far apart, finding a solution is troublesome.

The structure of the matrix can be exploited by creating algorithms that only take into account the non-zero elements. Iterative methods are especially suited for the solution of sparse matrices as they benefit from the matrix structure, requiring less computer storage and fewer operations than direct methods. These advantages are more noticeable for large problems, as the storage and operation counts increase considerably when using direct methods.

3.2 Basic Iterative Methods (BIM)

Iterative methods are techniques created to obtain an approximate solution of linear systems. For the implementation of these methods, successive approximations are used, beginning with an initial guess solution ($\mathbf{x_0}$), and iterating over the newly computed solution ($\mathbf{x}_i$) until an accurate enough approximation to the exact solution ($\mathbf{x}$) is found.

The accuracy of the $k-th$ approximation is given by the error, $\mathbf{e}_k = \mathbf{x} - \mathbf{x}_k$, and the norm of the relative error is given by:

$$e_k = \frac{||\mathbf{x} - \mathbf{x}_k||_2}{||\mathbf{x}||_2}. \tag{3.5}$$

However, computing the error of the approximation is not possible, as it is necessary to know the exact solution. Instead, the accuracy of the method is tested by computing the residual,

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k, \tag{3.6}$$

which is related to the error as $\mathbf{A}\mathbf{e}_k = \mathbf{r}_k$.

A common stopping criterion or tolerance (ϵ) for iterative methods is the relative residual, defined as the 2-norm of the residual of the $k-th$ iteration divided by the 2-norm of the right-hand side,

$$r_k = \frac{||\mathbf{r}_k||_2}{||\mathbf{b}||_2} \leq \epsilon, \tag{3.7}$$

which is related to the relative error as follows [8]:

$$e_k = \frac{||\mathbf{x} - \mathbf{x}_k||_2}{||\mathbf{x}||_2} \leq \kappa_2(\mathbf{A}) \frac{||\mathbf{r}_k||_2}{||\mathbf{b}||_2}. \tag{3.8}$$

To obtain an iterative method, the matrix $\mathbf{A}$ is decomposed into two matrices, $\mathbf{M}$ and $\mathbf{N}$. Such that $\mathbf{A} = \mathbf{M} - \mathbf{N}$, and the original linear system (Equation 3.1) transforms into:

$$\mathbf{A}\mathbf{x} = (\mathbf{M} - \mathbf{N})\mathbf{x} = \mathbf{b}, \tag{3.9}$$

rearranging terms we obtain:

$$\mathbf{M}\mathbf{x} = \mathbf{N}\mathbf{x} + \mathbf{b} = (\mathbf{M} - \mathbf{A})\mathbf{x} + \mathbf{b}.$$

The latter system is used to perform an iterative process, finding at each iteration (k) a more accurate solution. Most of the iterative methods are derived from the following recurrence relation:

$$\mathbf{M}\mathbf{x}_k = (\mathbf{M} - \mathbf{A})\mathbf{x}_{k-1} + \mathbf{b}, \quad \text{or} \quad \mathbf{x}_k = \mathbf{M}^{-1}(\mathbf{M} - \mathbf{A})\mathbf{x}_{k-1} + \mathbf{M}^{-1}\mathbf{b}, \tag{3.10}$$

where the matrix $\mathbf{M}$ is chosen such that the sequence $\{\mathbf{x}_k\}$ is easily computed and the iterations converge rapidly.

Basic iterative methods are obtained by decomposing the system matrix as $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$. $\mathbf{D}$ being the diagonal of $\mathbf{A}$, and $-\mathbf{E}$ and $-\mathbf{F}$ are the strictly lower and upper parts. The $\mathbf{M}$ and $\mathbf{N}$ matrices are based on this decomposition. Some of the basic iterative methods are presented in Table 3.1.

Method	$\mathbf{M}$	$\mathbf{N}$	Iteration
Richardson	$\mathbf{I}$	$\mathbf{I} - \mathbf{A}$	$\mathbf{x}_{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}_k + \mathbf{b}$
Jacobi	$\mathbf{D}$	$\mathbf{E} + \mathbf{F}$	$\mathbf{x}_{k+1} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})\mathbf{x}_k + \mathbf{D}^{-1}\mathbf{b}$
Damped Jacobi	$(1/\omega)\mathbf{D}$	$\mathbf{E} + \mathbf{F}$	$\mathbf{x}_{k+1} = \omega\mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})\mathbf{x}_k + \omega\mathbf{D}^{-1}\mathbf{b}$
Gauss Seidel	$\mathbf{D} - \mathbf{E}$	$\mathbf{F}$	$\mathbf{x}_{k+1} = \mathbf{D}^{-1}(\mathbf{F}\mathbf{x}_k + \mathbf{E}\mathbf{x}_k) + \mathbf{D}^{-1}\mathbf{b}.$
Successive	$\mathbf{D} - \omega\mathbf{E}$	$(1 - \omega)\mathbf{D}+$	$\mathbf{x}_{k+1} = \omega\mathbf{D}^{-1}(\mathbf{F}\mathbf{x}_k + \mathbf{E}\mathbf{x}_{k+1})+$
Over-Relaxation		$+\omega\mathbf{F}$	$+(1 - \omega)\mathbf{x}_k + \omega\mathbf{D}^{-1}\mathbf{b}.$

Table 3.1: *Basic iterative methods.*

For some iterative methods, an optimal relaxation parameter ω can be used to further accelerate the convergence. However, this value is not always known and a complex eigenvalue analysis is required to find it. To avoid this difficulty, other methods have been developed, among others, the Krylov subspace methods have been used as an alternative [16, 41–43].

3.3 Krylov subspace methods

The recursion for iterative methods is given by Equation (3.10), and can be rewritten as:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_{k-1}) = \mathbf{x}_{k-1} + \mathbf{M}^{-1}\mathbf{r}_{k-1},$$

being $\mathbf{r}_k$ the residual defined in Equation (3.5). Taking the first approximate solution $\mathbf{x}_0$ as an arbitrary guess solution, the first iterations are given by:

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 + (\mathbf{M}^{-1}\mathbf{r}_0), \\
\mathbf{x}_2 &= \mathbf{x}_1 + (\mathbf{M}^{-1}\mathbf{r}_1) = \mathbf{x}_0 + \mathbf{M}^{-1}\mathbf{r}_0 + \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0 - \mathbf{A}\mathbf{M}^{-1}\mathbf{r}_0) \\
&= \mathbf{x}_0 + 2\mathbf{M}^{-1}\mathbf{r}_0 - \mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-1}\mathbf{r}_0, \\
&\vdots \\
\mathbf{x}_k &= \mathbf{x}_0 + 2\mathbf{M}^{-1}\mathbf{r}_0 - (\mathbf{M}^{-1}\mathbf{A})^1\mathbf{M}^{-1}\mathbf{r}_0 + \ldots - (\mathbf{M}^{-1}\mathbf{A})^{k-1}\mathbf{M}^{-1}\mathbf{r}_0.
\end{aligned}$$

Therefore, the $k-th$ iteration can be written as:

$$\mathbf{x}_k \in \mathbf{x}_0 + span\{\mathbf{M}^{-1}\mathbf{r}_0, (\mathbf{M}^{-1}\mathbf{A})\mathbf{M}^{-1}\mathbf{r}_0, \ldots, (\mathbf{M}^{-1}\mathbf{A})^{k-1}(\mathbf{M}^{-1}\mathbf{r}_0)\},$$

or,

$$\mathbf{x}_k \in \mathbf{x}_0 + \mathcal{K}_k(\mathbf{M}^{-1}\mathbf{A}; \mathbf{r}_0),$$

where the subspace $\mathcal{K}_k(\mathbf{A}; \mathbf{r}_0) := span\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\}$ is known as the Krylov subspace of dimension k corresponding to the matrix $\mathbf{A}$ and initial residual $\mathbf{r}_0$.

3.3.1 Conjugate Gradient (CG) method

The Conjugate Gradient (CG) method is a Krylov subspace method used for SPD matrices, such that the resulting error $e_k = ||\mathbf{x} - \mathbf{x}_k||_\mathbf{A}$ is minimal. To derive the method, we compute the first iterate $\mathbf{x}_1$, and we solve the minimization problem by finding an α_0 such that the error is minimal in the $\mathbf{A} - norm$, defined as:

$$||\mathbf{x}||_\mathbf{A} = \sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}. \tag{3.11}$$

If the first approximation is given by $\mathbf{x}_1 = \alpha_0\mathbf{r}_0$, its corresponding error is

$$e_1 = ||\mathbf{x} - \alpha_0\mathbf{r}_0||_\mathbf{A} = (\mathbf{x} - \alpha_0\mathbf{r}_0)^T\mathbf{A}(\mathbf{x} - \alpha_0\mathbf{r}_0) = \mathbf{x}^T\mathbf{A}\mathbf{x} - 2\alpha_0\mathbf{r}_0^T\mathbf{A}\mathbf{x} + \alpha_0^2\mathbf{r}_0^T\mathbf{r}_0,$$

$$\frac{\partial e_1}{\partial \alpha_0} = -2\mathbf{r}_0^T\mathbf{A}\mathbf{x} + 2\alpha_0\mathbf{r}_0^T\mathbf{r}_0 = 0, \qquad \Rightarrow \qquad \alpha_0 = \frac{\mathbf{r}_0^T\mathbf{A}\mathbf{x}}{\mathbf{r}_0^T\mathbf{r}_0} = \frac{\mathbf{r}_0^T\mathbf{b}}{\mathbf{r}_0^T\mathbf{r}_0}.$$

Therefore, the solution of the $k-th$ iteration will be given by $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{v}_k$, where $\mathbf{v}_i$ are the search directions. If $\mathbf{v}_i = \mathbf{r}_i$, the method is called *steepest descent*.

However, with this method, the same direction is computed more than once in some cases. To avoid the extra computations, we select a set of directions $\mathbf{p}_i$ such that they are orthogonal to the residuals, i.e., $(\mathbf{r}_k, \mathbf{p}_j) = 0$, and $\mathbf{A} - orthogonal$ to the previously computed directions, $(\mathbf{A}\mathbf{p}_k, \mathbf{p}_j) = 0$, for $k \neq j$.

For the Conjugate Gradient method, the search directions are defined as:

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k,$$

and the updated solutions are given by:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k\mathbf{p}_k \\
\Rightarrow \mathbf{r}_{k+1} &= \mathbf{b} - \mathbf{A}\mathbf{x}_{k+1} = \mathbf{b} - \mathbf{A}(\mathbf{x}_k + \alpha_k\mathbf{p}_k) = \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k
\end{aligned}$$

using $(\mathbf{r}_{k+1}, \mathbf{r}_k) = 0$, and $(\mathbf{A}\mathbf{p}_k, \mathbf{r}_k) = (\mathbf{A}\mathbf{p}_k, \mathbf{p}_k - \beta_{k-1}\mathbf{p}_{k-1}) = (\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)$, we obtain

$$\alpha_k = \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{A}\mathbf{p}_k, \mathbf{r}_k)} = \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)},$$

the $\mathbf{A}-$orthogonality of $\mathbf{p}_k$ gives

$$\mathbf{p}_{k+1}^T \mathbf{A}\mathbf{p}_k = \mathbf{r}_{k+1}^T \mathbf{A}\mathbf{p}_k + \beta_k \mathbf{p}_k^T \mathbf{A}\mathbf{p}_k = 0;$$

therefore,

$$\beta_k = -\frac{\mathbf{r}_{k+1}^T \mathbf{A}\mathbf{p}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k} = -\frac{(\mathbf{r}_{k+1}, \mathbf{A}\mathbf{p}_k)}{(\mathbf{p}_k, \mathbf{A}\mathbf{p}_k)}.$$

The implementation of this method is given in Algorithm 3, and the error is bounded by:

$$||\mathbf{x} - \mathbf{x}_{k+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}_0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} \right)^{k+1}. \tag{3.12}$$

Algorithm 3 Conjugate Gradient (CG) method, solving $\mathbf{A}\mathbf{x} = \mathbf{b}$.

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

From Equation (3.12), we note that the convergence is related to the condition number of the matrix $\mathbf{A}$, where $\kappa_2(\mathbf{A}) = 1$, is the optimal value obtained for the identity matrix. Therefore, reducing $\kappa_2(\mathbf{A})$ results in a better performance. As the condition number of an SPD matrix is related to the largest and smallest eigenvalues of the matrix (see Equation (3.4)), a reduction can be obtained by clustering the spectrum, i.e., by putting together the extreme eigenvalues, or by removing them from the method. In the next section, we introduce some acceleration techniques here implemented to improve the spectral properties of iterative methods.

3.4 Acceleration techniques

When iterative methods do not converge in a reasonable amount of time, acceleration of these methods is necessary. In this section, we present the basics on preconditioners, together with a description of the deflation method, which are the chosen acceleration methods implememnted within this study.

3.4.1 Preconditioning

Acceleration of iterative methods is done by modifying the spectrum of the system, $\sigma(\mathbf{A})$. Some preconditioning strategies provide an initial acceleration by multiplying the original system by a matrix $\mathbf{M}$ that approximates $\mathbf{A}$. After multiplying the original system by $\mathbf{M}^{-1}$, it results in the preconditioned system:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}, \tag{3.13}$$

that has the same solution as the original but is easier to solve.

The new system 3.13 clusters the spectrum and reduces κ accordingly, for which

$$\kappa(\mathbf{M}^{-1}\mathbf{A}) = \frac{\lambda_{max}(\mathbf{M}^{-1}\mathbf{A})}{\lambda_{min}(\mathbf{M}^{-1}\mathbf{A})} \qquad < \qquad \kappa(\mathbf{A}). \tag{3.14}$$

For this methods to be effective, $\mathbf{M}^{-1}$ must be cheap to compute. Some common choices of preconditioners are based on the LU factorization, $\mathbf{M} = \mathbf{L}\mathbf{U}$ which, for SPD systems, becomes the Cholesky factorization $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, a frequently used preconditioner for the CG method.

If a good preconditioner is chosen, convergence of the preconditioned system (3.14) is accelerated, and the new convergence bound is given by:

$$||\mathbf{x} - \mathbf{x}_{k+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}_0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_2(\mathbf{M}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{M}^{-1}\mathbf{A})} + 1} \right)^{k+1}. \tag{3.15}$$

If the system matrix $\mathbf{A}$ is sparse, an incomplete factorization $\mathbf{A} = \mathbf{L}_i\mathbf{L}_i^T$ can be performed, such that $\mathbf{L}_i$ is cheaper to compute. This decomposition is called Incomplete Cholesky (IC), and if the matrix $\mathbf{L}_i$ has the same nonzero entries as the system matrix, the incomplete decomposition is of order 0, $\mathbf{L}_0$. Throughout this work, we use the IC decomposition of order 0 as preconditioner.

Preconditioned Conjugate Gradient (PCG). The CG method is implemented for SPD matrices; thus, to solve preconditioned systems, an SPD preconditioning $\mathbf{M}^{-1}$ matrix is required, resulting in:

$$\widetilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \tag{3.16}$$

where

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

The pseudocode of the PCG method is presented in Algorithm 4.

Lemma 3.4.1. The preconditioned system (3.16) is consistent, i.e., (see Lemma 3.1.3),

$$\mathbf{M}^{-1/2}\mathbf{b} \in \mathcal{R}(\mathbf{M}^{-1/2}\mathbf{A}\mathbf{M}^{-1/2}).$$

Proof. Let $\mathbf{x}$ be solution of the consistent system $\mathbf{A}\mathbf{x} = \mathbf{b}$, i.e., $\mathbf{b} \in \mathcal{R}(\mathbf{A})$, if $\mathbf{a}_i$ are the columns of $\mathbf{A}$. Multiplying $\mathbf{b}$ by $\mathbf{M}^{-1/2}$ we get:

$$\mathbf{b} = \sum_i \alpha_i \mathbf{a}_i \qquad \Leftrightarrow \qquad \mathbf{M}^{-1/2}\mathbf{b} = \sum_i \alpha_i \mathbf{M}^{-1/2}\mathbf{a}_i.$$

Therefore, $\mathbf{M}^{-1/2}\mathbf{b} \in \mathcal{R}(\mathbf{M}^{-1/2}\mathbf{A})$. Thus, it exists a solution $\mathbf{x}$ of

$$\begin{aligned} & \mathbf{M}^{-1/2}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1/2}\mathbf{b} \\ \Leftrightarrow \quad & \mathbf{M}^{-1/2}\mathbf{A}\mathbf{M}^{-1/2}\mathbf{M}^{1/2}\mathbf{x} = \mathbf{M}^{-1/2}\mathbf{b}, \\ \Leftrightarrow \quad & \mathbf{M}^{-1/2}\mathbf{b} \in \mathcal{R}(\mathbf{M}^{-1/2}\mathbf{A}\mathbf{M}^{-1/2}). \end{aligned}$$

Hence, the preconditioned system (3.16) is consistent and has at least one solution $\mathbf{x}$. □

Even when the spectrum of a preconditioned system is more favorable, a few eigenvalues can, nonetheless, spoil the performance of the iterative method. These eigenvalues can be treated with, e.g., deflation techniques further explained in the next section.

Algorithm 4 Preconditioned Conjugate Gradient (PCG) method, solving $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$.

Give an initial guess $\mathbf{x}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, solve $\mathbf{M}\mathbf{y}_0 = \mathbf{r}_0$,
set $\mathbf{p}_0 = \mathbf{y}_0$, and $\mathbf{r}\mathbf{y}_0 = (\mathbf{r}_0, \mathbf{y}_0)$.
for $k = 0, ...,$ until convergence
  $\mathbf{w}_k = \mathbf{A}\mathbf{p}_k$
  $\alpha_k = \frac{\mathbf{r}\mathbf{y}_k}{(\mathbf{w}_k, \mathbf{p}_k)}$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
  $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{w}_k$
  $\mathbf{r}\mathbf{y}_{k+1} = (\mathbf{r}_{k+1}, \mathbf{y}_{k+1})$
  Solve $\mathbf{M}\mathbf{y}_{k+1} = \mathbf{r}_{k+1}$
  $\beta_k = \frac{\mathbf{r}\mathbf{y}_{k+1}}{\mathbf{r}\mathbf{y}_k}$
  $\mathbf{p}_{k+1} = \mathbf{y}_{k+1} + \beta_k \mathbf{p}_k$
end for

3.4.2 Deflation

Deflation techniques accelerate the convergence of an iterative method by annihilating the effect of extreme eigenvalues on the convergence of the method [31, 44], such that the condition number of the system is reduced.

The *deflation subspace* matrix $\mathbf{Z}$ determines the part of the spectrum that is removed, and it is usually problem dependent. In this section, we introduce the deflation method, together with some properties, and common choices of $\mathbf{Z}$.

Definition 3.4.1. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an *SPD* matrix, and $\mathbf{Z} \in \mathbb{R}^{n\times p}$ be a full rank matrix. The invertible Galerkin matrix, $\mathbf{E} \in \mathbb{R}^{p\times p}$, the correction matrix, $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and the deflation matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$ are defined as ([23, 44, 45]):

$$\mathbf{P} = \mathbf{I} - \mathbf{A}\mathbf{Q}, \qquad \mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T, \qquad \mathbf{E} = \mathbf{Z}^T\mathbf{A}\mathbf{Z}. \tag{3.17}$$

Lemma 3.4.2. The matrices in Definition 3.4.1 have the following properties [45]:

a) $\mathbf{E}^T = \mathbf{E}$.
b) $\mathbf{Q}^T = \mathbf{Q} = \mathbf{QAQ}$.
c) $\mathbf{QAZ} = \mathbf{Z}$.
d) $\mathbf{PAQ} = \mathbf{0}^{n\times n}$.
e) $\mathbf{P}^2 = \mathbf{P}$.
f) $\mathbf{AP}^T = \mathbf{PA}$.
g) $(\mathbf{I} - \mathbf{P}^T)\mathbf{x} = \mathbf{Qb}$.
h) $\mathbf{PAZ} = \mathbf{0}^{n\times m}$.
i) $\mathbf{P}^T\mathbf{Z} = \mathbf{0}^{n\times m}$.
j) $\mathbf{PA}$ is SPD.
k) $\mathbf{QA} = \mathbf{I} - \mathbf{P}^T$.

Proof.

a) $\mathbf{E}^T = (\mathbf{Z}^T\mathbf{AZ})^T = \mathbf{Z}^T\mathbf{A}^T\mathbf{Z} = \mathbf{Z}^T\mathbf{AZ} = \mathbf{E}$.

b) $\mathbf{Q}^T = (\mathbf{ZE}^{-1}\mathbf{Z}^T)^T = \mathbf{ZE}^{-T}\mathbf{Z}^T = \mathbf{ZE}^{-1}\mathbf{Z}^T = \mathbf{Q}$.

$\mathbf{QAQ} = (\mathbf{ZE}^{-1}\mathbf{Z}^T)\mathbf{A}(\mathbf{ZE}^{-1}\mathbf{Z}^T) = \mathbf{ZE}^{-1}\mathbf{EE}^{-1}\mathbf{Z}^T = \mathbf{ZE}^{-1}\mathbf{Z}^T = \mathbf{Q}$.

c) $\mathbf{QAZ} = (\mathbf{ZE}^{-1}\mathbf{Z}^T)\mathbf{AZ} = \mathbf{ZE}^{-1}\mathbf{Z}^T\mathbf{AZ} = \mathbf{ZE}^{-1}\mathbf{E} = \mathbf{Z}$.

d) $\mathbf{PAQ} = (\mathbf{I} - \mathbf{AQ})\mathbf{AQ} = \mathbf{AQ} - \mathbf{AQAQ} = \mathbf{AQ} - \mathbf{AQ} = \mathbf{0}^{n\times n}$.

e) $\mathbf{P}^2 = (\mathbf{I} - \mathbf{AQ})^2 = \mathbf{I} - 2\mathbf{AQ} + \mathbf{AQAQ} = \mathbf{I} - \mathbf{AQ} = \mathbf{P}$.

f) $\mathbf{AP}^T = \mathbf{A}(\mathbf{I} - \mathbf{AQ})^T = \mathbf{A}(\mathbf{I} - \mathbf{Q}^T\mathbf{A}^T) = \mathbf{A} + \mathbf{AQA} = (\mathbf{I} + \mathbf{AQ})\mathbf{A} = \mathbf{PA}$.

g) $(\mathbf{I} - \mathbf{P}^T)\mathbf{x} = (\mathbf{I} - \mathbf{I} + \mathbf{Q}^T\mathbf{A}^T)\mathbf{x} = \mathbf{QAx} = \mathbf{Qb}$.

h) $\mathbf{PAZ} = (\mathbf{I} - \mathbf{AQ})\mathbf{AZ} = \mathbf{AZ} - \mathbf{AQAZ} = \mathbf{AZ} - \mathbf{AZ} = \mathbf{0}^{n\times m}$.

i) $\mathbf{P}^T\mathbf{Z} = (\mathbf{I} - \mathbf{AQ})^T\mathbf{Z} = (\mathbf{I} - \mathbf{Q}^T\mathbf{A}^T)\mathbf{Z} = \mathbf{Z} - \mathbf{QAZ} = \mathbf{Z} - \mathbf{Z} = \mathbf{0}^{n\times m}$.

j) Symmetry, f).

Positive definite, $(\mathbf{PAx}, \mathbf{x}) \geq 0, \quad \forall \mathbf{x}$, we know that $(\mathbf{Ax}, \mathbf{x}) \geq 0 \quad \mathbf{x}$, if we choose $\mathbf{x} = \mathbf{P}^T\mathbf{y}$

$$\begin{aligned}(\mathbf{Ax}, \mathbf{x}) &= (\mathbf{P}^T\mathbf{y})^T\mathbf{AP}^T\mathbf{y} = \mathbf{y}^T\mathbf{PAP}^T\mathbf{y} = \\ &= \mathbf{y}^T\mathbf{P}^2\mathbf{AP}^T\mathbf{y} = \mathbf{y}^T\mathbf{P}(\mathbf{PA})\mathbf{P}^T\mathbf{y} = \mathbf{y}^T\mathbf{P}(\mathbf{A} - \mathbf{AQA})\mathbf{P}^T\mathbf{y} = && \text{e)} \\ &= \mathbf{y}^T\mathbf{PAy} - \mathbf{y}^T(\mathbf{PAQ})\mathbf{AP}^T\mathbf{y} = \mathbf{y}^T\mathbf{PAy} \geq 0 && \text{d)}\end{aligned}$$

Therefore $\mathbf{PA}$ is SPD.

k) $\mathbf{P}^T = (\mathbf{I} - \mathbf{AQ})^T = \mathbf{I}^T - (\mathbf{AQ})^T = \mathbf{I} - \mathbf{Q}^T\mathbf{A}^T = \mathbf{I} - \mathbf{QA}$

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

□

The columns of $\mathbf{Z}$, $(\mathbf{z}_i)$, are the *deflation* or *projection* vectors, and are chosen in such a way that $\mathbf{E}$ is nonsingular, i.e., invertible. The following Lemma presents a property of the matrices $\mathbf{A}$ and $\mathbf{Z}$, so that $\mathbf{E}$ is non singular.

Lemma 3.4.3. Let $\mathbf{A}$, $\mathbf{Z}$, and $\mathbf{E}$ be as given in Definition 3.4.1. If $\mathcal{N}(\mathbf{A}) \nsubseteq \mathcal{R}(\mathbf{Z})$, then, $\mathbf{E}$ is nonsingular.

Proof. Let $\mathbf{y} \neq \mathbf{0} \in \mathbb{R}^n$, $\mathbf{x} = \mathbf{Z}\mathbf{y}$, such that

$$\mathbf{Z}\mathbf{y} \notin \mathcal{N}(\mathbf{A}),$$

as $\mathbf{A}$ is SPD, then,

$$0 < \mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{y}^T\mathbf{Z}^T\mathbf{A}\mathbf{Z}\mathbf{y} = \mathbf{y}^T\mathbf{E}\mathbf{y} \quad \Rightarrow \quad \mathbf{y}^T\mathbf{E}\mathbf{y} > 0,$$

as $\mathbf{y} \neq \mathbf{0}$, then $\mathbf{E}\mathbf{y} \neq \mathbf{0}$, then $\mathbf{E}$ is nonsingular. □

Lemma 3.4.4. Let $\mathbf{A}$, $\mathbf{Z}$, and $\mathbf{P}$ be as given in Definition 3.4.1, and $\mathbf{y} \in \mathcal{R}(\mathbf{Z})$ then

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

ii. $\mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{A}\mathbf{P})$.

Proof.

i. As $\mathbf{y} \in \mathcal{R}(\mathbf{Z})$, we can write it as $\mathbf{y} = \sum_i w_i\mathbf{y}_i = \mathbf{Z}\mathbf{w}$, using Lemma 3.4.2 i), we have $\mathbf{P}^T\mathbf{y} = \mathbf{P}^T\mathbf{Z}\mathbf{w} = \mathbf{0}$.

ii. From Lemma 3.4.2 h), we have

$$\mathbf{P}\mathbf{A}\mathbf{y} = \mathbf{P}\mathbf{A}\mathbf{Z}\mathbf{w} = \mathbf{0}.$$

then $\mathbf{y} \in \mathcal{N}(\mathbf{A}\mathbf{P})$; therefore $\mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{A}\mathbf{P})$

□

Lemma 3.4.5. Let $\mathbf{P}$ be the projection matrix as in Definition 3.4.1, the solution o the linear system (3.1) is given by:

$$\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^T\hat{\mathbf{x}}, \tag{3.18}$$

where $\hat{\mathbf{x}}$ is a solution to the deflated system

$$\mathbf{P}\mathbf{b} = \mathbf{P}\mathbf{A}\hat{\mathbf{x}}. \tag{3.19}$$

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

$$\begin{aligned} \mathbf{x} =& \mathbf{I}\mathbf{x} - \mathbf{P}^T\mathbf{x} + \mathbf{P}^T\mathbf{x} \\ =& (\mathbf{I} - \mathbf{P}^T)\mathbf{x} + \mathbf{P}^T\mathbf{x} \\ =& \mathbf{Q}\mathbf{b} + \mathbf{P}^T\mathbf{x} \qquad \text{Lemma 3.4.2 g)} \end{aligned} \tag{3.20}$$

If we multiply system (3.20) by $\mathbf{A}$ we obtain:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{A}(\mathbf{I} - \mathbf{P}^T)\mathbf{b} + \mathbf{A}\mathbf{P}^T\mathbf{x}, \\ \mathbf{A}\mathbf{x} &= \mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{A}\mathbf{P}^T\mathbf{x}, && \text{Lemma 3.4.2 g)} \\ \mathbf{b} &= \mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{P}\mathbf{A}\mathbf{x}, && \text{Lemma 3.4.2 f)} \\ \mathbf{P}\mathbf{b} &= \mathbf{P}\mathbf{A}\mathbf{Q}\mathbf{b} + \mathbf{P}^2\mathbf{A}\mathbf{x}, && \text{Multiplying by } \mathbf{P} \\ \mathbf{P}\mathbf{b} &= \mathbf{P}\mathbf{A}\mathbf{x} && \text{Lemma 3.4.2 d)} \end{aligned}$$

The resulting system $\mathbf{Pb} = \mathbf{PAx}$ is singular; according to Lemma 3.1.2 it can be written as:

$$\mathbf{x} = \mathbf{x}_n + \mathbf{x},$$

where $\mathbf{x}_n \in \mathcal{R}(\mathbf{Z}) \subset \mathcal{N}(\mathbf{PA}) \neq \{\mathbf{0}\}$. Therefore, it is not the same solution $\mathbf{x}$ of Equation (3.1) and

$$\mathbf{P}^T\mathbf{x} = \mathbf{P}^T\mathbf{x}_n + \mathbf{P}^T\mathbf{x} = \mathbf{P}^T\mathbf{x} \quad \text{see Lemma 3.4.4 [i].}$$

Then, Equation (3.20) transforms into:

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

□

The new solution $\hat{\mathbf{x}}$ is known as the *deflated solution* of the *deflated system* (Equation (3.19)).

Lemma 3.4.6. The deflated system (3.19) is consistent, i.e., $\mathbf{Pb} \in \mathcal{R}(\mathbf{PA})$, see Lemma 3.1.3.

Proof. Let $\mathbf{x}$ be solution of the consistent system $\mathbf{Ax} = \mathbf{b}$, i.e., $\mathbf{b} \in \mathcal{R}(\mathbf{A})$, if $\mathbf{a}_i$ are the columns of $\mathbf{A}$, $\mathbf{b}$ can be expressed as

$$\mathbf{b} = \sum_i \alpha_i \mathbf{a}_i$$
$$\mathbf{Pb} = \sum_i \alpha_i \mathbf{Pa}_i$$

Then $\mathbf{Pb} \in \mathcal{R}(\mathbf{PA})$, i.e., system (3.19) is consistent and has at least one solution $\hat{\mathbf{x}}$. □

As system (3.19) is consistent, and from Lemma 3.4.2 j) it is SPD, therefore CG can be used to solve it. In the next two section we introduce the deflated CG and the deflated and preconditioned CG methods.

3.4.3 Deflated CG method

Deflation techniques can be combined with iterative methods, for the acceleration of the solution process. In particular, they can be combined with the CG method for the solution of Equation (3.1). The solution of a linear system with the deflated CG method requires the solution of the deflated system (Equation (3.19)):

$$\mathbf{PA}\hat{\mathbf{x}} = \mathbf{Pb}.$$

for a deflated solution $\hat{\mathbf{x}}$. The solution $\mathbf{x}$ of the original system is then obtained from Equation (3.18):

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

This procedure is presented in Algorithm 5. The convergence of the deflated CG

Algorithm 5 Deflated CG method, solving $\mathbf{Ax} = \mathbf{b}$.

Given an initial guess $\mathbf{x}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, and set $\hat{\mathbf{r}}_0 = \mathbf{Pr}_0$
$\mathbf{p}_0 = \hat{\mathbf{r}}_0$.
for $k = 0, ...,$ until convergence
$\hat{\mathbf{w}}_k = \mathbf{PAp}_k$
$\alpha_k = \frac{(\hat{\mathbf{r}}_k, \hat{\mathbf{r}}_k)}{(\hat{\mathbf{w}}_k, \mathbf{p}_k)}$
$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k$
$\hat{\mathbf{r}}_{k+1} = \hat{\mathbf{r}}_k - \alpha_k \hat{\mathbf{w}}_k$
$\beta_k = \frac{(\hat{\mathbf{r}}_{k+1}, \hat{\mathbf{r}}_{k+1})}{(\hat{\mathbf{r}}_k, \hat{\mathbf{r}}_k)}$
$\mathbf{p}_{k+1} = \hat{\mathbf{r}}_{k+1} + \beta_k \mathbf{p}_k$
end for
$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T \hat{\mathbf{x}}_{k+1}$.

method is given in Equation (3.21),

$$||\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}||_{\mathbf{PA}} \leq 2||\hat{\mathbf{x}} - \hat{\mathbf{x}}_0||_{\mathbf{PA}} \left(\frac{\sqrt{\kappa_{eff}(\mathbf{PA})} - 1}{\sqrt{\kappa_{eff}(\mathbf{PA})} + 1} \right)^{k+1}. \tag{3.21}$$

As mentioned before, some of the eigenvalues of the matrix $\mathbf{A}$ are shifted to zero with deflation (see Lemma 4.1.1), which implies that the smallest eigenvalue is zero leading to a condition number $\kappa_2(\mathbf{PA})$ very large. To prevent this behavior, the effective condition number is used instead, and it is defined as the ratio between the largest eigenvalue and the smallest non-zero eigenvalue,

$$\kappa_{eff}(\mathbf{PA}) = \frac{\lambda_{max}(\mathbf{PA})}{\lambda_{min}(\mathbf{PA}) \neq 0}. \tag{3.22}$$

Lemma 3.4.7. Let $\mathbf{P}$, and $\mathbf{A}$, be given as in Definition 3.4.1, then

$$||\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}||_{\mathbf{PA}} = ||\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}||_{\mathbf{A}}.$$

Proof.

$$\begin{aligned}
||\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}||_{\mathbf{PA}} &= (\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1})^T \mathbf{PA}(\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}) \\
&= (\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1})^T \mathbf{PAP}^T(\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}) && \text{Lemma 3.4.2 e) \& f)} \\
&= (\mathbf{P}^T\hat{\mathbf{x}} - \mathbf{P}^T\hat{\mathbf{x}}_{k+1})^T \mathbf{A}(\mathbf{P}^T\hat{\mathbf{x}} - \mathbf{P}^T\hat{\mathbf{x}}_{k+1}) \\
&= (\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1})^T \mathbf{A}(\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}) && \mathbf{PA}(\mathbf{P}^T\hat{\mathbf{x}}) = \mathbf{P}^2\mathbf{A}\hat{\mathbf{x}} = \mathbf{PA}\hat{\mathbf{x}} \\
&= ||\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}||_{\mathbf{A}}
\end{aligned}$$

□

Equation (3.21) requires the $\mathbf{PA}-norm$ of the error; however, Lemma 3.4.7 shows that it is the same as the $A-norm$ of the error. Taking Lemma 3.4.7 into account, Equation 3.21 can be rewritten as:

$$||\hat{\mathbf{x}} - \hat{\mathbf{x}}_{k+1}||_{\mathbf{A}} \leq 2||\hat{\mathbf{x}} - \hat{\mathbf{x}}_0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_{eff}(\mathbf{PA})} - 1}{\sqrt{\kappa_{eff}(\mathbf{PA})} + 1} \right)^{k+1}. \tag{3.23}$$

Furthermore, as we remove some of the smallest eigenvalues, the effective condition number of the deflated system is smaller than the condition number of the original system, i.e.,

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

which can lead to faster convergence rates.

3.4.4 Deflated PCG method

On one hand, preconditioning techniques help to cluster the spectrum of the system matrix, but it is possible that some small eigenvalues hamper the performance of the method. On the other hand, deflation techniques help to abate the influence of some of the bad eigenvalues hampering the convergence of the iterative method. Therefore, combining these two methodologies provides a way to accelerate the solution of linear systems efficiently.

In particular, a deflated preconditioned variant of the CG method can be implemented, using an SPD matrix $\mathbf{M}^{-1}$ and applying the deflation procedure introduced before. For a given preconditioned deflation subspace matrix $\widetilde{\mathbf{Z}} \in \mathbb{R}^{n \times k}$, the resulting preconditioned deflated system is given by:

$$\widetilde{\mathbf{P}}\widetilde{\mathbf{A}}\hat{\tilde{\mathbf{x}}} = \widetilde{\mathbf{P}}\tilde{\mathbf{b}}, \tag{3.24}$$

where

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

and

$$\widetilde{\mathbf{P}} = \mathbf{I} - \widetilde{\mathbf{A}}\widetilde{\mathbf{Q}}, \qquad \widetilde{\mathbf{Q}} = \widetilde{\mathbf{Z}}\widetilde{\mathbf{E}}^{-1}\widetilde{\mathbf{Z}}^{T}. \qquad \widetilde{\mathbf{E}} = \widetilde{\mathbf{Z}}^{T}\widetilde{\mathbf{A}}\widetilde{\mathbf{Z}}, \tag{3.25}$$

Algorithm 6 Deflated PCG (DPCG) method,
solving $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$.

Give an initial guess $\mathbf{x}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, and $\hat{\mathbf{r}}_0 = \mathbf{P}\mathbf{r}_0$,
solve $\mathbf{M}\mathbf{y}_0 = \hat{\mathbf{r}}_0$, set $\mathbf{p}_0 = \mathbf{y}_0$.
for $k = 0, ...,$ until convergence
  $\hat{\mathbf{w}}_k = \mathbf{P}\mathbf{A}\mathbf{p}_k$
  $\alpha_k = \frac{(\hat{\mathbf{r}}_k, \mathbf{y}_k)}{(\hat{\mathbf{w}}_k, \mathbf{p}_k)}$
  $\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k \mathbf{p}_k$
  $\hat{\mathbf{r}}_{k+1} = \hat{\mathbf{r}}_k - \alpha_k \hat{\mathbf{w}}_k$
  Solve $\mathbf{M}\mathbf{y}_{k+1} = \hat{\mathbf{r}}_{k+1}$
  $\beta_k = \frac{(\hat{\mathbf{r}}_{k+1}, \mathbf{y}_{k+1})}{(\hat{\mathbf{r}}_k, \mathbf{y}_k)}$
  $\mathbf{p}_{k+1} = \mathbf{y}_{k+1} + \beta_k \mathbf{p}_k$
end for
$\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^T\hat{\mathbf{x}}$

In practice (see Algorithm 6),

$$\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{P}\mathbf{b} \tag{3.26}$$

is computed and the error is bounded by:

$$||\mathbf{x} - \mathbf{x}_{k+1}||_{\mathbf{A}} \leq 2||\mathbf{x} - \mathbf{x}_0||_{\mathbf{A}} \left(\frac{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})} - 1}{\sqrt{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})} + 1} \right)^{k+1},$$

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

Choices of deflation vectors

As mentioned before, the deflation vectors contained in the matrix $\mathbf{Z}$ determine the performance of the deflation procedure. In this section, we introduce some common choices of deflation vectors. More details about the deflation vectors used in this work are presented in Chapter 4.

A good selection of the deflation vectors is usually problem-dependent, and available information of the system is, in general, used to construct these vectors. Most of the techniques used to select deflation vectors are based on eigenvectors or approximated eigenvectors, recycling vectors [38], subdomain deflation vectors [32] or multigrid and multilevel based deflation vectors [23, 25]. A summary of some of these techniques is given below.

Eigenvectors as deflation vectors If the matrix $\mathbf{Z}$ contains eigenvectors corresponding to the most unfavorable eigenvalues, these eigenvalues are removed from the deflated system and convergence of the iterative method is achieved faster, this behavior is illustrated in Lemma 4.1.1. However, the eigenvalues are usually unknown, and it is costly to obtain them. If it is not possible to obtain the eigenvalues of $\mathbf{A}$, a good choice should efficiently approximate the eigenvectors for good applicability of the method.

Recycling deflation vectors. A set of solutions previously obtained is reused to build the deflation-subspace matrix [38]. The vectors, also known as *snapshots* can be p solution vectors of the linear system, $\mathbf{x}_i$ with different right-hand sides or of at various time steps. The matrix $\mathbf{Z}$ containing these solutions is given by

$$\mathbf{Z} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p].$$

If the selected snapshots contain most of the system information, the solution of the deflated system is achieved in one iteration. This behavior is shown in Chapter 4, Lemma 4.1.2 and Lemma 4.1.3.

Subdomain deflation vectors. For this approach, the domain is divided into several subdomains, using domain decomposition techniques or taking into account the properties of the problem. For each subdomain, there is a deflation vector that contains ones for each cell inside the subdomain and zeros for cells outside [32].

Multigrid and Multilevel deflation vectors. For the multigrid and multilevel methods, the prolongation and restriction matrices are used to pass from one level or grid to another. These matrices can also be used as the deflation-subspace matrices $\mathbf{Z}$ [23], being the columns of these matrices the deflation vectors.

In this work, we propose the use of POD techniques as a way of selecting the deflation vectors, in the next section we introduce the POD methodology.

3.5 Proper Orthogonal Decomposition (POD) method

Proper Orthogonal Decomposition (POD) is a Model Order Reduction (MOR) method, where a high-order model is projected onto a space spanned by a small set of orthonormal basis vectors $\Psi = [\psi_1\ \psi_2\ ..\ \psi_p]$, $\Psi \in \mathbb{R}^{n\times l}$. The basis vectors $\psi_i \in \mathbb{R}^n$ are computed from a set of 'snapshots' $\{\mathbf{x}_i\}_{i=1}^m$, obtained by simulation or experiments [12]. The vectors $\{\psi_j\}_{j=1}^p$ are p eigenvectors corresponding to the p largest eigenvalues $\{\lambda_j\}_{j=1}^p$ of the data correlation matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$,

$$\mathbf{R} := \frac{1}{m}\mathbf{X}\mathbf{X}^T \equiv \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T, \qquad \mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_m]. \tag{3.27}$$

In some cases, the covariance matrix $\overline{\mathbf{R}}$ is used instead of $\mathbf{R}$, this matrix is defined as

$$\overline{\mathbf{R}} := \frac{1}{m-1}\sum_{i=1}^{m}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T, \qquad \overline{\mathbf{X}} := [\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2, ...\overline{\mathbf{x}}_m], \tag{3.28}$$

where $\overline{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$ is the mean of the snapshots. In this work, we also normalize the snapshots, i.e., we use the following relation:

$$\overline{\mathbf{R}} := \frac{1}{m-1}\sum_{i=1}^{m}\frac{\mathbf{x}_i - \overline{\mathbf{x}}}{||\mathbf{x}_i - \overline{\mathbf{x}}||_2}\frac{(\mathbf{x}_i - \overline{\mathbf{x}})^T}{||(\mathbf{x}_i - \overline{\mathbf{x}})^T||_2}. \tag{3.29}$$

If the system is large, the matrix $\overline{\mathbf{R}}$ is also large, and to compute the eigenvalues can be costly. However, it is not necessary to compute the eigenvalues of $\overline{\mathbf{R}} = \overline{\mathbf{X}}\overline{\mathbf{X}}^T \in \mathbb{R}^{n\times n}$, but instead, it is possible to compute the eigenvalues of the much smaller matrix $\overline{\mathbf{X}}^T\overline{\mathbf{X}} \in \mathbb{R}^{m\times m}$, $m << n$. To do so, we perform the Singular Value Decomposition (SVD) of $\overline{\mathbf{R}}^T = \mathbf{V}\mathbf{D}\mathbf{V}^T$. Here $\mathbf{V} \in \mathbb{R}^{n\times m}$ are the eigenvalues of $\overline{\mathbf{R}}^T$, and $diag(\mathbf{D})$ are the corresponding eigenvalues. These latter ones are the same as the eigenvalues of $\overline{\mathbf{R}}$, and the eigenvectors of $\overline{\mathbf{R}}$ can be obtained from [46]:

$$\mathbf{U} = \overline{\mathbf{X}}\mathbf{V}(\Lambda^T)^{\frac{1}{2}} \in \mathbb{R}^m,$$

where $\Lambda = [\lambda_1, ..., \lambda_m] \cdot \mathbf{I} \in \mathbb{R}^{m\times m}$. Once the basis is computed, the high dimensional variable $\mathbf{x} \in \mathbb{R}^n$ is approximated by a linear combination of p orthonormal basis vectors [11]:

$$\mathbf{x} \approx \sum_{i=1}^{p} c_i\psi_i. \tag{3.30}$$

The p eigenvectors are chosen such that they contain almost all the variability of the snapshots. Usually, they are the eigenvectors corresponding to the maximal number of eigenvalues satisfying [12]:

$$\frac{\sum_{j=1}^{p} \lambda_j}{\sum_{j=1}^{m} \lambda_j} \leq \alpha, \qquad 0 < \alpha \leq 1, \tag{3.31}$$

with α close to 1. The eigenvalues λ_j are ordered from large to small with λ_1 being the largest eigenvalue of $\overline{\mathbf{R}}$.

Once the basis Ψ is obtained, the linear system from Equation (3.1) is projected onto the subspace spanned by the basis [11] as follows:

$$\Psi^T \mathbf{A} \Psi \mathbf{z} = \Phi^T \mathbf{b},$$

leading to the reduced model:

$$\mathbf{A}_r \mathbf{z} = \mathbf{b}_r \qquad \mathbf{A}_r \in \mathbb{R}^{p \times p}, \mathbf{b}_r \in \mathbb{R}^p.$$

The reduced model is dense; however, it is much smaller than the original system and can be solved efficiently with direct methods.

Concluding remarks. Acceleration of iterative methods is addressed throughout this thesis. In this chapter, we introduced some of these methods and acceleration techniques, together with their properties algorithms. Additionally, we introduced the POD method, that we combine with deflation techniques and for the acceleration of iterative methods. Details about this methodology are presented in the next chapter.

Chapter 4

Proposed methodology: POD-based deflation method

In Chapter 3, we introduced the deflation methodology, for which a good performance strongly depends on the selection of the deflation vectors, required to construct the deflation subspace matrix $\mathbf{Z}$, necessary for the implementation of the deflation methodology.

In this chapter, we introduce some theoretical results to give an insight into the performance of the method. In particular, we perform a spectral analysis of the deflation method when using eigenvectors of the system matrix as deflation vectors, and we make a complexity analysis of the algorithms.

Furthermore, we introduce a new way of selecting the deflation vectors by using a POD basis computed from previously obtained system information. With this choice, we expect to obtain the smallest number of deflation vectors containing the most relevant information, reducing the overall costs of the method.

4.1 Deflation vectors.

Acceleration of iterative methods can be achieved with preconditioning techniques, that cluster the spectrum of the system matrix $\mathbf{A}$. However, the preconditioned system can still contain some extreme eigenvalues hampering the convergence rate. Deflation techniques aim to project these eigenvalues of $\mathbf{A}$ to zero, accelerating in this way the iterative solver. To this end, a set of projection vectors has to be selected; they can be, among others, the eigenvectors associated with the extreme eigenvalues or recycled vectors.

In Section 4.1.1, we introduce a lemma that analyses the behavior of the method when using eigenvectors as deflation vectors. However, since computing the eigenvalues and eigenvectors is usually expensive, new ways of selecting deflation vectors are required.

Recycled vectors is another common choice of selecting deflation vectors. If essential system information is stored in a set of snapshots, the use of these snapshots leads

to an acceleration of the iterative method. In particular, if all the system information is contained in a set of linearly independent snapshots, convergence is achieved in one iteration. In Section 4.1.2 we present two lemmas that show this behaviour (Lemma 4.1.2 and Lemma 4.1.3).Furthermore, some lemmas are presented, that exhibit the correct selection of snapshots depending on the boundary conditions and the required accuracy.

A proper selection of snapshots as deflation vectors leads to a good performance of the deflation methodology. However, selecting these snapshots is usually not straightforward. The main goal of this work is to develop a methodology to obtain a set of deflation vectors containing the most relevant system information.

In Section 4.2, we introduce the POD-based deflation method, developed throughout this work, for an effective collection of essential system information on a POD basis for later use as deflation vectors.

The computation of this basis requires the collection of a set of snapshots, made with two approaches: a moving window, that computes snapshots 'on-the-fly' to solve the following time steps; and a training phase, that computes the POD basis with an initial simulation, and it is used to solve similar problems. This methodology is further explained later in this chapter.

4.1.1 Eigenvectors as deflation vectors

Selecting the eigenvectors of the system matrix $\mathbf{A}$ as deflation vectors, results in setting the corresponding eigenvalues to zero, removing them from the deflated system. This procedure improves the conditioning of the system and accelerates the convergence of the iterative solver; as presented in Theorem 4.1.1.

Theorem 4.1.1. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{E} \in \mathbb{R}^{p\times p}$, $\mathbf{Q} \in \mathbb{R}^{n\times n}$, and $\mathbf{P} \in \mathbb{R}^{n\times n}$ be given as in Definition 3.4.1. Let $\mathbf{A}$ be a symmetric matrix with spectrum $\sigma(\mathbf{A}) = \{\lambda_1, ..., \lambda_n\}$, and eigenvectors $\Sigma(\mathbf{A}) = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$, with $\mathbf{v}_i^T\mathbf{v}_j = \delta_{ij}$, such that

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

If the deflation sub-space matrix contains the eigenvectors of $\mathbf{A}$, $\mathbf{Z} = [\mathbf{v}_1, ..., \mathbf{v}_p]$, the following properties hold:

a) $\mathbf{A}\mathbf{Z} = \mathbf{Z}\Lambda$, and $\mathbf{A}\mathbf{Z}^c = \mathbf{Z}^c\Lambda^c$,

b) $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}$,

c) $\mathbf{Z}^T\mathbf{v}_j = \mathbf{e}_j$, and

$$\mathbf{Z}\mathbf{Z}^T\mathbf{v}_j = \begin{cases} 0, & \mathbf{v}_j \notin \mathbf{Z}, \\ \mathbf{v}_j, & \mathbf{v}_j \in \mathbf{Z}, \end{cases}$$

d) $\mathbf{E} = \Lambda$,

e) $\mathbf{E}^{-1} = \Lambda^{-1}$, with $\Lambda^{-1} = diag(\lambda_1^{-1}, ..., \lambda_p^{-1})$,

f) $\mathbf{Q} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^T$,

g)
$$\mathbf{Q}\mathbf{v}_j = \begin{cases} \lambda_j^{-1}\mathbf{v}_j, & \mathbf{v}_j \in \mathbf{Z}, \\ 0, & \mathbf{v}_j \notin \mathbf{Z}, \end{cases}$$

h) $\mathbf{QZ} = \mathbf{Z}\Lambda^{-1}$, and $\mathbf{QZ}^c = \mathbf{0}$,

i) $\mathbf{PZ} = \mathbf{0}$, and $\mathbf{PZ}^c = \mathbf{Z}^c$,

j) $\mathbf{P}^T\mathbf{Z} = \mathbf{0}$, and $\mathbf{P}^T\mathbf{Z}^c = \mathbf{Z}^c$,

where we define the complement of $\mathbf{Z}$ as the matrix containing the eigenvalues of $\mathbf{A}$ not in $\mathbf{Z}$, i.e., $\mathbf{Z^c} = [\mathbf{v}_{p+1} \dots \mathbf{v}_n]$, $\Lambda = diag(\lambda_1, ..., \lambda_p)$, $\Lambda^{-1} = diag(\lambda_1^{-1}, ..., \lambda_p^{-1})$, $\Lambda^c = diag(\lambda_{p+1}, ..., \lambda_n)$, and $\mathbf{e}_j \in \mathbb{R}^n$ is the j^{th} unit vector.

Proof.

a)
$$\begin{aligned} \mathbf{AZ} &= \mathbf{A}[\mathbf{v}_1, ..., \mathbf{v}_p] = [\mathbf{A}\mathbf{v}_1, ..., \mathbf{A}\mathbf{v}_p] \\ &= [\lambda_1\mathbf{v}_1, ..., \lambda_p\mathbf{v}_p] = [\mathbf{v}_1, ..., \mathbf{v}_p]\Lambda = \mathbf{Z}\Lambda. \end{aligned}$$

$$\begin{aligned} \mathbf{AZ}^c &= \mathbf{A}[\mathbf{v}_{p+1}, ..., \mathbf{v}_n] = [\mathbf{A}\mathbf{v}_{p+1}, ..., \mathbf{A}\mathbf{v}_n] \\ &= [\lambda_{p+1}\mathbf{v}_{p+1}, ..., \lambda_n\mathbf{v}_n] = [\mathbf{v}_{p+1}, ..., \mathbf{v}_n]\Lambda^c = \mathbf{Z}^c\Lambda^c. \end{aligned}$$

b)
$$\mathbf{Z}^T\mathbf{Z} = [\mathbf{v}_1, ..., \mathbf{v}_p]^T[\mathbf{v}_1, ..., \mathbf{v}_p] = [\mathbf{v}_1^T\mathbf{v}_1, ..., \mathbf{v}_p^T\mathbf{v}_p] = \mathbf{I}.$$

c)
$$\begin{aligned} & \mathbf{Z}^T\mathbf{v}_j = [\mathbf{v}_1, ..., \mathbf{v}_p]^T\mathbf{v}_j = \mathbf{e}_j, \\ \Rightarrow \quad & \mathbf{Z}\mathbf{Z}^T\mathbf{v}_j = \mathbf{Z}\mathbf{e}_j = [\mathbf{v}_1, ..., \mathbf{v}_p]\mathbf{e}_j = \begin{cases} 0, & \mathbf{v}_j \notin \mathbf{Z}, \\ \mathbf{v}_j, & \mathbf{v}_j \in \mathbf{Z}. \end{cases} \end{aligned}$$

d) Using a) and b)
$$\begin{aligned} & \mathbf{E} = \mathbf{Z}^T\mathbf{A}\mathbf{Z} = \mathbf{Z}^T\mathbf{Z}\Lambda = \Lambda, \\ \Rightarrow \quad & \mathbf{E} = \Lambda. \end{aligned}$$

e) From d), $\mathbf{E} = \Lambda$, then
$$\mathbf{E}^{-1} = (diag(\lambda_1, ..., \lambda_p))^{-1} = diag(\lambda_1^{-1}, ..., \lambda_p^{-1}) = \Lambda^{-1}.$$

f) From Definition 3.4.1, e), and Lemma 3.4.2 b), $\mathbf{Q}$ is given by:
$$\mathbf{Q} = \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^T.$$

g) From f) and c) we have:

$$\mathbf{Q}\mathbf{v}_j = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^T\mathbf{v}_j = \mathbf{Z}\Lambda^{-1}\mathbf{e}_j = \begin{cases} \lambda_j^{-1}\mathbf{v}_j, & \mathbf{v}_j \in \mathbf{Z}, \\ \mathbf{0}, & \mathbf{v}_j \notin \mathbf{Z}. \end{cases}$$

h) From g) we have:

$$\mathbf{Q}\mathbf{Z} = [\mathbf{Q}\mathbf{v}_1, ..., \mathbf{Q}\mathbf{v}_p] = [\lambda_1^{-1}\mathbf{v}_1, ..., \lambda_p^{-1}\mathbf{v}_p] = \mathbf{Z}\Lambda^{-1}.$$

$$\mathbf{Q}\mathbf{Z}^c = [\mathbf{Q}\mathbf{v}_{p+1}, ..., \mathbf{Q}\mathbf{v}_n] = [\mathbf{0}, ..., \mathbf{0}] = \mathbf{0}.$$

i) Using Definition of $\mathbf{P}$, h), and a), we have

$$\mathbf{P}\mathbf{Z} = (\mathbf{I} - \mathbf{A}\mathbf{Q})\mathbf{Z} = \mathbf{Z} - \mathbf{A}\mathbf{Q}\mathbf{Z} = \mathbf{Z} - \mathbf{A}\mathbf{Z}\Lambda^{-1} = \mathbf{Z} - \mathbf{Z}\Lambda\Lambda^{-1} = \mathbf{Z} - \mathbf{Z} = \mathbf{0}.$$

$$\mathbf{P}\mathbf{Z}^c = (\mathbf{I} - \mathbf{A}\mathbf{Q})\mathbf{Z}^c = \mathbf{Z}^c - \mathbf{A}\mathbf{Q}\mathbf{Z}^c = \mathbf{Z}^c - \mathbf{0} = \mathbf{Z}^c.$$

j) Using h) and Lemma 3.4.2 k) and c) we have

$$\mathbf{P}^T\mathbf{Z} = (\mathbf{I} - \mathbf{Q}\mathbf{A})\mathbf{Z} = \mathbf{Z} - \mathbf{Q}\mathbf{A}\mathbf{Z} = \mathbf{Z} - \mathbf{Z} = \mathbf{0}.$$

$$\mathbf{P}^T\mathbf{Z}^c = (\mathbf{I} - \mathbf{Q}\mathbf{A})\mathbf{Z}^c = \mathbf{Z}^c - \mathbf{Q}\mathbf{A}\mathbf{Z}^c = \mathbf{Z}^c - \mathbf{Q}\mathbf{Z}^c\Lambda^c = \mathbf{Z}^c - \mathbf{0} = \mathbf{Z}^c.$$

□

Lemma 4.1.1. Let $\mathbf{P}$ be as in Definition 3.4.1, and $\mathbf{A}$ as in Theorem 4.1.1. If the deflation-subspace matrix is chosen as p eigenvectors, $\mathbf{v}_j$, of $\mathbf{A}$, i.e., $\mathbf{Z} = [\mathbf{v}_1, ..., \mathbf{v}_p]$, then $\mathbf{PA}$ has the same eigenvectors as $\mathbf{A}$ and the spectrum is given by:

$$\sigma(\mathbf{PA}) = \{0, ..., \lambda_{p+1}..., \lambda_n\}.$$

Proof. The deflation subspace matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \mathbf{I} - \mathbf{A}\mathbf{Q},$$

multiplying by $\mathbf{A}$, we have

$$\mathbf{P}\mathbf{A} = \mathbf{A} - \mathbf{A}\mathbf{Q}\mathbf{A}.$$

Using Theorem 4.1.1 g), the eigenvalues, λ_j, of $\mathbf{PA}$ are given by:

$$\mathbf{P}\mathbf{A}\mathbf{v}_j = \mathbf{A}\mathbf{v}_j - \mathbf{A}\mathbf{Q}\mathbf{A}\mathbf{v}_j = \lambda_j\mathbf{v}_j - \lambda_j\mathbf{A}\mathbf{Q}\mathbf{v}_j = \begin{cases} \lambda_j\mathbf{v}_j - \lambda_j\lambda_j^{-1}\mathbf{A}\mathbf{v}_j, & \mathbf{v}_j \in \mathbf{Z}, \\ \lambda_j\mathbf{v}_j - \mathbf{0}, & \mathbf{v}_j \notin \mathbf{Z}. \end{cases}$$

$$= \begin{cases} \lambda_j\mathbf{v}_j - \lambda_j\mathbf{v}_j = \mathbf{0}, & \mathbf{v}_j \in \mathbf{Z}, \\ \lambda_i\mathbf{v}_j, & \mathbf{v}_j \notin \mathbf{Z}. \end{cases}$$

$$\Rightarrow \quad \sigma(\mathbf{PA}) = \{0, ..., \lambda_{p+1}..., \lambda_n\}.$$

□

Note that, the eigenvalues corresponding to the eigenvectors contained in the deflation subspace matrix $\mathbf{Z}$ are removed from the system matrix. If these eigenvalues are hampering the convergence of the method, removing them leads to an acceleration. Therefore, using eigenvectors as deflation vectors is a suitable choice of deflation vectors; furthermore, we can select which eigenvalues are removed. However, computing the eigenvectors is, usually, computationally expensive.

4.1.2 Recycling deflation

Another possible choice of deflation vectors is using solutions of the system with diverse characteristics, e.g., different right-hand sides, or solutions of the same system at various time steps, also referred to as *snapshots*.

If we want to solve the system $\mathbf{Ax} = \mathbf{b}$ and we collect a set of snapshots that contain diverse right-hand side vectors $\mathbf{b}_i$, such that $\mathbf{b} = \sum_{i=1}^{p} c_i \mathbf{b}_i$, then, the solution is a linear combination of the snapshots, i.e., $\mathbf{x} = \sum_{i=1}^{p} c_i \mathbf{x}_i$, as shown in Lemma 4.1.2.

Using these solutions as deflation vectors leads to convergence of the deflation method in one iteration, this result is presented in Lemma 4.1.3.

Lemma 4.1.2. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a non-singular matrix, and $\mathbf{x}$ is the solution of

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

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

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

The following equivalence holds

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

Proof. " $\Rightarrow$ "

Substituting $\mathbf{x} = \sum_{i=1}^{p} c_i \mathbf{x}_i$ into $\mathbf{Ax} = \mathbf{b}$ leads to:

$$\mathbf{Ax} = \sum_{i=1}^{p} \mathbf{A} c_i \mathbf{x}_i = \mathbf{A}(c_1 \mathbf{x}_1 + ... + c_p \mathbf{x}_p) = \mathbf{b},$$

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

$$\mathbf{A} c_1 \mathbf{x}_1 + ... + \mathbf{A} c_p \mathbf{x}_p = c_1 \mathbf{b}_1 + ... + c_p \mathbf{b}_p,$$

combining the two equations above we get:

$$\mathbf{Ax} = c_1 \mathbf{b}_1 + ... + c_p \mathbf{b}_p = \sum_{i=1}^{p} c_i \mathbf{b}_i = \mathbf{b}.$$

" $\Leftarrow$ "

Substituting $\mathbf{b} = \sum_{i=1}^{p} c_i \mathbf{b}_i$ into $\mathbf{Ax} = \mathbf{b}$ leads to:

$$\mathbf{Ax} = \sum_{i=1}^{p} c_i \mathbf{b}_i.$$

Since $\mathbf{A}$ is non-singular, we multiply the equation above by $\mathbf{A}^{-1}$, and we obtain:

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

using the linearity of $\mathbf{A}^{-1}$, the equation above can be rewritten as:

$$\mathbf{x} = c_1\mathbf{A}^{-1}\mathbf{b}_1 + ... + c_p\mathbf{A}^{-1}\mathbf{b}_p = c_1\mathbf{x}_1 + ... + c_p\mathbf{x}_p,$$

combining the two equations above we get:

$$\mathbf{x} = \sum_{i=1}^{p} c_i\mathbf{x}_i.$$

□

Lemma 4.1.3. If the the deflation matrix $\mathbf{Z}$ is constructed with a set of p vectors

$$\mathbf{Z} = \left[\mathbf{x}_1, ..., \mathbf{x}_p\right],$$

such that $\mathbf{x} = \sum_{i=1}^{p} c_i\mathbf{x}_i$, with $\mathbf{x}_i$ *l.i.*, then the solution of system (3.1) is obtained with one iteration of DCG.

Proof. The relation between $\hat{\mathbf{x}}$ and $\mathbf{x}$ is given in Equation (3.18):

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

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

$$\begin{aligned}
\mathbf{Q}\mathbf{b} &= \mathbf{Z}\mathbf{E}^{-1}\mathbf{Z}^T\left(\sum_{i=1}^{p} c_i\mathbf{b}_i\right) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T\left(\sum_{i=1}^{p} c_i\mathbf{A}\mathbf{x}_i\right) \qquad \text{using Lemma 4.1.2} \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T\left(\mathbf{A}\mathbf{x}_1c_1 + ... + \mathbf{A}\mathbf{x}_pc_p\right) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T(\mathbf{A}\mathbf{Z}\mathbf{c}) \\
&= \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})\mathbf{c} \\
&= \mathbf{Z}\mathbf{c} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + c_3\mathbf{x}_3 + c_4\mathbf{x}_4 + c_5\mathbf{x}_5 \\
&= \sum_{i=1}^{p} c_i\mathbf{x}_i = \mathbf{x}.
\end{aligned}$$

Therefore,

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

is the solution to the original system.
For the second term of Equation (3.18), $\mathbf{P}^T\hat{\mathbf{x}}$, we compute $\hat{\mathbf{x}}$ from Equation (3.19):

$$\begin{aligned}
& \mathbf{P}\mathbf{A}\hat{\mathbf{x}} = \mathbf{P}\mathbf{b} \\
\Leftrightarrow \quad & \mathbf{A}\mathbf{P}^T\hat{\mathbf{x}} = (\mathbf{I} - \mathbf{A}\mathbf{Q})\mathbf{b} && \text{using Lemma 3.4.2 f) and Definition 3.4.1} \\
\Leftrightarrow \quad & \mathbf{A}\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{b} - \mathbf{A}\mathbf{Q}\mathbf{b} \\
\Leftrightarrow \quad & \mathbf{A}\mathbf{P}^T\hat{\mathbf{x}} = \mathbf{b} - \mathbf{A}\mathbf{x} = \mathbf{0} && \text{taking } \mathbf{Q}\mathbf{b} = \mathbf{x} \text{ from above} \\
\Leftrightarrow \quad & \mathbf{P}^T\hat{\mathbf{x}} = \mathbf{0}, && \text{as } \mathbf{A} \text{ is invertible.}
\end{aligned}$$

Then, the solution

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

is obtained in one step of DCG. □

Note that, if the coefficients c_i are know, the solution can be obtained directly using $\mathbf{x} = \sum_{i=1}^{p} c_i\mathbf{x}_i$ In the application considered the coefficients c_i can be easily obtained since the rhs $\mathbf{b}$ is nonzero only where the cells contain wells. Otherwise, one can obtain the coefficient c_i by solving the system associated to $\mathbf{b} = \sum_{i=1}^{p} c_i\mathbf{b}_i$. In this case solving the normal equations is faster than applying deflation.

Since Lemma 4.1.3 holds whenever $\mathbf{x} = \sum_{i=1}^{p} c_i\mathbf{x}_i$, it is important to take all possible l.i. snapshots into account; therefore, it is required to consider the sources and system boundary conditions. If homogeneous Neumann boundary conditions are imposed, only snapshots accounting for the sources are required; however, for the case of non-homogeneous Dirichlet boundary conditions, an extra snapshot incorporating them has to be considered. An analysis on the selection of snapshots based on the sources and boundary conditions is presented next.

4.1.3 Snapshots selection.

Wells are modeled with the Peaceman model (see Section 2.2), each of them is associated to a vector $\mathbf{b}_i$ with a non-zero value $(\mathbf{b}_i)_k = J * b_{bh}$ in the cell containing the well (See Equation 2.8), where $(\mathbf{b}_i)_k$ is a non-zero entry of the vector $\mathbf{b}_i$ at position k.

Given a reservoir containing p wells, i.e., a right-hand side $\mathbf{b} \in \mathbb{R}^n$ containing p non-zero elements, the boundary conditions imposed on the system determine the number of possible snapshots of the form $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$, such that the deflation method converges within one iteration.

For homogeneous Neumann boundary conditions, the right-hand side vector $\mathbf{b} = \sum_{i=1}^{p-1} \mathbf{b}_i$ is a linear combination of $p-1$ vectors $\mathbf{b}_i$, where each $\mathbf{b}_i$ corresponds to a well.

For non-homogeneous Dirichlet boundary conditions, an extra vector $\mathbf{b}_b$, accounting for the non-homogeneous boundaries, has to be considered and the right-hand side vector is given by $\mathbf{b} = \sum_{i=1}^{p} \mathbf{b}_i + \mathbf{b}_b$. These results are presented in Lemma 4.1.4 and Lemma 4.1.5 for homogeneous Neumann and non-homogeneous Dirichlet boundary conditions, respectively.

Lemma 4.1.4. Let $\mathbf{b} \in \mathbb{R}^n$ be a vector containing $p \leq n$ non-zero entries such that $\mathbf{A}\mathbf{x} = \mathbf{b}$, where each non-zero entry corresponds to a cell containing a source. Imposing homogeneous Neumann boundary conditions, there are at most $p-1$ linearly independent vectors $\mathbf{b}_i$ such that $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$. Then,

$$\mathbf{b} = \sum_{i=1}^{p-1} c_i\mathbf{b}_i. \tag{4.1}$$

Furthermore, $dim[\mathcal{R}(\mathbf{A})] \leq n-1$.

Proof. According to Lemma 2.2.1, for each $\mathbf{b}_i$ the following holds:

$$\sum_k (\mathbf{b}_i)_k = 0,$$

where $\mathbf{b}_i \neq \mathbf{0}$. Hence, each $\mathbf{b}_i$ should contain at least two non-zero entries. Therefore, p linear independent vectors $\mathbf{b}_i$ cannot be found.

However, by putting $(\mathbf{b}_i)_l = (\mathbf{b})_l \neq 0$ for every $\mathbf{b}_i$, each $\mathbf{b}_i$ can have one extra unique non-zero entry $(\mathbf{b}_i)_k$ from the $p-1$ remaining. This leads to $p-1$ linear independent vectors $\mathbf{b}_i$, and we have:

$$\mathbf{b} = \sum_{i=1}^{p-1} c_i \mathbf{b}_i.$$

Since we consider homogeneous Neumann boundary conditions, $\mathbf{A}$ is singular; hence, the vector $\mathbf{x}_n = [1 \quad 1 \quad ... \quad 1 \quad 1]^T$ is in the null space. Therefore, the null space is non-empty and has dimension ≥ 1. As a consequence, we have:

$$\begin{aligned} n = dim[A] &= dim[\mathcal{N}(\mathbf{A})] + dim[\mathcal{R}(\mathbf{A})] \\ \text{as} \quad \mathbf{x}_n \in \mathcal{N}(\mathbf{A}) \quad &\Rightarrow \quad dim[\mathcal{N}(\mathbf{A})] \geq 1 \\ \Rightarrow \quad dim[\mathcal{R}(\mathbf{A})] &\leq n-1. \end{aligned}$$

□

This implies that, for a vector $\mathbf{b} \in \mathbb{R}^n$ we can construct, at most, $n-1$ linearly independent subsystems $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$.

Additionally, from Lemma 4.1.2, the solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ is given by:

$$\mathbf{x} = \sum_{i=1}^{p-1} c_i \mathbf{x}_i.$$

Thus, the solution space has dimension $p-1$ and $\mathbf{x} \in span\{\mathbf{x}_1, ..., \mathbf{x}_{p-1}\}$. Here, each $\mathbf{x}_i$ is a snapshot, or solution of the subsystem $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$ with the same homogeneous Neumann boundary conditions.

In our applications, considering a system containing p wells, this translates into obtaining $p-1$ independent snapshots $\mathbf{x}_i$, each of them containing two of the wells. Therefore, if the deflation matrix is chosen as:

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

with accurate enough snapshots used as deflation vectors (see Lemma 4.1.6), the solution is achieved in one DCG iteration (see Lemma 4.1.3).

Lemma 4.1.5. In case of a system containing p sources and non-homogeneous Dirichlet boundary conditions, $p+1$ linearly independent vectors can be found such that:

$$\mathbf{b} = \sum_{i=1}^{p} c_i \mathbf{b}_i + \mathbf{b}_b, \tag{4.2}$$

where $\mathbf{b}_b$ accounts for the non-homogeneous Dirichlet boundary conditions.

Proof. Compared to the previous Lemma, $\mathbf{b}$ contains p_b extra non-zero entries, accounting for the non-homogeneous Dirichlet boundary conditions. Furthermore, each $\mathbf{b}_i$ can contain only one non-zero entry, since condition of Lemma 2.2.1 is not required.

Therefore, a set of p l.i. vectors can be constructed, each of them representing a source. Furthermore, we can define one extra l.i. vector, associated with the non-homogeneous Dirichlet boundary conditions, containing all the remaining p_b non-zero entries.

Hence, we have:

$$\mathbf{b} = \sum_{i=1}^{p+1} c_i \mathbf{b}_i = \sum_{i=1}^{p} c_i \mathbf{b}_i + \mathbf{b}_b,$$

which implies that $p+1$ l.i. vectors can indeed be found. □

As in the previous case, from Lemma 4.1.2, the solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ is given by:

$$\mathbf{x} = \sum_{i=1}^{p} c_i \mathbf{x}_i + c_b \mathbf{x}_b.$$

This means that the solution space has dimension $p+1$, where each $\mathbf{x}_i$ is a snapshot, or solution of a subsystem $\mathbf{A}\mathbf{x}_i = \mathbf{b}_i$. The first p correspond to the p sources, in our test cases p wells, with homogeneous Dirichlet boundary conditions and the last one corresponds to the non-homogeneous one. Therefore, by taking

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

as deflation matrix, the solution is achieved in one DCG iteration (see Lemma 4.1.3). The snapshots, $\mathbf{x}_i$, are usually obtained with iterative methods, which implies that the solution is an approximation; therefore, it is necessary to take into account the error of the approximation. The influence of the accuracy on the deflation method is studied next.

4.1.4 Accuracy of the snapshots

Iterative methods result in an approximated solution, for which the accuracy can be prescribed. In this work, we use as stopping criterion the relative residual, that, for a preconditioned system is given by:

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

Here, $\mathbf{M}^{-1}$ is the preconditioner, $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ is the residual corresponding to the approximation $\mathbf{x}_k$. The snapshots are computed by using iterative methods, which implies that they contain an error. To achieve the solution within one DCG iteration, it is necessary to take into account the accuracy of the snapshots. Lemma 4.1.6 presents the relation between the snapshots and the deflation approximation accuracy.

Lemma 4.1.6. Given a set of p snapshots $\{\mathbf{x}_i\}$ computed with an iterative method using a tolerance of $r_k = \frac{||\mathbf{r}_i^k||_2}{||\mathbf{b}_i||_2} = 10^{-\eta}$, such that

$$\mathbf{x} = \sum_{i=1}^{p} c_i \mathbf{x}_i,$$

then, the error of the approximated solution computed with deflation after one iteration is given by $e^1 = \kappa_2(\mathbf{A}) \times 10^{-\eta}$.

Proof. After one iteration of DCG we obtain the solution (see Lemma 4.1.3):

$$\mathbf{x}^1 = \sum_{i=1}^{p} c_i \mathbf{x}_i,$$

From Equation (3.5), the error of this solution is given by:

$$\begin{aligned} e^1 =& \frac{||\mathbf{x} - \mathbf{x}^1||_2}{||\mathbf{x}||_2} = \frac{||\sum_{i=1}^{p} c_i(\mathbf{x}_i - \mathbf{x}_i^1)||_2}{||\sum_{i=1}^{p} c_i \mathbf{x}_i||_2} = \\ =& \frac{\sum_{i=1}^{p} ||\mathbf{x}_i - \mathbf{x}_i^1||_2}{\sum_{i=1}^{p} ||\mathbf{x}_i||_2} \leq \frac{\sum_{i=1}^{p} ||\mathbf{r}_i^1||_2}{\sum_{i=1}^{p} ||\mathbf{b}_i||_2} \kappa_2(\mathbf{A}), \qquad \text{Using Eq. (3.8)} \\ =& r_k \kappa_2(\mathbf{A}) = \kappa_2(\mathbf{A}) \times 10^{-\eta}. \end{aligned}$$

□

Therefore, the accuracy of the solution obtained using recycling deflation vectors is not the same as the accuracy of the snapshots, and it is related to the condition number of the system matrix.

We showed that selecting a set of accurate enough linearly independent (l.i.) vectors we achieve convergence in one iteration. However, the selection of the l.i. set is not always possible and new strategies to obtain this information are required. In the next section, we introduce the POD-based deflation method, as an alternative to efficiently obtain system information and reuse it in a deflation procedure.

4.2 Implementation of the POD-based deflation method

The implementation of the deflation method requires the selection of set of deflation vectors that can enhance the convergence of iterative methods by improving the spectrum of the system matrix $\mathbf{A}$.

We propose the use of a POD basis $\mathbf{\Psi}$ as the subspace-deflation matrix $\mathbf{Z}$ in a deflation procedure. The complete solution strategy consists of three stages:

i) **Snapshots collection**. As mentioned in Section 4.1.3, each well results in one non-zero term $(\mathbf{b})_k = J * b_{bh}$ in the right-hand side vector. One or more

wells can be associated with a vector $\mathbf{b}_i$, where each of them represents a well configuration.

A set $\mathbf{X}$ of snapshots, vectors $\mathbf{b}_i$ with different configurations or at various time steps, is obtained for the computation of the POD basis. To capture the snapshots, we propose three approaches. The first one is recycled deflation, used for a time-independent problem $\mathbf{Ax} = \mathbf{b}$. The second one is a moving window approach, and the last one is a training simulation approach, both of them used when the system varies in time, $\mathbf{A}^t\mathbf{x}^t = \mathbf{b}^t$. These approaches are further explained below.

Recycling deflation approach: the snapshots are a set of system solutions $\mathbf{x}_i$ with different right-hand sides $\mathbf{b}_i$, such that $\mathbf{Ax}_i = \mathbf{b}_i$, where $\mathbf{A}$ is the original system matrix, for p snapshots we get:

$$\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_p].$$

Moving window approach: the snapshots are captured 'on the fly,' i.e., they are the most recently computed solutions, obtained during the previous time steps. Taking p snapshots for a time step t we have:

$$\mathbf{X} = [\mathbf{x}^{t-p+1}, ..., \mathbf{x}^{t-1}].$$

With this approach, the first snapshots cannot be computed for the first time steps with the DICCG method, as we lack a deflation subspace matrix $\mathbf{Z}$. Instead, the first p time steps are computed with the ICCG method. The rest of the snapshots are obtained with DICCG.

Training phase approach: a full simulation is run, where the right-hand sides are randomly varied by changing the pressure in the production wells. For a simulation consisting of n time steps we obtain:

$$\mathbf{X} = [\mathbf{x}^1, ..., \mathbf{x}^n].$$

The solutions of this simulation are obtained with the ICCG method.

During the training phase, the bhp of the production wells P is varied randomly, and it takes values between P^1 and P^2, which implies that the right-hand side $\mathbf{b}^t$ is constantly changing during the simulation.

ii) **POD basis computation**. The previously obtained snapshots are used to construct a basis ($\mathbf{\Psi}$), as introduced in Section 3.5. The algorithm is presented below, Algorithm 7.

iii) **Solution of the linear system**. The POD basis is used as subspace-deflation matrix ($\mathbf{Z}$) in a deflation procedure for the solution of the linear system. The algorithms of the complete procedure are presented in Algorithm 8 for a moving window approach, and Algorithm 9 for a training phase approach.

Algorithm 7 Computing the POD basis from a set of snapshots.

Given a set of snapshots,
$\mathbf{X}^{1:p} = \{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^p\}$*,
compute the average value of the set,
$\overline{\mathbf{x}} = \frac{1}{p}\sum_{j=1}^{p} \mathbf{x}^j$
substract the average from the original set,
$\overline{\mathbf{x}}^j = \mathbf{x}^j - \overline{\mathbf{x}}$
construct the covariance matrix,
$\overline{\mathbf{R}} := \frac{1}{p}\overline{\mathbf{X}}\overline{\mathbf{X}}^T, \qquad \overline{\mathbf{X}}^{1:p} = \{\overline{\mathbf{x}}^1, \overline{\mathbf{x}}^2, ..., \overline{\mathbf{x}}^p\}$
obtain the SVD of the transposed covariance matrix,
$\mathbf{V}\Lambda\mathbf{V}^T = \overline{\mathbf{R}}^T$
compute the eigenvectors of the covariance matrix,
$\mathbf{U} = \mathbf{X}\mathbf{V}(\Lambda^T)^{\frac{1}{2}}$
construct the basis (Ψ) with the eigenvectors ($\mathbf{u}^i \in \mathbf{U}$) corresponding to the p largest eigenvalues ($\lambda_i \in diag(\mathbf{D})$),
$\Psi^{1:p} = \{\mathbf{u}^1, ..., \mathbf{u}^p\}$.

*where, $\mathbf{X}^{a:b} := \{\mathbf{x}^a, \mathbf{x}^{a+1}, ..., \mathbf{x}^b\}$.

Algorithm 8 Deflation, moving window variant, solving $\mathbf{A}^t\mathbf{x}^t = \mathbf{b}^t$.

Compute the solution of the first p time steps with ICCG.
for $t = 1, \quad ..., \quad p$
$\mathbf{x}^t = (\mathbf{A}^t)^{-1}\mathbf{b}^t$
end for
Compute the POD basis from collected the snapshots, see Algorithm 7.
$\mathbf{Z} = \Psi^{1:p} = [\mathbf{v}^1, ..., \mathbf{v}^p]$
Compute the solution of the remaining time steps with DICCG.
for $t = p+1, \quad ..., \quad steps$
$\mathbf{x}^t = (\mathbf{A}^t)^{-1}\mathbf{b}^t$
Update the POD basis with the recently computed solution, see Algorithm 7.
$\mathbf{Z} = \Psi^{t-p+1:t} = [\mathbf{v}^{t-p+1}, ..., \mathbf{v}^p]$
end for

Algorithm 9 Deflation, training phase variant, $\mathbf{A}^t\mathbf{x}^t = \mathbf{b}^t$.

Run a training phase simulation with ICCG changing randomly the producer's pressure.
for $t = 1, \quad ..., \quad steps$
$\mathbf{x}^t = (\mathbf{A}^t)^{-1}\mathbf{b}^t$
end for
$\mathbf{X}^{1:steps} = \{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{steps}\}$
Compute the POD basis from the snapshots, see Algorithm 7,
$\mathbf{Z} = \Psi^{1:p} = \{\mathbf{v}^1, ..., \mathbf{v}^p\}$
Run various simulation with fixed pressure in the wells using DICCG,
for $t = 1, \quad ..., \quad steps$
$\mathbf{x}^t = (\mathbf{A}^t)^{-1}\mathbf{b}^t$
end for

We study the performance of the POD-based deflation method by observing the change in the number of iterations required to achieve a certain accuracy when finding an approximate solution $\mathbf{x}_k$. We compare the performance of DICCG and the ICCG methods regarding the number of iterations and the number of flops per iteration; the latter are presented in the following section.

4.3 Complexity

The number of operations required to perform the methods studied in this work is presented in this section. Specific details about the computation of operations can be found in Appendix 9.2, and [46–48].

As mentioned before, the implementation of the deflation method consists of three stages: snapshots collection with the ICCG method, Singular Value Decomposition (SVD) of the covariance matrix constructed with the snapshots, and solution procedure with the DICCG method.

In Table 4.1, we present the number of operations required during the initialization of the method and per iteration, for various sparsity values (m) and a different number of deflation vectors (p) for both: the ICCG and DICCG methods. The number of operations are computed as follows, for more details see 5:

Initial

$$\text{ICCG} \sim (m^2 + 4m + 2)n$$

$$\text{DICCG} \sim ((m + 2p + 4)m + (4p + 2)p + 2)n + p^3/3$$

Iteration

$$\text{ICCG} \sim (4m + 9)n$$

$$\text{DICCG} \sim (4m + 4p + 9)n$$

Initial work						
	2D, $m = 5$			3D, $m = 7$		
p	1	4	10	1	4	10
ICCG	$47n$			$79n$		
DICCG	$63n$	$159n$	$567n$	$99n$	$207n$	$639n$
$\frac{\text{DICCG}}{\text{ICCG}}$	$1 + \frac{16}{47}$	$3 + \frac{18}{47}$	$12 + \frac{3}{47}$	$1 + \frac{20}{79}$	$2 + \frac{49}{79}$	$8 + \frac{7}{79}$
SVD	$[(8p - 1)n + 10p^2 - 2p + 1]p$, for p snapshots					
Work per iteration						
p	1	4	10	1	4	10
ICCG	$29n$			$37n$		
DICCG	$33n$	$45n$	$69n$	$41n$	$53n$	$77n$
$\frac{\text{DICCG}}{\text{ICCG}}$	$1 + \frac{4}{29}$	$1 + \frac{16}{29}$	$2 + \frac{11}{29}$	$1 + \frac{4}{37}$	$1 + \frac{16}{37}$	$2 + \frac{3}{37}$

Table 4.1: *Number of Operations for the ICCG and DICCG methods.*

For each iteration, the computational cost using the ICCG method is $29n$ for a 2D case, and $37n$ for a 3D problem of size n, while for the DICCG method using p deflation vectors is $(29 + 4p)n$ for 2D and $(37 + 4p)n$ for 3D cases. This implies that the DICCG method requires $\sim 1 + \frac{4p}{29}$ of the number of ICCG operations for the 2D case, and $\sim 1 + \frac{p}{10}$ for the 3D case. Therefore, the fewer deflation vectors used, the more gain can be achieved.

Note that, more work is required to initialize the DICCG method when compared with the work per iteration. However, the initialization process is performed only once, and DICCG pays off if the gain per iteration is significant.

Concluding remarks In this chapter, we presented some theoretical results related to the selection of deflation vectors. We studied two common choices of deflation vectors: eigenvectors and recycled vectors. We showed that using eigenvectors the corresponding eigenvalues are set to zero, and that selecting a set of linearly independent vectors spanning the solution space as deflation vectors leads to convergence of the deflation method in one iteration.

These choices lead to a good performance for the deflated method; however, computing eigenvectors is computationally expensive, and the l.i. set spanning the solution space is not always known. To overcome this situation, we introduced the POD-based deflation methodology, that makes use of a POD basis as deflation vectors that aims to effectively capture system information, i.e., capture a large amount of system information in the less number of vectors.

In the following chapters, we illustrate the theoretical results presented in this chapter with numerical experiments, and we explore the performance of the POD-based deflation method with reservoir simulation problems.

Chapter 5

Numerical experiments 1: Incompressible single-phase reservoir simulation

In Chapter 4, we introduced the POD-based deflation methodology, together with some theoretical results that help us to find the right deflation vectors, such that optimal performance of the deflation method is achieved. We illustrate the application of the previously-introduced methodology for reservoir simulation problems in this chapter, Chapter 6, and Chapter 7.

In particular, we test the methodology for the simulation of incompressible and compressible single-phase flows in a layered reservoir with a contrast between permeability layers varying from $10^1 - 10^7$, and for the SPE 10 benchmark presenting a contrast on permeability coefficients of $\mathcal{O}(10^7)$.

We compare the use of snapshots, eigenvalues, POD-based, and subdomain-based vectors as deflation vectors in a deflation procedure for the solution of the resulting linear system. The numerical experiments for the incompressible case are presented in this chapter, and the compressible experiments are studied in Chapter 6.

This chapter is based on:

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

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

and additional work.

5.1 Incompressible fluid

These experiments are performed in order to analyze the behavior of the Deflated Conjugate Gradient method preconditioned with the Incomplete Cholesky factorization (DICCG). We solve an elliptic equation, resulting from flow simulation of an incompressible fluid throughout an incompressible reservoir (see Equation (2.28)).

We study various choices of deflation vectors: eigenvectors, subdomain based vectors, snapshots and the snapshots-based basis functions obtained from a POD procedure. According to Lemma 4.1.3, if the deflation vectors contain all the system information, convergence is achieved within one DICCG iteration. The correct selection of snapshots such that the most relevant system information is captured depends on the boundary conditions.

In Lemma 4.1.4 and Lemma 4.1.5, we presented a way of selecting a set of snapshots that captures the essential system information, when the system presents Neumann and Dirichlet boundary conditions.

If homogeneous Neumann boundary conditions are imposed, only snapshots related to the sources are required; however, the resulting system is singular, and it is necessary to balance the incoming and outcoming flux of the system (see Lemma 2.2.1); thus, this balance has to be considered when selecting the set of linearly independent vectors.

If the system contains non-homogeneous Dirichlet boundary conditions, aside from the sources-related snapshots, an extra snapshot associated with the boundary conditions is required. We illustrate these results with a series of experiments, for which the sources are wells located inside the reservoir and the boundary conditions.

Another relevant element to take into account to obtain an optimal performance is the tolerance of the snapshots collected with the ICCG method, which has a large influence on the accuracy of the approximation obtained with the DICCG method. We present some experiments to investigate the dependence of the performance of the DICCG method on the accuracy of the snapshots by changing the tolerance of the solvers during the snapshots collection and the solution of the test cases.

We compare the performance of the deflation method when using the previously introduced *l.i.* snapshots, and the POD-based vectors with some common choices of deflation vectors: subdomain-based and eigenvalues of the system matrix $\mathbf{A}$, and the preconditioned system matrix $\mathbf{M}^{-1}\mathbf{A}$.

The experiments are performed on a Cartesian grid with different grid sizes. We investigate the influence of the size of the problem on the performance of the methods. A general overview of the studied model is presented below, but more specifications are presented for each experiment.

The matrices corresponding to the linear systems $\mathbf{A}$ and the right-hand sides $\mathbf{b}$ are obtained with the Matlab Reservoir Simulation Toolbox (MRST) [7].

Model. We simulate flow through porous media with a constant porosity field of 0.2. We study an incompressible single-phase model for a fluid presenting a viscosity of $\mu = 1\ cp$, and a density of $\rho = 1014\ kg/m^3$. We study cases with homogeneous Neumann and non-homogeneous Dirichlet boundary conditions.

For the modeling, we make use of Darcy's law and the mass balance equation (see Section 2.1). The spatial discretization scheme used is the finite volumes method, also

known as the Two-Point Flux Approximation (TPFA), which is implemented with MRST (see Section 2.2.1). After discretization, the resulting linear system reads:

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

where $\mathbf{T}$ is the transmissibility matrix, and $\mathbf{q}$ are the sources: wells controlled via the bhp; or prescribed pressures on the boundary for the test cases presenting non-homogeneous Dirichlet.

The model presenting homogeneous Neumann boundary conditions contains five wells, four producers ($P_{1:4}$) on the corners of the domain and one injector (I) in the center. When imposing non-homogeneous Dirichlet boundary conditions, we place two production wells and two injection wells inside the reservoir.

Snapshots. As mentioned above, for the DICCG method we need to select a set of deflation vectors. In the first series of experiments, the deflation vectors are solutions of the system with various well configurations and boundary conditions. These solutions, called snapshots, are obtained with the ICCG method, the stopping criterion is given for each problem. The well configuration used to obtain each snapshot depends on the problem. We present the configuration of the system to solve for each experiment.

POD-based deflation vectors. The POD method requires a basis for the projection of the high-order model into a smaller one. This basis contains the eigenvectors corresponding to the largest eigenvalues of the data snapshot covariance matrix $\overline{\mathbf{R}}$, as defined in Equation (3.27).

In the second set of experiments, the previously-obtained snapshots are used to construct the matrix $\overline{\mathbf{X}}$. The eigenvectors corresponding to the largest eigenvalues of the covariance matrix $\overline{\mathbf{R}} = \overline{\mathbf{X}}\overline{\mathbf{X}}^T$ are used as deflation vectors. This selection of vectors is addressed to as POD-based deflation vectors.

Solvers. The solution of the linear system is approximated with the ICCG and DICCG methods. Implementation of the DICCG method requires the computation of a set of deflation vectors. For the first set of experiments, we use snapshots as deflation vectors. For the second set, the POD-based deflation vectors are selected. These two proposed options are compared with the usual selections of deflation vectors: subdomain-based vectors and eigenvectors of $\mathbf{A}$ and $\mathbf{M}^{-1}\mathbf{A}$.

The preconditioned relative residual computed at each PCG iteration, $\mathbf{M}^{-1}\mathbf{r}_k$, is employed as tolerance or stopping criterion, and it is varied for each problem. The residual $\mathbf{r}_k$ is the one computed at each iteration. We study the influence of the snapshots' accuracy on the performance of the DICCG method by varying their tolerance.

For some cases, we study the relative residual of the current iteration $\mathbf{r}_k$ obtained by multiplying $\mathbf{r}_k$ by $\mathbf{M}$, and the true relative residual. This residual is computed from the approximate solution at iteratio k as $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$. The definition of these variables, together with the true relative error (e_k), are presented in Table 5.1.

Variable	Definition
Relative preconditioned residual	$M^{-1}r_k = \frac{\|\|\mathbf{M}^{-1}\mathbf{r}_k\|\|_2}{\|\|\mathbf{M}^{-1}\mathbf{b}\|\|_2}$.
Relative residual	$r_k = \frac{\|\|\mathbf{r}_k\|\|_2}{\|\|\mathbf{b}\|\|_2}$.
True relative preconditioned residual	$M^{-1}r_k^t = \frac{\|\|\mathbf{M}^{-1}(\mathbf{b}-\mathbf{A}\mathbf{x}_k)\|\|_2}{\|\|\mathbf{M}^{-1}(\mathbf{b})\|\|_2}$.
True relative error	$e_k = \frac{\|\|\mathbf{x}-\mathbf{x}_k\|\|_2}{\|\|\mathbf{x}\|\|_2}$.

Table 5.1: *Stopping criteria.*

The influence of preconditioning and deflation on the spectrum of the linear system is investigated via the condition number. Note that, for the deflation method, we use the effective condition number, which is computed dividing the largest eigenvalue by the smallest non-zero eigenvalue (See Equation 3.22). As mentioned before, when using p system eigenvectors as deflation vectors, the p corresponding system eigenvalues are set to zero. Thus, these eigenvalues are not taken into account for the condition number.

5.1.1 Homogeneous Neumann boundary conditions

In this section, we present a set of experiments in a reservoir containing homogeneous Neumann boundary conditions on all boundaries. The fluid characteristics and the well configurations are as mentioned in the model section.

To construct the deflation subspace matrix ($\mathbf{Z}$) we study three cases :

1. Four linearly independent snapshots

   (DICCG$_4$), $\mathbf{Z} = [\mathbf{x}_1 \quad ... \quad \mathbf{x}_4]$.

2. All of the 15 possible snapshots

   (DICCG$_{15}$), $\mathbf{Z} = [\mathbf{x}_1 \quad ... \quad \mathbf{x}_{15}]$.

3. Four POD basis vectors obtained from the set of 15 snapshots (DICCG$_{POD_4}$),

   $\mathbf{Z} = [\psi_1 \quad ... \quad \psi_4]$.

According to Lemma 2.2.1, the solution of a system with homogeneous Neumann boundary conditions can be found if the fluxes in and out the reservoir are balanced, i.e., the total flux is zero. 15 possible well configurations fulfill this requirement; they are presented in Table 5.2, where P is a given bhp in the wells, and P_0 is the pressure in the reservoir. The position of the active wells can be observed in Figure 5.2 for the first four snapshots.

	P_1	P_2	P_3	P_4	I
$\mathbf{x}_1$	P_0	-P	-P	-P	3P-P_0
$\mathbf{x}_2$	-P	P_0	-P	-P	3P-P_0
$\mathbf{x}_3$	-P	-P	P_0	-P	3P-P_0
$\mathbf{x}_4$	-P	-P	-P	P_0	3P -P_0
$\mathbf{x}_5$	-P	-P	-P	-P	4P
$\mathbf{x}_6$	-P	P_0	P_0	-P	2P-2P_0
$\mathbf{x}_7$	-P	-P	P_0	P_0	2P-2P_0
$\mathbf{x}_8$	-P	P_0	-P	P_0	2P-2P_0
$\mathbf{x}_9$	P_0	-P	-P	P_0	2P-2P_0
$\mathbf{x}_{10}$	P_0	-P	P_0	-P	2P -2P_0
$\mathbf{x}_{11}$	P_0	P_0	-P	-P	2P -2P_0
$\mathbf{x}_{12}$	-P	P_0	P_0	P_0	P-3P_0
$\mathbf{x}_{13}$	P_0	-P	P_0	P_0	P-3P_0
$\mathbf{x}_{14}$	P_0	P_0	-P	P_0	P-3P_0
$\mathbf{x}_{15}$	P_0	P_0	P_0	-P	P -3P_0

Table 5.2: *Possible well configurations.*

The snapshots are obtained by solving systems with homogeneous Neumann boundary conditions and the well configurations presented in Table 5.2; note that, the first four vectors are linearly independent.

Once the correlation matrix is obtained, the system $\mathbf{x}_5$ is solved, which contains all the wells active, i.e., the pressure on the production wells is given by $\mathrm{P}_{i=1:4} = \mathrm{P}$, and the pressure of the injector is the negative sum of the production wells' pressure, I = -4 P. We use this configuration to study an academic layered problem and for the SPE 10 benchmark [10].

Layered problem. For this set of problems, we study a layered reservoir consisting on a 2D Cartesian grid containing 35×35 cells of length $L_x = 10$ $[m]$, $L_y = 10$ $[m]$. The domain is divided into five equally sized layers. Four of these layers have a permeability $\sigma_1 = 0.1$ mD, each of them followed by a layer with a different permeability σ_2 (see Figure 5.1). The contrast in permeability between adjacent layers is given by $c_\sigma = (\frac{\sigma_2}{\sigma_1})$, σ_1 is fixed, and σ_2 changes for each problem; thus, c_σ depends on the value of σ_2, varying from $\sigma_2 = 1$ mD to $\sigma_2 = 10^7$ mD. The producers' pressure is P = 200 [bars].

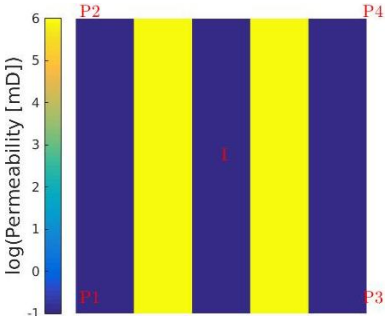


Figure 5.1: *Permeability field and wells location, layers y direction.*

For the comparison with other choices of deflation vectors, selecting subdomain-based deflation vectors, each subdomain represents one layer. Our domain consist on five layers. Therefore, we can divide it into five subdomains. We construct one deflation vector for each subdomain Ω_s by putting ones if a cell (x_i, y_j) lies inside the subdomain, and zeros otherwise.

Results. As mentioned before, to perform the POD-based deflation method, it is required to compute a set of snapshots. For the case of homogeneous Neumann boundary conditions with four producers and one injector, we require four linearly independent snapshots, each of them with different wells configuration (see Table 5.2 and Section 4.1.3). The resulting pressure field of these snapshots ($\mathbf{x}_{1:4}$) as well as the pressure field of the studied system ($\mathbf{x}_5$) are presented in Figure 5.2 for a contrast on permeability layers of $c_\sigma = 10^1$.

The condition number of the system and preconditioned matrices ($\mathbf{A}$, $\mathbf{M}^{-1}\mathbf{A}$), and the effective condition number of the deflated system ($\mathbf{P}\mathbf{M}^{-1}\mathbf{A}$) for various values of c_σ's are presented in Table 5.3.

In Table 5.3, we observe that the condition number of the matrix increases when we increase c_σ and that, after preconditioning, the condition number decreases one order of magnitude. When using the deflation method, the effective condition number is decreased to $\mathcal{O}(10^2)$ for all cases. This behavior suggests that the POD-based deflation method is independent of the condition number of the problem, or, in our case, on the contrast between permeability layers c_σ.

The number of iterations required to achieve convergence for the studied methods is presented in Table 5.4 for a tolerance of $5 \cdot 10^{-11}$. We note that the deflation method

with four linearly independent vectors (DICCG$_4$) and with four POD basis vectors (DICCG$_{P_4}$) requires only one iteration to converge, as expected from Lemma 4.1.3.

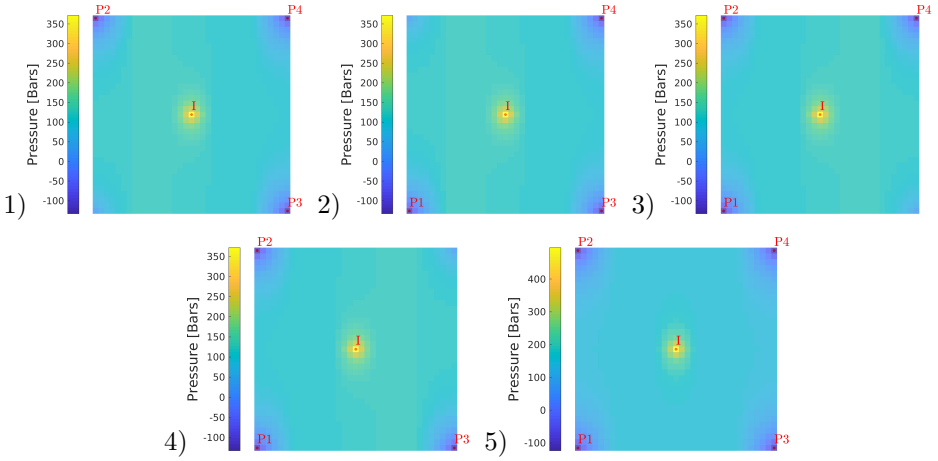


Figure 5.2: *Pressure field of the first four snapshots (1-4) and the studied problem 5).*

c_σ	$\kappa_2(\mathbf{A})$	$\kappa_2(\mathbf{M}^{-1}\mathbf{A})$	$\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})$		
			Deflation subspace matrix		
			$\mathbf{Z} = [\mathbf{x}_1, ..., \mathbf{x}_4]$	$\mathbf{Z} = [\mathbf{x}_1, ..., \mathbf{x}_{15}]$	$\mathbf{Z} = [\psi_1, ..., \psi_4]$
10^1	$3{\cdot}10^4$	$1{\cdot}10^3$	$5{\cdot}10^1$	6	$5{\cdot}10^1$
10^3	$3{\cdot}10^6$	$1{\cdot}10^5$	$9{\cdot}10^1$	6	$9{\cdot}10^1$
10^5	$3{\cdot}10^8$	$1{\cdot}10^7$	$9{\cdot}10^1$	$2{\cdot}10^1$	$9{\cdot}10^1$
10^7	$3{\cdot}10^{10}$	$1{\cdot}10^9$	$9{\cdot}10^1$	7	$9{\cdot}10^1$

Table 5.3: *Condition number of the solved systems for various values of c_σ.*

Tol ICCG $= 5 \cdot 10^{-11}$, Tol DICCG$= 5 \cdot 10^{-11}$				
$\frac{\sigma_2}{\sigma_1}$	ICCG	DICCG$_4$	DICCG$_{15}$	DICCG$_{P_4}$
10^1	67	1	1229	1
10^3	75	1	1229	1
10^5	84	1	1229	1
10^7	73	7	1229	12

Table 5.4: *Number of iterations for various values of c_σ.*

However, when we use 15 snapshots as deflation vectors (DICCG$_{15}$), the deflation method does not behave as expected. For this case, the deflation vectors are linearly depedent, which leads to a singular Galerkin matrix $\mathbf{E}$ (see Section 3.17). In fact, the determinant of $\mathbf{E}$, computed with the Matlab function *det* is $det(\mathbf{E}) = -2.228 \cdot 10^{-26}$, which is much smaller than the machine precision, therefore zero.

As this matrix is singular, it cannot be inverted, therefore the deflation projection

matrix $\mathbf{P}$ uses an incorrect matrix $\mathbf{E}^{-1}$ and the results are wrong, or in this case, the method does not converge.

For DICCG$_4$ and DICCG$_{POD_4}$, we also note that the number of iterations does not change when varying c_σ, except for $c_\sigma = 10^7$, for which it requires 7 and 12 iterations. These results exhibit the independence on the contrast between permeability layers suggested by the almost constant effective condition number of the deflated systems.

We also observe that the number of iterations increases with c_σ when using the ICCG method, except for $c_\sigma = 10^7$, for which we observe an apparent reduction. To further analyze this behavior, Figure 5.3 shows the relative preconditioned residual computed at each iteration $M^{-1}r_k$ (see Algorithms 4 and 6), the relative residual r_k, and the true relative error e_k, for the cases $c_\sigma = 10^1$ and $c_\sigma = 10^7$ (see Table 5.1).

The convergence of the methods is achieved when the relative preconditioned residual computed at each iteration ($M^{-1}r_k$) reaches the desired accuracy $5 \cdot 10^{-11}$. We note in Figure 5.3, that for $c_\sigma = 10^1$, $M^{-1}r_k$, r_k and e_k reach a similar value ($\mathcal{O}(10^{-11})$). However, for $c_\sigma = 10^7$, we observe that $M^{-1}r_k \sim \mathcal{O}(10^{-11})$, $r_k \sim \mathcal{O}(10^{-8})$ and $e_k \sim \mathcal{O}(10^{-6})$.

Furthermore, the true error does not change significantly after the first iteration, thus, we have reached the machine precision and more accurate results are not possible, which is not observed when using $M^{-1}r_k$ as stopping criterion. A better approximation is achieved when using the relative residual r_k. However, obtaining r_k requires the multiplication of $\mathbf{M}^{-1}r_k$ by $\mathbf{M}$, resulting in $(2m+1)n$ extra flops.

From Equation 3.8, we know that the residual is related to the true error via the condition number as $e_k = \kappa \cdot r_k$. As showed in Table 5.3, κ increases when we increase the contrast between permeability layers. Therefore, to compute an accurate approximation, the stopping criterion has to be modified in such a way that it takes into account the system's condition number. This implies that a high accuracy is difficult to obtain if the condition number is large.

Table 5.5 presents the stopping criteria required to achieve a relative error of $5 \cdot 10^{-7}$, i.e., the stopping criterion is $r_k = \frac{5 \cdot 10^{-7}}{\kappa_2}$.

c_σ	$r_k = \frac{5 \cdot 10^{-7}}{\kappa_2(\mathbf{A})}$	$r_k = \frac{5 \cdot 10^{-7}}{\kappa_2(\mathbf{M}^{-1}\mathbf{A})}$	$r_k = \frac{5 \cdot 10^{-7}}{\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})}$		
			Deflation subspace matrix $\mathbf{Z}$		
			$[\mathbf{x}_1, ..., \mathbf{x}_4]$	$[\mathbf{x}_1, ..., \mathbf{x}_{15}]$	$[\psi_1, ..., \psi_4]$
10^1	$3 \cdot 10^{-11}$	$1 \cdot 10^{-10}$	$5 \cdot 10^{-8}$	$6 \cdot 10^{-7}$	$5 \cdot 10^{-8}$
10^3	$3 \cdot 10^{-13}$	$1 \cdot 10^{-12}$	$9 \cdot 10^{-8}$	$6 \cdot 10^{-7}$	$9 \cdot 10^{-8}$
10^5	$3 \cdot 10^{-15}$	$1 \cdot 10^{-13}$	$9 \cdot 10^{-8}$	$2 \cdot 10^{-6}$	$9 \cdot 10^{-8}$
10^7	$3 \cdot 10^{-17}$	$1 \cdot 10^{-15}$	$9 \cdot 10^{-8}$	$7 \cdot 10^{-7}$	$9 \cdot 10^{-8}$

Table 5.5: *Required stopping criteria to have an error in the approximation* $e_k = 5 \cdot 10^{-7}$.

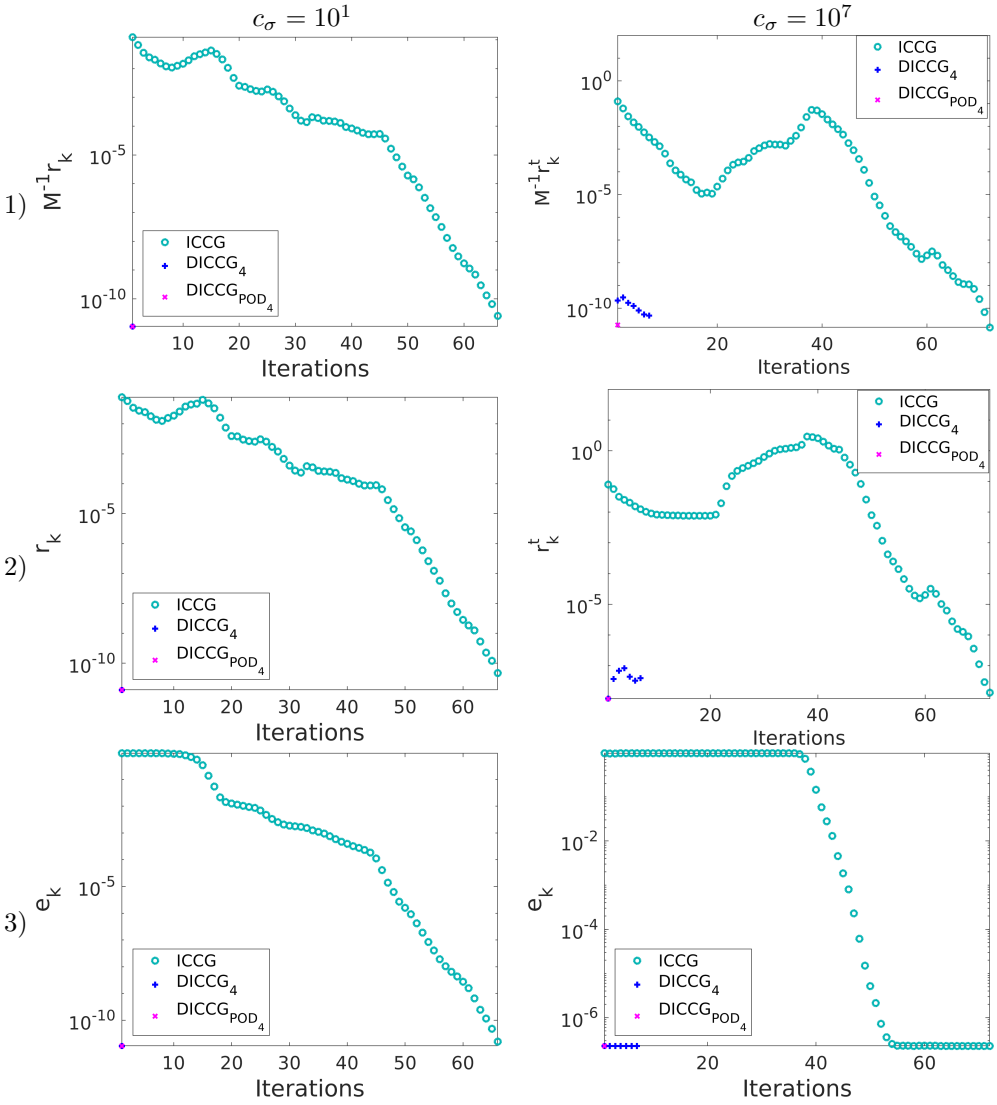


Figure 5.3: *1) Relative preconditioned residual* $M^{-1}r_k$*, 2) Relative residual* r_k*, 3) Relative true error* e_k*, left:* $c_\sigma = 10^1$*, right:* $c_\sigma = 10^7$*,* $Tol = 5 \cdot 10^{-11}$.

We note that the system with $c_\sigma = 10^7$ and without preconditioner requires an accuracy of 10^{-17} for the residual, which is larger than the machine precision; therefore, it is impossible to reach. The rest of the cases do not present this problem. In the following experiments we study the same cases as before, but taking into account the systems' condition number. The results are presented in Table 5.6.

We note that taking into account the condition number for the selection of the stopping criterion, the number of iterations of the ICCG method increases with c_σ, including the case when $c_\sigma = 10^7$. By contrast, the DICCG method reaches convergence in one iteration when using the four linearly independent snapshots or the four

POD basis vectors as deflation vectors, for all cases.

Once a correct tolerance that considers the condition number is chosen, the optimal performance of DICCG is still not guaranteed. In Section 4.1.2, we showed that the condition number and the tolerance of the snapshots influence the convergence of the deflation method.

This behavior is observed in Table 5.4 for $c_\sigma = 10^7$, where the snapshots are not accurate enough, and, in consequence, DICCG$_4$ and DICCG$_{POD_4}$ do not converge in one iteration. In the next examples, we analyze the performance of the deflation method when changing the accuracy of the snapshots $e_s = \frac{\text{Tol ICCG}}{\kappa_{eff}(\mathbf{PM}^{-1}\mathbf{A})}$.

c_σ	Tolerance		Iterations		
	ICCG	DICCG	ICCG	DICCG	
				4	POD_4
10^1	$1 \cdot 10^{-10}$	$9 \cdot 10^{-8}$	64	1	1
10^3	$1 \cdot 10^{-12}$	$9 \cdot 10^{-8}$	79	1	1
10^5	$1 \cdot 10^{-13}$	$9 \cdot 10^{-8}$	89	1	1
10^7	$1 \cdot 10^{-15}$	$9 \cdot 10^{-8}$	139	1	1

Table 5.6: *Number of iterations for various values of c_σ.*

e_1	e_s
$5 \cdot 10^{-6}$	$5 \cdot 10^{-7}$
$5 \cdot 10^{-8}$	$5 \cdot 10^{-9}$
$5 \cdot 10^{-10}$	$5 \cdot 10^{-11}$

Table 5.7: *Snapshots' accuracy (e_s), various DICCG tolerances (e_1), $c_\sigma = 10^1$.*

From Lemma 4.1.6, we know that the true error of DICCG after one iteration is given by $e_1 = \kappa_2(\mathbf{A}) \times 10^{-\eta}$, where $10^{-\eta}$ is the accuracy of the snapshots. The optimal snapshots' accuracy is presented in Table 5.7. For this problem, the effective preconditioned condition number is given by $\kappa_{eff}(\mathbf{PM}^{-1}\mathbf{A}) \sim 10^2$.

In this set of experiments, the accuracy of DICCG is varied from $e_1 = 5 \cdot 10^{-6}$ to $e_1 = 5 \cdot 10^{-10}$, and, for each value, snapshots with diverse accuracy (e_s) are tested. Table 5.8 shows the required number of iterations for ICCG, DICCG$_4$ and DICCG$_{P_4}$, where, we observe that the number of iterations required to reach e_1 decreases when using more accurate snapshots.

$e_1 = 5 \cdot 10^{-6}$			
e_s	ICCG	DICCG$_4$	DICCG$_{P_4}$
$5 \cdot 10^{-1}$	2	48	48
$5 \cdot 10^{-3}$	20	21	23
$5 \cdot 10^{-5}$	47	4	3
$5 \cdot 10^{-7}$	54	1	1
$e_1 = 5 \cdot 10^{-8}$			
$5 \cdot 10^{-3}$	20	36	35
$5 \cdot 10^{-5}$	47	18	20
$5 \cdot 10^{-7}$	54	3	3
$5 \cdot 10^{-9}$	60	1	1
$e_1 = 5 \cdot 10^{-10}$			
$5 \cdot 10^{-5}$	20	31	34
$5 \cdot 10^{-7}$	54	15	15
$5 \cdot 10^{-9}$	60	3	3
$5 \cdot 10^{-11}$	67	1	1

Table 5.8: *Number of iterations required to reach a given tolerance (e_1) for the DICCG method, $c_\sigma = 10^1$.*

As an example, take a tolerance for DICCG of $e_1 = 5 \cdot 10^{-8}$, the required snapshots' accuracy for this case must be higher than $e_k = 1 \cdot 10^{-9}$ (See Table 5.7), using a smaller accuracy (10^{-3}) leads to convergence in 36 iterations; whereas, using the required accuracy ($5 \cdot 10^{-9}$) only one DICCG iteration is required.

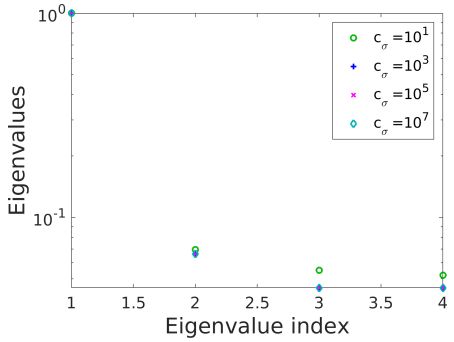


Figure 5.4: *Eigenvalues of* $\mathbf{R}$*, various values of* c_σ*.*

We study the acceleration of the ICCG method via the deflation method by analyzing the change in the number of iterations. To further understand the behavior of the deflation method we analyze the spectrum of the covariance matrix used to construct the POD basis, and the spectrum of the studied deflated systems.

Results showed that four linearly independent (*l.i.*) snapshots contain all of the system information; thus, using them as deflation vectors leads to an optimal performance of DICCG.

The normalized eigenvalues of the covariance matrix constructed with the 15 snapshots presented in Table 5.2 are presented in Figure 5.4 for diverse contrasts on permeability layers. We observe that four of the eigenvalues are larger than the rest in all cases. Therefore, the *l.i.* set of vectors leading to an optimal DICCG performance can be obtained with POD, reaching also convergence in one iteration, as showed in Table 5.4.

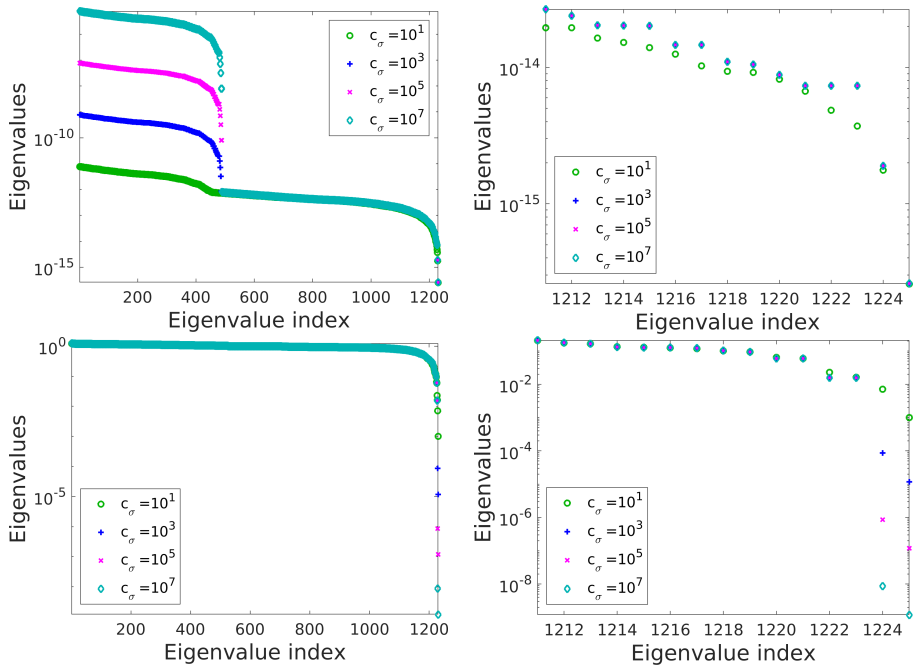


Figure 5.5: *Spectrum of the studied systems, upper:* $\mathbf{A}$ *and the 15 smallest eigenvalues of* $\mathbf{A}$*, lower:* $\mathbf{M}^{-1}\mathbf{A}$ *and the 15 smallest eigenvalues of* $\mathbf{M}^{-1}\mathbf{A}$*.*

The eigenvalues of the system matrix $\mathbf{A}$ and the preconditioned system matrix $\mathbf{M}^{-1}\mathbf{A}$ are presented in Figure 5.5 for various contrast between permeability layers (c_σ). We observe that the largest eigenvalues of the system matrix $\mathbf{A}$ decrease with

c_σ; furthermore, for a matrix presenting $c_\sigma = 10^7$, they are six orders of magnitude larger than the eigenvalues of the matrix with $c_\sigma = 10^1$.

Once the system is preconditioned ($\mathbf{M}^{-1}\mathbf{A}$), the largest eigenvalues are clustered together with the rest of the eigenvalues; nevertheless, the smallest eigenvalues are not treated and the convergence process is affected by them. Moreover, we note that the contrast between permeability layers is now reflected in the smallest eigenvalues. As c_σ increases, the smallest eigenvalues get smaller values, and again, the difference between the smallest $c_\sigma = 10^1$ and the largest $c_\sigma = 10^7$ is around six orders of magnitude.

Deflation methods are implemented to remove the influence of the smallest eigenvalues of the system matrix still present after preconditioning. Figure 5.6 shows the spectrum of the deflated system $\mathbf{PM}^{-1}\mathbf{A}$, where we can observe that the last four are removed from the spectrum, i.e., they are close to zero; therefore they do not influence the performance of the method.

If we remove the smallest eigenvalues (see Figure 5.6, right), we note that the range of the spectrum is reduced considerably. The most significant reduction is achieved for a contrast of 10^7 where we reduce the range of the spectrum from around nine orders of magnitude to only two orders. The Table 5.3 shows the reduction of the condition number from $\kappa(\mathbf{M}^{-1}\mathbf{A}) = 1 \cdot 10^9$ to $\kappa_{eff}(\mathbf{PM}^{-1}\mathbf{A}) = 1 \cdot 10^2$.

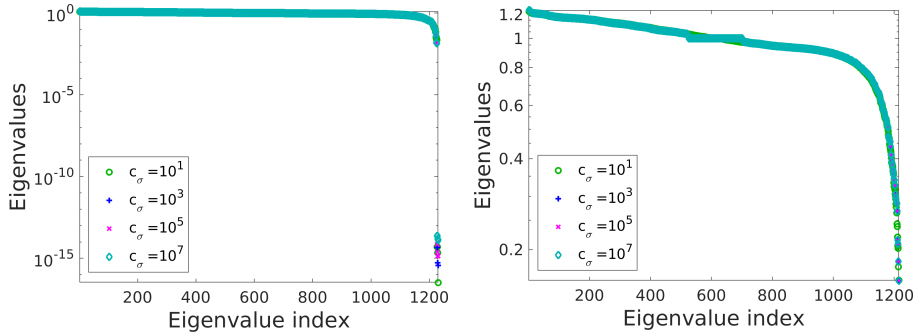


Figure 5.6: *Spectrum of the deflated system, left:* $\mathbf{PM}^{-1}\mathbf{A}$*, right:* $\mathbf{PM}^{-1}\mathbf{A}$*, without the four smallest eigenvalues.*

We showed that the deflation method removes the smallest eigenvalues of the system and that each eigenvector reduces one extreme eigenvalue (Lemma). In the following experiments, we vary the number of deflation vectors to explore the performance of the method further.

p	Iterations		Condition number	
	$\{\mathbf{x}_i\}$	POD	$\{\mathbf{x}_i\}$	POD
1	54	53	200.8	174.2
2	51	50	95.3	75.7
3	51	45	92.5	54
4	1	1	53.3	53.3

Table 5.9: *Condition number and number of iterations, various deflation vectors.*

We study a case with a contrast between permeability layers of $c_\sigma = 10^7$, and we select a stopping criterion for the snapshots of $Tol_{ICCG} = 5 \cdot 10^{-12}$, removing in this way the influence of the accuracy. For this problem, the ICCG method requires 139 iterations (see Table 5.5). As deflation vectors (p) the four *l.i.* snapshots presented in Table 5.2 are selected ($\{\mathbf{x}_i\}$), as well as the basis vectors obtained from

this set (POD). We solved the system $\mathbf{x}_6$ from Table 5.2, similar results are obtained for the other systems.

We vary the number of deflation vectors from one to four, and the stopping criterion for the DICCG method is $Tol_{DICCG} = 5 \cdot 10^{-7}$. The number of iterations and condition number of the deflated systems are presented in Table 5.9.

In both cases, we observe a small decrease in the number of iterations and the condition number when using more snapshots, which is slightly larger for the POD basis. However, the optimal case is using the four *l.i.* snapshot or the four largest POD basis vectors.

Next, we compare some usual choices of deflation vectors with the proposed methodology, in particular, we compare subdomain-based deflation vectors and eigenvectors of the system matrix, and preconditioned system matrix as deflation vectors.

As described before, we study a reservoir containing five layers, for this study, the contrast between permeability layers is 10^7. With this configuration, we can use as deflation vectors subdomain vectors, where each subdomain represents a layer. Therefore, we can create up to five deflation vectors, constructed with ones in the entries corresponding to cells contained in the current subdomain (layer), and zeros in the rest of the entries. We study the use of 1 to 5 subdomain-based deflation vectors.

We also select eigenvectors of the system matrix $\mathbf{A}$ and eigenvectors of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ as deflation vectors; we vary the number of eigenvectors from 1 to 100.

The number of iterations and the condition number of the deflated method for all cases are presented in Table 5.10. The ten smallest eigenvalues of the deflated system with four deflation vectors are presented in Figure 5.7. In this figure, we also show the 200 smallest eigenvalues of the deflated system when we use 100 eigenvalues of the system matrix and the preconditioned system matrix.

From Table 5.10 we observe that, as we increase the number of subdomain vectors and eigenvectors, the number of iterations decreases. If we compare the performance of the different choices of deflation vectors when using 4 of them, we note that using subdomain-based vectors(DICCG_{SD_4}) we require 27 iterations, and for eigenvectors ($\text{DICCG}_{Eigs(A)_4}$ and $\text{DICCG}_{Eigs(M^{-1}A)_4}$) we need 21.

Thus, the performance of the method when selecting four *l.i.* snapshots (DICCG_4) or POD-based vector (DICCG_{P_1}) is considerably better, requiring only one iteration, even if the effective number is smaller for some other choices of deflation vectors.

Regarding the subdomain vectors, the maximum number of subdomains is 5, requiring 23 iterations, which is not a significant improvement compared to the DICCG_4 and (DICCG_{POD_4}) options. The advantage of subdomain vectors compared with the other methods is that they are easy to compute and sparse, that results in less computational work than the rest of the other options. However, if the subdomains are not well defined, it is difficult to construct them, which is the case for the SPE 10 benchmark.

The maximum number of eigenvectors used in this example as deflation vectors is 100, for which a reduction to 11 and 7 iterations is achieved. Hence, a factor of three times fewer iterations than the first case with only one eigenvector, but 100 times more eigenvectors are required. Thus, a further significant reduction in the number of iterations is only achieved with a large number of eigenvectors. Considering that

the eigenvectors also have to be computed, this choice of deflation vectors is the most expensive option for the method. Moreover, the cost of each single iteration increases with the number of deflation vector.

Therefore, we can conclude that when all the system information is collected in a set of snapshots, recycling deflation and the POD basis show the best performance requiring only one iteration.

ICCG		
Iter		$\kappa(\mathbf{A})$
54		$1 \cdot 10^9$
Vectors	Iter	$\kappa_{eff}(\mathbf{PM}^{-1}\mathbf{A})$
Snapshots $\{x_i\}$		
4	1	90
POD		
4	1	90
Subdomain (SD)		
1	53	$1 \cdot 10^9$
2	55	$6 \cdot 10^8$
3	53	$6 \cdot 10^8$
4	27	86
5	23	79
Eigenvectors of $\mathbf{A}$		
1	21	$1 \cdot 10^8$
4	21	80
5	20	80
20	13	80
100	11	80
Eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$		
1	21	$1 \cdot 10^8$
4	21	21
5	21	20
20	13	4
100	7	1.7

Table 5.10: *Number of iterations and condition number, various choices of deflation vectors, tol* $= 5 \cdot 10^{-7}$, $c_\sigma = 10^7$.

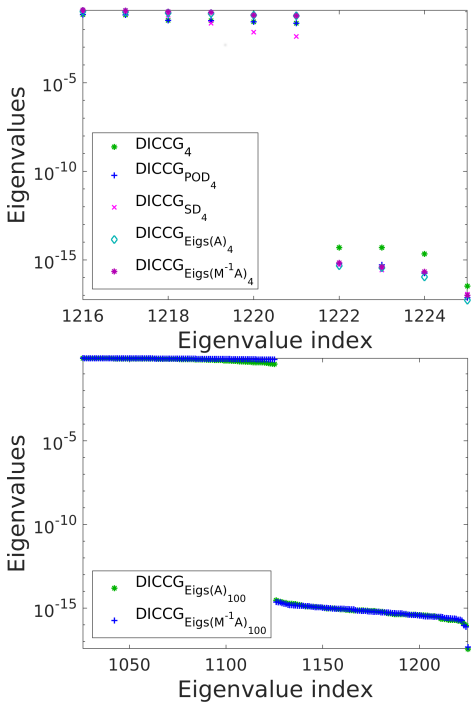


Figure 5.7: *Smallest eigenvalues of the deflated systems,* $(\mathbf{PM}^{-1}\mathbf{A})$*, various choices of deflation vectors,* $c_\sigma = 10^7$.

SPE 10 benchmark. In this section, we study the SPE 10 model, presenting variations in the permeability coefficients up to $\mathcal{O}(10^7)$. Different grid sizes are studied: 16 x 56, 30 x 110, 46 x 166 and 60 x 220, and the full model containing 60 x 220 x 85 cells. The permeability is upscaled averaging the permeability in each grid using the function *sampleFromBox* from MRST.

As in the layered problem, this model contains 5 wells, four producers in the corners and one injector in the center. The permeability field of the full model and the wells' location are shown in Figure 5.8.

The snapshots are obtained by solving the system with different well configurations, as presented in Table 5.2, from where the first four snapshots, used as deflation vectors, are obtained with ICCG and tolerance of $Tol_{ICCG} = 5 \cdot 10^{-12}$.

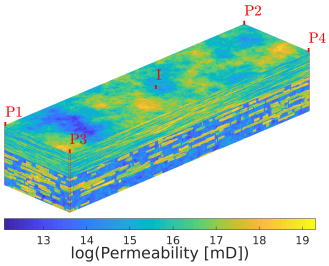


Figure 5.8: *Permeability field and wells location, full SPE model.*

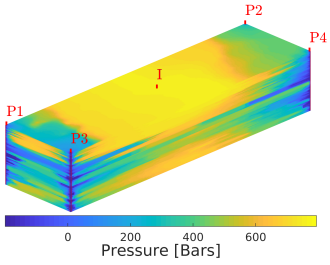


Figure 5.9: *Pressure field, full SPE 10 model.*

A system containing the five wells is solved with the DICCG method and compared with the ICCG method using a stopping criterion of $Tol = 5 \cdot 10^{-7}$. The pressure field is presented in Figure 5.9 for the full model.

We study the performance of the deflation method when using four deflation vectors selected in different ways for the coarse grid problem, 16 x 56 cells. We compare the methods with each other and with the ICCG method, the number of iterations and the condition number of the studied cases are presented in Table 5.11.

Method	Iterations	$\kappa_2(\mathbf{M}^{-1}\mathbf{A})/\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})$
ICCG	67	$5.3 \cdot 10^3$
DICCG	1	185
DICCG$_{POD}$	1	185
DICCG$_{eigs(\mathbf{A})}$	52	224
DICCG$_{eigs(\mathbf{M}^{-1}\mathbf{A})}$	36	33

Table 5.11: *Number of iterations and condition number for various choices of* $\mathbf{Z}$*, four deflation vectors,* $\kappa_2(\mathbf{A}) = 4.6 \cdot 10^6$.

The condition number of the system matrix is $\kappa_2(\mathbf{A}) = 4.6 \cdot 10^6$, after preconditioning, we observe a reduction of four orders of magnitude in the condition number, and a further reduction of around one order of magnitude when using deflation.

Comparing the number of iterations for the different choices of deflation vectors, we note that when using four *l.i.* snapshots (DICCG), or four POD basis vectors (DICCG$_{POD}$) to construct the deflation subspace matrix ($\mathbf{Z}$) we require only one iteration to converge. On the other hand, if we use four eigenvectors of the system matrix (DICCG$_{Eigs(A)_4}$), we require 52 iterations, and with four eigenvectors of the preconditioned system matrix (DICCG$_{Eigs(M^{-1}A)_4}$) we need 36.

Thus, the performance of the method when selecting the four *l.i.* snapshots or four POD-based vectors require the fewer iterations, and when using eigenvectors,

the number of ICCG iterations is reduced to around 80% and 50% which shows a big difference in the performance of the methods.

For the next experiments, we vary the size of the grid, and we study the number of iterations required to achieve an accuracy of $Tol = 5 \cdot 10^{-7}$ with the ICCG method, and the deflation method with four *l.i.* snapshots, and four POD basis vectors as deflation vectors. The number of iterations, together with the contrast in permeability coefficients for the diverse problems is presented in Table 5.12.

We observe that the contrast in the permeability coefficients for different grid sizes varies slightly; nonetheless, the order of magnitude remains the same for all cases.

We note that increasing the size of the problem results in an increment on the number of iterations for the ICCG method, contrary to the deflation method that requires one iteration.

Grid size	16 x 56	30 x 110	46 x 166	60 x 220	60 x 220 x 85
System size	896	3300	7636	13200	$1.12 \cdot 10^6$
Contrast ($\times 10^7$)	1.04	2.52	2.6	2.8	3.01
ICCG	67	136	199	282	1983
DICCG	1	1	1	1	1
DICCG$_{POD}$	1	1	1	1	1

Table 5.12: *Contrast in permeability layers for various grid sizes.*

In the following examples, we vary the number of deflation vectors when using snapshots and POD basis vectors for the solution of the full SPE 10 model. The number of iterations required to converge for both choices of deflation vectors is presented in Table 5.13.

We note that the selection of one POD vectors as deflation vectors reduces the number of ICCG iterations to 63%, while the same number of *l.i.* vectors reduces this number to 84% of the ICCG iterations. Using three, the reduction is to 33% of the ICCG iterations with POD basis vectors, and 78% with *l.i.* vectors. This indicates that the POD basis vectors contain more information than the *l.i.* set.

	Iterations		Percentage	
ICCG	1938		100	
p	$\{\mathbf{x}_i\}$	POD	$\{\mathbf{x}_i\}$	POD
1	1630	1222	84	63
2	1550	878	80	45
3	1507	646	78	33
4	1	1		

Table 5.13: *Number of iterations for various p. Full SPE 10, Neumann boundary conditions.*

Figure 5.10 shows the eigenvalues of the data correlation matrix constructed from the snapshot. In this case, all the system information is contained in four eigenvalues, in Table 5.14, we compute the percentage of the information that is collected when adding eigenvalues (see Equation (3.31)). The four eigenvalues imply that all the system information (100%) is pbtained, and we note that using only three of them 95% of the information is obtained. However, for the POD method to effectively capture information, it is required to have $\alpha \sim 0.99$, i.e. 99% [11].

We present the relative residual of the deflation method in Figure 5.11 for snap-

shots as deflation vectors (left), and POD vectors as deflation vectors (right). We observe in this figure that the number of iterations is smaller when using POD-based deflation vector, as shown in Table 5.13.

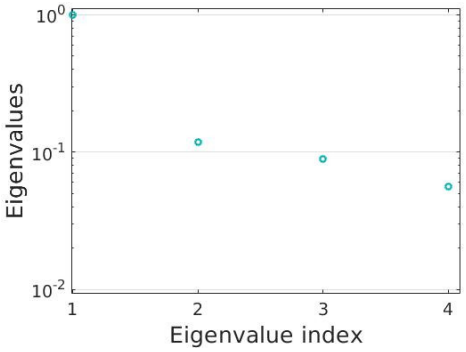


Figure 5.10: *Eigenvalues of* $\mathbf{R}$*, full SPE 10.*

p	$\sum_{j=1}^{p} \lambda_j$	$\alpha = \frac{\sum_{j=1}^{p} \lambda_j}{\sum_{j=1}^{m} \lambda_j}$
[1]	1	0.79
[1,2]	1.12	0.88
[1,2,3]	1.21	0.95
[1,2,3,4]	1.27	1

Table 5.14: *Information contained in the* $\lambda's$.

Additionally, we observe that the first approximation is more accurate $\mathcal{O}(10^{-4})$. This implies that only a few iterations are needed when using POD vectors to achieve this tolerance. Therefore, further acceleration is achieved when using POD-basis deflation vectors instead of snapshots.

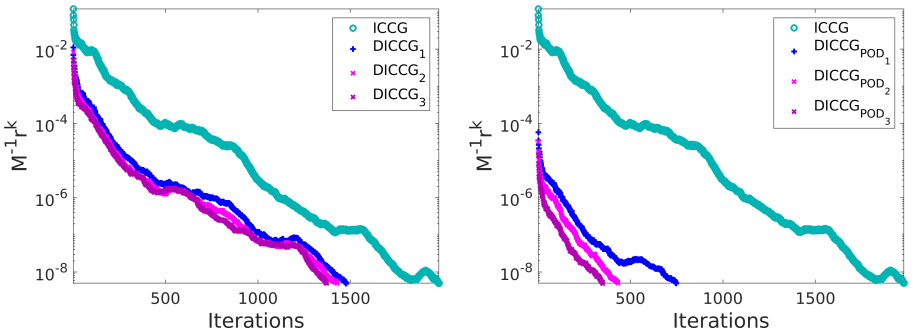


Figure 5.11: *Relative preconditioned residual, full SPE, left: snapshots, right: POD basis vectors as deflation vectors.*

5.1.2 Non-homogeneous Dirichlet boundary conditions

In this section, we simulate flow through a reservoir containing four wells, two injectors, and two producers, with pressures P = 200 bars and P = -200 bars. In the boundaries, we impose a pressure drop in the x-direction (Dirichlet boundary conditions), where the right boundary has a pressure of $P_R = 100$ bars, the left $P_L = 0$ bars, and for the y-direction, we set homogeneous Neumann boundary conditions. We study a layered problem (see Figure 5.12) and the SPE 10 model (see Figure 5.15), with a fluid presenting the same characteristics as in the previous examples.

From Lemma 4.1.5, when solving a system with non-homogeneous Dirichlet boundary conditions, a full description of the system is obtained by collecting one snapshot

for each well present in the domain, together with one snapshot accounting for the boundary conditions.

We study a reservoir containing four wells; thus, the collection of five snapshots is required. The first four snapshots ($x_1 - x_4$) are obtained setting only one well different from zero, taking no-flux boundary conditions on the $y-$direction and homogeneous Dirichlet conditions on the $x-$direction boundaries. A fifth snapshot is obtained setting all the wells to zero and setting a pressure drop in the x-direction ($\mathbf{x}_b$).

We study the case when all the wells have values different from zero and non-homogeneous Dirichlet conditions are imposed as in the fifth snapshot ($\mathbf{x}$). A summary of the snapshots is presented below in Table 5.15, for which the given bhp in the wells is $P = 200$ bars, and the pressure in the reservoir is set as $\mathrm{P}_0 = 0$ bars.

The performance of the ICCG to the DICCG methods is compared for various choices of deflation vectors. The stopping criterion for the snapshots is $5 \cdot 10^{-12}$ and for the studied methods is $5 \cdot 10^{-7}$.

The snapshots are obtained with the ICCG method with a tolerance of $5 \cdot 10^{-12}$, and used as deflation vectors, together with the POD basis vectors obtained from this set of snapshots, the deflation subspace matrix is constructed as follows:

	P_1	P_2	P_3	P_4
Bc: P(x=1)=P(x=nx)=0				
$\mathbf{x}_1$	-P	P_0	P_0	P_0
$\mathbf{x}_2$	P_0	P	P_0	P_0
$\mathbf{x}_3$	P_0	P_0	P	P_0
$\mathbf{x}_4$	P_0	P_0	P_0	-P
Bc: P(x=1)=0, P(x=nx)=P				
$\mathbf{x}_b$	-P	-P	-P	-P
$\mathbf{x}$	-P	P	P	P

Table 5.15: *Snapshots configurations,* $\frac{\partial P(y=1)}{\partial n} = \frac{\partial P(y=ny)}{\partial n} = 0$.

1. Five linearly independent snapshots (DICCG$_5$), $\mathbf{Z} = [\mathbf{x}_1 \quad ... \quad \mathbf{x}_5]$.

2. Five POD basis vectors (DICCG$_{POD}$), $\mathbf{Z} = [\psi_1 \quad ... \quad \psi_5]$.

Layered problem. In this section we study a layered problem for various permeability contrast between the layers. The wells location and permeability field are presented in Figure 5.12. The pressure field of the snapshots ($\mathbf{x}_{1:4,b}$) as well as the pressure field of the studied system ($\mathbf{x}$) are presented in Figure 5.13 for a contrast on permeability layers of $c_\sigma = 10^1$.

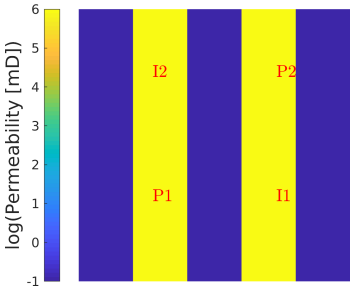


Figure 5.12: *Permeability field and wells location, layers x direction.*

In Table 5.16, we present the condition number of the system matrix ($\mathbf{A}$), the preconditioned matrix ($\mathbf{M}^{-1}\mathbf{A}$), and the effective condition number of the deflated system ($\mathbf{P}\mathbf{M}^{-1}\mathbf{A}$) for various c_σ's, where we observe that the condition number of the matrix increases when we increase c_σ.

After preconditioning and deflation, the condition number (κ) is very similar $\mathcal{O}(10^2)$ for all cases, and the effective condition number of the deflated system is approximately one third of the preconditioned system $\kappa_2(\mathbf{M}^{-1}\mathbf{A}) \sim 3 \times \kappa_{eff}(\mathbf{M}^{-1}\mathbf{P}\mathbf{A})$. This implies that the preconditioned system is not ill conditioned anymore.

c_σ	$\kappa_2(\mathbf{A})$	$\kappa_2(\mathbf{M}^{-1}\mathbf{A})$	$\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})$	
			Deflation subspace matrix $\mathbf{Z}$	
			$[\mathbf{x}_1, ..., \mathbf{x}_5]$	$[\psi_1, ..., \psi_5]$
10^1	3e+03	70	21	21
10^3	7e+04	76	21	21
10^5	6e+06	76	21	21
10^7	6e+08	76	21	21

Table 5.16: *Condition number of the solved systems, various values of* c_σ.

The number of iterations required to achieve convergence for the studied methods is presented in Table 5.17 for the ICCG and DICCG methods, the latter using 5 snapshots and 5 POD vectors as deflation vectors. We note that the number of iterations required for the ICCG method does not change considerably when changing the contrast between permeability layers, and we also note that the deflation method with five linearly independent vectors (DICCG$_5$) or five POD basis vectors (DICCG$_{P_5}$) requires only one iteration to converge.

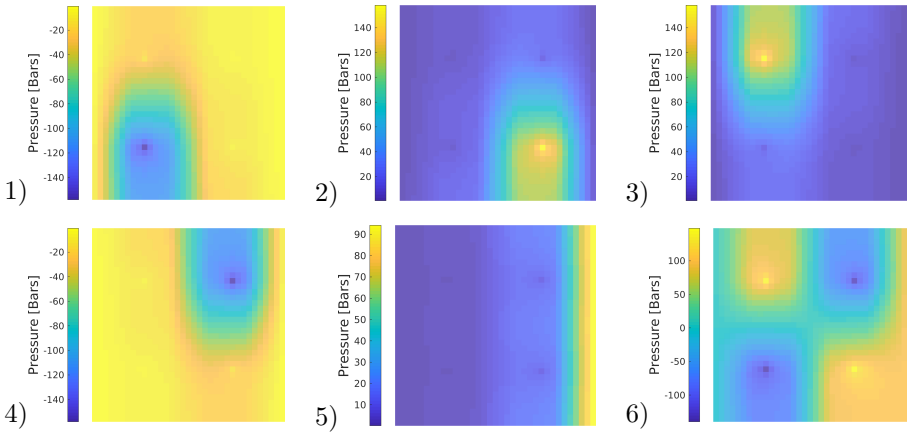


Figure 5.13: *Pressure field of the five snapshots and the studied problem,* $c_\sigma = 10^7$.

Figure 5.14 presents the relative preconditioned residual $M^{-1}r_k$, the relative residual r_k, and the relative true error e_k (see Table 5.1), for $c_\sigma = 10^1$ and $c_\sigma = 10^7$.

We note that for $c_\sigma = 10^1$ the expected accuracy is reached in all cases, but for $c_\sigma = 10^7$ the true error e_k is around 10^{-4} when using the ICCG method, contrary to the expected accuracy of 10^{-7}. Therefore, the solution is not accurate enough. In contrast, the DICCG method achieves the prescribed accuracy after the first iteration, making it a more accurate approximation.

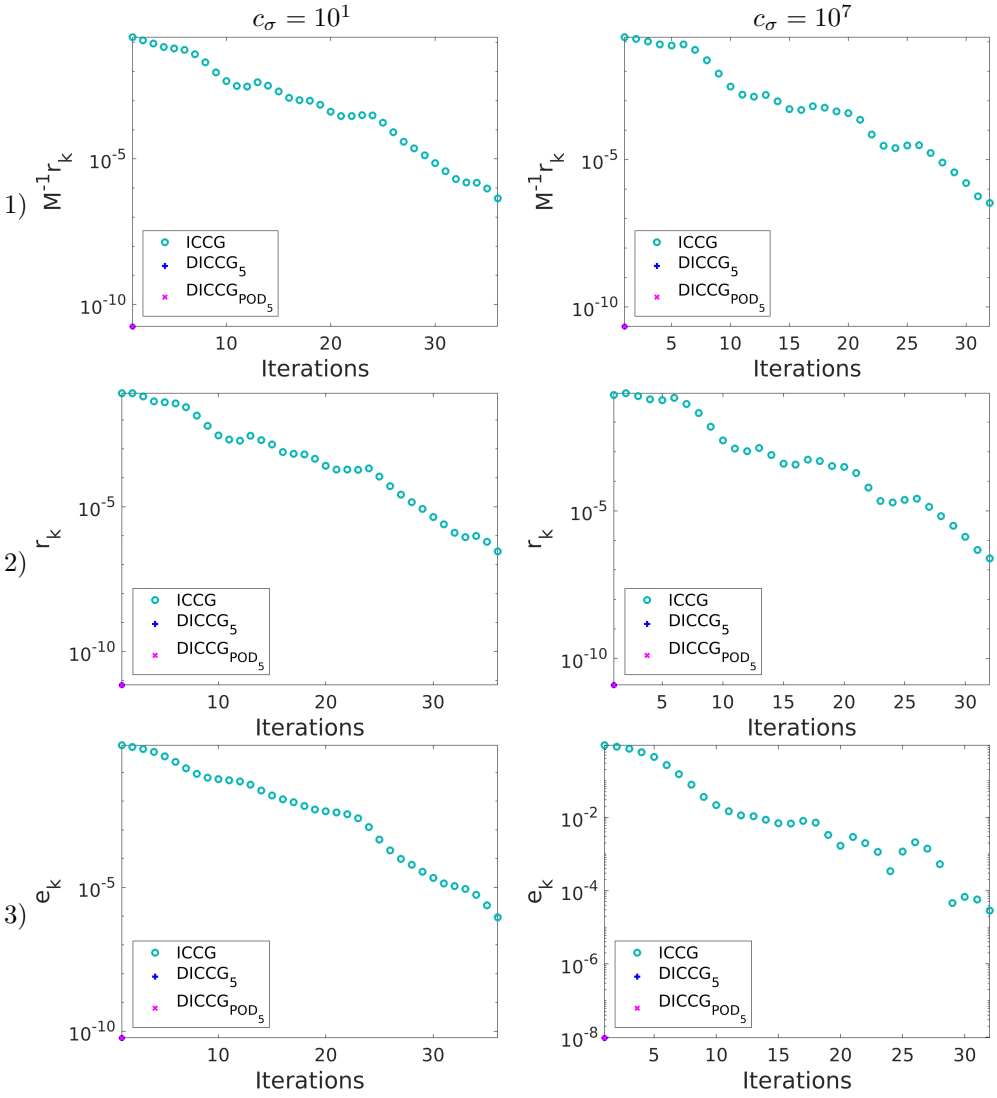


Figure 5.14: *1) Relative preconditioned residual $M^{-1}r_k$, 2) relative residual r_k, 3) relative true error e_k, left: $c_\sigma = 10^1$, right: $c_\sigma = 10^7$, Tol $= 5 \cdot 10^{-7}$.*

Next, we compare different choices of five deflation vectors, for a problem with $c_\sigma = 10^1$. The selected vectors are recycled deflation vectors (DICCG), using the five linearly independent snapshots of this problem, five POD basis obtained from the *l.i.* snapshots, five subdomain snapshots, and five eigenvalues of the system matrix, and the preconditioned system matrix. The number of iterations and the effective condition number of the deflation method using the various choices of deflation vectors are presented in

c_σ	ICCG	DICCG	
		5	5 POD
10^1	38	1	1
10^3	37	1	1
10^5	33	1	1
10^7	33	1	1

Table 5.17: *Number of iterations, various values of c_σ.*

Table 5.18.

We note that the number of iterations required when using subdomain-based and eigenvalues as deflation vectors is considerably larger than using *l.i.* snapshots or POD basis vectors, even if the condition number is similar, which is the case with subdomain-based and eigenvalues of the preconditioned system.

From this experiments, we can conclude that, when using Dirichlet boundary conditions, five linearly independent vectors are required as deflation vectors to converge in one iteration, where one of them corresponds to the boundary conditions, as expected from Lemma 4.1.5. Furthermore, we observe that this choice requires less iterations than the other choices of deflation vectors, as in the previous case.

Method	Iterations	$\kappa_2(\mathbf{M}^{-1}\mathbf{A})/\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})$
ICCG	38	14
DICCG	1	12
DICCG_{POD}	1	12
DICCG_{SD}	31	18
$\text{DICCG}_{eigs(\mathbf{A})}$	35	38
$\text{DICCG}_{eigs(\mathbf{M}^{-1}\mathbf{A})}$	23	11

Table 5.18: *Number of iterations and condition number for various choices of* $\mathbf{Z}$*, five deflation vectors.*

SPE 10 benchmark. In this section, we study the SPE 10 model containing four wells, two injectors and two producers, and the same non-homogeneous Dirichlet boundary conditions as in the previous example. For this test case, we study diverse choices of deflation vectors and different grid sizes: 16 x 56, 30 x 110, 46 x 166 and 60 x 220; furthermore, we study the full SPE 10 model containing 60 x 220 x 85 cells (3D model). The wells' location can be observed in Figure 5.15.

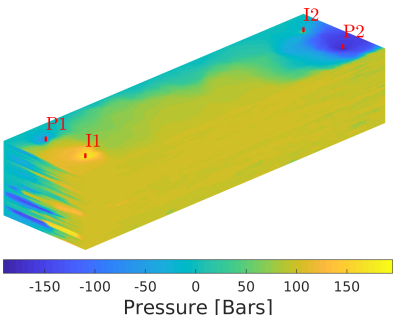


Figure 5.15: *Pressure field for the full SPE 10 model.*

The snapshots are obtained by solving the system with different well configurations as presented in Table 5.15, from where the first five snapshots, used as deflation vectors, are obtained with ICCG and tolerance of $Tol_{ICCG} = 5 \cdot 10^{-12}$. The resulting system is solved with the DICCG method and compared with the ICCG method using a stopping criterion of $Tol = 5 \cdot 10^{-7}$. The resulting pressure field is presented in Figure 5.15 for the full model .

We study the performance of the deflation method when using five deflation vectors selected in different ways for the coarse grid problem, 16 x 56 cells. We compare the different choices of deflation vectors, the number of iterations and the condition number of the studied cases in Table 5.19.

The condition number of the original system matrix is $\kappa_2(\mathbf{A}) = 2 \cdot 10^6$, after preconditioning, we observe a reduction to 14, leading to fewer iterations to reach convergence, in this case, 30. As the condition number of the preconditioned system is already small, only a minor reduction is achieved when using deflation; it is reduced to 12 when using five deflation vectors and five POD-based deflation vectors and to 14 and 6 when using five eigenvectors of the original system or the preconditioned system.

Method	Iterations	$\kappa_{eff}(\mathbf{M}^{-1}\mathbf{PA})$
ICCG ($\kappa_2(\mathbf{M}^{-1}\mathbf{A})$)	30	14
DICCG	1	12
DICCG$_{POD}$	1	12
DICCG$_{eigs(\mathbf{A})}$	29	14
DICCG$_{eigs(\mathbf{M}^{-1}\mathbf{A})}$	18	6

Table 5.19: *Number of iterations and condition number for various choices of* $\mathbf{Z}$*, four deflation vectors.*

Regarding the number of iterations, using the five *l.i.* snapshots or the POD-basis obtained from these snapshots leads to convergence in one iteration. However, using the eigenvectors of the system matrix, we do not observe a considerable reduction, while using the eigenvectors of the preconditioned matrix we reduce the number of iterations to one third. Hence, as in the previous cases, the most significant reduction is achieved with the five *l.i.* snapshots or the five POD basis vectors obtained from these snapshots.

Grid size	16 x 56	30 x 110	46 x 166	60 x 220	60 x 220 x 85
ICCG	30	57	92	110	830
DICCG	1	1	1	1	1
DICCG$_{POD}$	1	1	1	1	1

Table 5.20: *Number of iterations for various grid sizes.*

To study the performance with respect to the size, we present the number of iterations required for convergence of the studied methods in Table 5.20. As in the previous cases, increasing the size of the problem results in an increment on the number of iterations for the ICCG method; however, for the DICCG method, the only one iteration is preserved.

	Iterations		Percentage	
ICCG	830		100	
p	$\{\mathbf{x}_i\}$	POD	$\{\mathbf{x}_i\}$	POD
1	823	817	99	98
2	782	784	94	94
3	786	756	94	91
4	788	691	94	83
5	1	1		

Table 5.21: *Number of iterations for various p. Full SPE 10, Dirichlet boundary conditions.*

Next, the number of deflation vectors when using *l.i.* snapshots and POD basis vectors for the full model is varied. The number of iterations required to converge for both choices of deflation vectors is presented in Table 5.21. Note that this number is smaller when using POD vectors than when using snapshots, being 83% ICCG iterations when using four POD-based deflation vectors, and 94% with four *l.i.* deflation vectors. This result indicates that each POD basis vector contains more information than the *l.i.* vectors. However, if only one deflation vector is used, only a small gain is achieved.

In Figure 5.16 we present the eigenvectors of the correlation matrix constructed with the snapshots, we note that the second and third eigenvalues of the correlation matrix are smaller than the first one; however, the difference is not significant. Thus, the corresponding eigenvectors are essential to achieve the optimal performance of the deflation method.

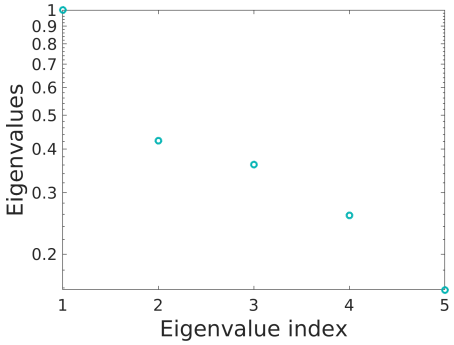


Figure 5.16: *Eigenvalues of* $\mathbf{R}$*, full SPE 10.*

p	$\sum_{j=1}^{p} \lambda_j$	$\alpha = \frac{\sum_{j=1}^{p} \lambda_j}{\sum_{j=1}^{m} \lambda_j}$
1	1	0.45
2	1.4	0.64
3	1.8	0.81
4	2	0.93
5	2.2	1

Table 5.22: *Information captured in the* $\lambda's$*.*

We present the relative residual of the deflation method in Figure 5.10 for snapshots as deflation vectors (left), and POD vectors as deflation vectors (right). For this example, the number of iterations only has a small decrease when using one to four vectors (see Table 5.13). We note that the decrease in the number of iterations is slightly larger when using POD vectors; however, it is not bigger than 15%.

The first ICCG iteration gives a residual of 0.187, while the first DICCG$_{POD}$ iterations go from 0.025 to 0.07, which means that the first DICCG iteration is slightly better (See Figure 5.17). However, the gain is not as large as the one observed for the first iteration of the case with homogeneous Neumann boundary conditions (See Figure 5.11). If we observe Table 5.22, we note that selecting four POD basis vectors we capture 93% of the information, while in the previous case we captured 95% with three (See Table 5.14).

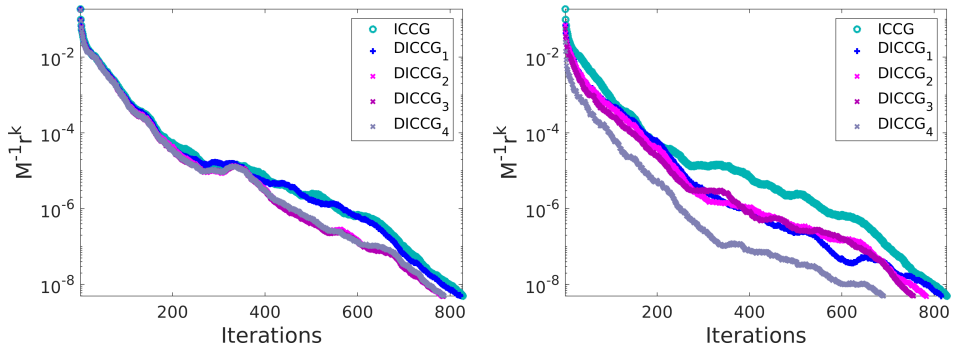


Figure 5.17: *Relative preconditioned residual, full SPE 10, left: snapshots, right: POD basis vectors as deflation vectors.*

Concluding remarks. Chapter 2 introduces the POD-based deflation method for the acceleration of linear systems, together with some theoretical results of the expected behavior of the deflation method when choosing different deflation vectors. In this chapter, we performed several numerical experiments to verify this theory.

The proposed *l.i.* set and the POD basis vectors as deflation vectors resulted in convergence in one iteration for the DICCG method, regardless of the studied case. Besides, for some cases, the approximation obtained with this method was more accurate than the one obtained with the ICCG method. However, the performance of the DICCG method showed a dependence on the accuracy of the snapshots.

The proposed deflation vectors showed better performance when compared with the usual choices. For problems with homogeneous Neumann boundary conditions, selecting subdomain vectors as deflation vectors reduced the ICCG iterations to 27%, and using eigenvectors to 21% for the layered problem. In the SPE 10 problem. the reduction with eigenvalues was 77% (eigenvectors of $\mathbf{A}$) and 54% (eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$).

For the cases with Dirichlet boundary conditions, the reduction was 81% ICCG iterations for subdomain vectors, and eigenvectors of ($\mathbf{A}$), and 74% for eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$ for the layered problem, while for the SPE 10 the reduction was 96% (eigenvectors of $\mathbf{A}$) and 60% (eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$).

If the number of deflation vectors was reduced, the number of ICCG iterations was reduced to 62% and 94% removing one *l.i.* vector, and to 33% and 83% selecting the same number of POD basis vectors, for the full SPE 10 problems with Neumann and Dirichlet boundary conditions, respectively. If only one deflation vector was used, the reduction was $\sim 99\%$ for the POD basis vectors and the *l.i.* vectors for the Dirichlet problem, and 63% for the POD basis vectors and 84% for the *l.i.* vectors. Thus, better performance was achieved when using POD basis vectors as deflation vectors, and for problems with Neumann boundary conditions.

Additionally, the POD basis vectors are linearly independent; and therefore, using them as deflation vectors avoids difficulties that can appear when using a linearly dependent set of snapshots.

The contrast between permeability layers is reflected in the condition number of the system matrix; when increasing the contrast, the largest eigenvalues become larger, and the condition number increases accordingly. After preconditioning, the influence of the contrast goes to the smallest eigenvalues. The deflation vectors proposed in this work, effectively treated these eigenvalues, they were set to zero, and their influence was removed from the iterative process.

For the examples studied in this section, the set of *l.i.* vectors containing all the system information is known and collecting them and reducing them with POD only results in extra work. However, the *l.i.* set is usually unknown. For those cases, the most significant advantages of combining the POD and deflation methodologies can be exploited as is the case for time-dependent problems, studied later in Chapter 6 and 7, and when information can be reused.

Chapter 6

Numerical experiments 2: Compressible single-phase reservoir simulation

In Chapter 5, we presented some numerical experiments that show the performance of the POD-based deflation methodology. There, we studied the simulation of a single-phase incompressible fluid. The studied system was time independent, and, therefore, the only option for collecting snapshots was via recycling (see Section 1.4).

In this chapter, we investigate the performance of the method for a compressible single-phase case. Here, the problem is time-dependent, and as such, it has to be solved for various time-steps. Thus, the moving window approach is tested in this chapter.

With this approach, the previous time steps are used as snapshots in a recycling scheme, or to compute a POD basis. The snapshot or the POD basis are then used as deflation vectors to solve the current linear system. As before, we test the methodology for an *academic* layered problem with various contrast between permeability layers and for the SPE 10 benchmark.

This chapter is based on:

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

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

6.1 Compressible fluid

Model. We study a compressible fluid, flowing through a porous media with a constant porosity field of $\phi = 0.3$ (incompressible rock). The fluid's viscosity is constant with value $\mu = 5$ $[cp]$, and the relation between the pressure and the density is given by Equation (2.7). We assume a constant fluid compressibility of $c_f = 1\times10^{-3}$ [bars^{-1}], and a reference fluid density of $\rho_r = 850$ $[kg/m^3]$ at a pressure of 200 [bars].

With these parameters, integration of Equation (2.7) results in a pressure dependent function for the density, given by:

$$\rho(p) = \rho_r e^{c_f(\mathbf{p}-\mathbf{p}_r)} = 850[kg/m^3]e^{1\times10^{-3}(\mathbf{p}-200)}. \tag{6.1}$$

For the modeling, we make use of Darcy's law and the mass balance equation (see Section 2.1), resulting in a non-linear system given by (see Equation (2.12)):

$$\phi\frac{\partial\rho(p)}{\partial t} - \nabla\cdot(\rho(p)\lambda) = \rho q,$$

disregarding gravity terms and with $\rho(p)$ as given in Equation (6.1).

We study an academic layered problem with various sizes and various contrast between permeability layers c_σ, and the full SPE 10 benchmark presenting a contrast in permeability coefficients of 3×10^7.

We prescribe homogeneous Neumann boundary conditions and we include five wells, as described in Section 5.1.1. The initial pressure of the reservoir is set to 200 [bars]. The bhp in the corner wells is 100 [bars] and in the central well is 600 [bars]. The simulation was performed for 156 days for 52 time steps and a step size of 3 days.

Snapshots. In this set of problems, we use recycling and the moving window approach for the collection of the snapshots, as introduced in Section 4.2.

POD-based deflation vectors. A POD basis is obtained at each time step from the ten most recently-collected snapshots. This basis is used as deflation vectors to solve the current time step. The number of POD deflation vectors is specified for each problem.

Solvers. Equation (2.12) is linearized via the Newton-Raphson (NR) method and the spatial discretization is the same as in Chapter 5. The resulting linear system is given by (see Equation (2.31)):

$$\mathbf{J}(\mathbf{p}^k)\delta\mathbf{p}^{k+1} = -\mathbf{F}(\mathbf{p}^{k+1};\mathbf{p}^0).$$

After linearization, the resulting system is solved with the ICCG method for the first time steps, and with the DICCG method for the rest of the time steps. For the deflation method, we use snapshots and POD basis vectors as deflation vectors.

A summary of the procedure is presented in Algorithm 2. The simulation, except for the linear solvers, is performed with MRST. Automatic Differentiation (AD) is used for the NR loop [7].

The tolerance of the NR method and the linear solvers is $\epsilon = 10^{-5}$. For the linear solvers, the preconditioned relative residual computed at each iteration, $M^{-1}r_k$, is employed as stopping criterion (see Table 5.1).

For all the experiments, only the first time step requires more than two NR iterations. Hence, we solely study the behavior of the linear solvers during the first two NR iterations.

Layered problem In this section, we study the academic layered problem presented in Section 5.1.1, (see Figure 5.1). The first layer has a permeability of $\sigma_1 = 1$ [mD], and the permeability of the second layer is varied, taking the following values $\sigma_2 = 10, 100, 100$ [mD]. Therefore, the contrast between the layers $c_\sigma = \frac{\sigma_2}{\sigma_1}$ is 10^1, 10^2 and 10^3. The length of the domain is 70 [m] and the grid contains 105 cells in each dimension.

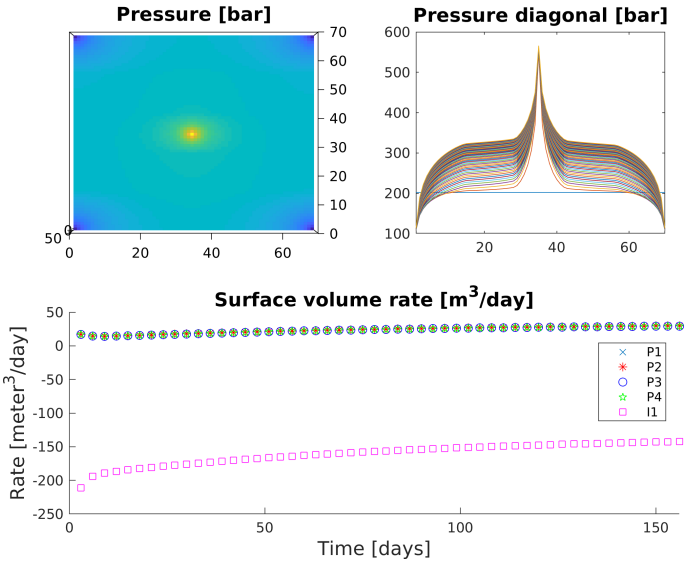


Figure 6.1: *Solution of the compressible problem solved with the ICCG method for a layered problem with a contrast between permeability layers of* 10^1.

In Figure 6.1, the solution obtained with the ICCG method is presented for a contrast in permeability layers of 10^1. In the lower figure, we observe the surface volume rate for the five wells during the simulation. The upper left figure represents the pressure field at the final time step.

The upper right figure represents the pressure across the diagonal joining the (1,1) and (35,35) grid cells for all the time steps. We observe the initial pressure, 200 [bars], across this diagonal and the evolution of the pressure field in time.

As mentioned before, for each time step, the previous ten solutions are used as snapshots to compute the POD basis. The eigenvalues of the snapshot correlation matrix $\mathbf{R} = \frac{1}{10}\mathbf{X}\mathbf{X}^T$ constructed with the previous ten time steps are presented in Figure 6.2 for the 20^{th} time step, for a contrast between permeability layers of 10.

We observe that six eigenvalues are larger than the rest. Then, we use the eigenvectors corresponding to these six eigenvalues as deflation vectors, DICCG$_{POD_6}$. When

$c_\sigma = 10^2$ and $c_\sigma = 10^3$ we used seven eigenvalues instead.

In Table 6.1 and Table 6.2, we compare the number of iterations necessary to reach convergence with the ICCG method (second column) and the deflation method using snapshots and POD basis vectors.

For this examples, we use the moving window approach; therefore, it is required to compute the first d snapshots with ICCG (fourth column), the rest of the time steps are computed with DICCG (fifth column).

The total number of iterations needed to perform the DICCG method (d time steps computed with ICCG + Total-d computed with DICCG) are presented in the sixth column. In the last column, we compute the percentage of DICCG iterations with respect to the total ICCG iterations.

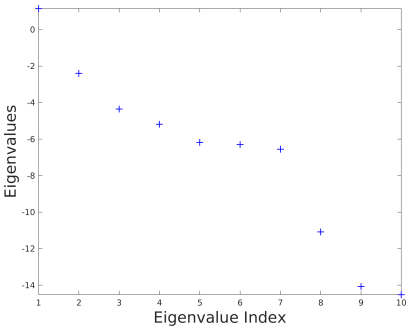


Figure 6.2: *Eigenvalues of the data snapshot correlation matrix, 20-th time step, $c_\sigma = 10^1$.*

Cumulative linear iterations in the 1^{st} NR Iteration						
c_σ	Total ICCG	Deflation method	ICCG Snapshots	DICCG	Total ICCG +DICCG	% of total ICCG
10^1	1026	DICCG_{10}	160	84	224	21
		DICCG_{POD_7}	160	84	224	21
10^2	2468	DICCG_{10}	430	210	640	26
		DICCG_{POD_7}	430	210	640	26
10^3	2815	DICCG_{10}	580	210	790	28
		DICCG_{POD_7}	580	252	832	29

Table 6.1: *Comparison of number of iterations between the ICCC and DICCG methods during the first NR iteration for various c_σ.*

Cumulative linear iterations in the 2^{nd} NR Iteration						
c_σ	Total ICCG	Deflation method	ICCG Snapshots	DICCG	Total ICCG +DICCG	% of total ICCG
10^1	229	DICCG_{10}	180	26	206	90
		DICCG_{POD_7}	180	24	204	90
10^2	1285	DICCG_{10}	540	132	672	52
		DICCG_{POD_7}	540	209	749	58
10^3	982	DICCG_{10}	680	60	740	75
		DICCG_{POD_7}	680	63	743	76

Table 6.2: *Comparison of number of iterations between the ICCC and DICCG methods during the second NR iteration for various c_σ.*

We study the deflation method with ten snapshots as deflation vectors, DICCG$_{10}$, and seven POD basis vectors as deflation vectors, DICCG$_{POD_7}$.

For the first NR iteration, we observe a significant reduction in the number of linear iterations when using DICCG. For $c_\sigma = 10^1, 10^2$, the DICCG method requires $23 - 29\%$ ICCG iterations. When $c_\sigma = 10^3$ this percentage is reduced to 17%.

For the second NR iteration (see Table 6.2), we also observe a significant reduction on the number of linear iterations, achieved by the use of deflation. A reduction to 26% and 38% of the ICCG linear iterations is achieved for $c_\sigma = 10^1$. When $c_\sigma = 10^2$ we require 28% and 33% of the ICCG iterations and for $c_\sigma = 10^3$, we require 23% and 29% of the ICCG iterations.

SPE 10 model We study the complete SPE 10 benchmark presented in Section 5.1.1. The eigenvalues of the snapshot correlation matrix are presented in Figure 6.3.

We observe 4 eigenvalues larger than the rest, which implies that most of the information is contained in these eigenvalues. Therefore, we study the deflation method with 10 snapshots as deflation vectors and 4 POD basis vectors.

The number of iterations is presented in Table 6.3 and Table 6.4 for the first and second NR iterations, respectively. With the deflated methods DICCG$_{10}$ and DICCG$_{POD_4}$, for the first NR iteration, we only need to perform 28% and 32% of the ICCG linear iterations.

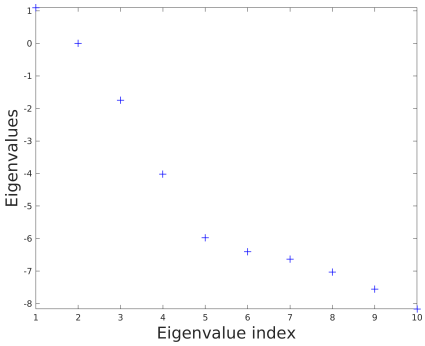


Figure 6.3: *Eigenvalues of the data snapshot correlation matrix, 20-th time step, full SPE 10 benchmark.*

For the second NR iteration, the reduction is more significant, requiring 20% of the ICCG linear iterations. We also observe that for the first NR iteration we need 1770 linear iterations to compute the ten initial snapshots (computed with ICCG) and 1134 to compute the solution of the rest of the solutions (computed with DICCG).

For the second NR iteration, the number of linear iterations is 1830 for the ten initial snapshots and 200 for the deflated methods. Thus, that the largest amount of work is carried out for the computation of the snapshots with the ICCG method, which is more evident for the second NR iteration.

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

Table 6.3: *Average number of linear iterations for the first NR iteration, full SPE 10 benchmark.*

2^{nd} NR Iteration					
Total ICCG	Deflation method	ICCG Snapshots	DICCG	Total ICCG +DICCG	% of total ICCG
10231	$DICCG_{10}$	1830	200	2030	20
10231	$DICCG_{POD_4}$	1830	200	2030	20

Table 6.4: *Average number of linear iterations for the second NR iteration, full SPE 10 benchmark.*

Concluding remarks For the compressible case, we propose the use of solutions of previous time steps, *snapshots*, and POD basis vectors obtained from these snapshots as deflation vectors. The collection of the snapshots was made using a moving window approach.

With the DICCG we reduce the number of iterations up to 20% of the number of ICCG iterations with only a small increase in the number of flops. The best performance was achieved using four deflation vectors, for which each DICCG iteration needs around 1.4 times the number of flops required with the ICCG method (see Table 4.1).

In these experiments, we used the moving window approach, and we observed that a large amount of work is carried out by the computation of the snapshots. In the next chapter, we compare the use of a moving window and a training phase approach for the solution of two-phase subsurface flow simulation.

Chapter 7

Numerical experiments 3: Incompressible two-phase reservoir simulation

While dealing with two fluid phases inside a reservoir, the presence of one phase affects the dynamics of the second one. Then, the pressure of each phase is influenced not only by the medium but also by the other fluid. This influence is known as capillary pressure, and the larger the capillary pressure is, the more difficult the solution of the system becomes.

Many of the recently developed acceleration techniques are not yet able to completely overcome this influence [11, 49] and new ways to solve it are required. In this chapter, we present some examples that demonstrate the applicability of the POD-based deflation methodology for the solution of two-phase problems.

7.1 Incompressible two-phase simulation

In Chapter 2, we introduced the equations describing single- and two-phase flow inside a reservoir, together with some discretization schemes. In particular, we introduced the fractional flow formulation, that help us to decouple the pressure equation from the saturation equation.

This chapter is based on:

G.B. Diaz Cortes, J.D. Jansen, and C. Vuik. On The Acceleration Of Ill-Conditioned Linear Systems: A Pod-Based Deflation Method For The Simulation Of Two-Phase Flow. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, 2018,

and additional work.

With this formulation, a sequential procedure has to be implemented (see Algorithm 1), where an elliptic pressure equation (see Equation (2.12)) is solved at each time step and it is used to update the saturation equation (see Equation (2.21)). This formulation is used throughout this chapter to simulate two-phase flow, where our main focus is on the solution of the pressure equation.

Model problem. We model waterflooding, i.e., water injection through wells and through the boundaries into a reservoir originally filled with oil. We study an academic layered reservoir and the SPE 10 benchmark [10] presenting a contrast in permeability coefficients up to $\mathcal{O}(10^7)$.

	Water	Oil	Units
μ	1	10	cp
ρ	1000	700	kg/m^3
k_r	$(S_w)^{n_w}$	$(1-S_w)^{n_{nw}}$	
C_p	$10*(1-S_w)$		bar

Table 7.1: *Fluid properties.*

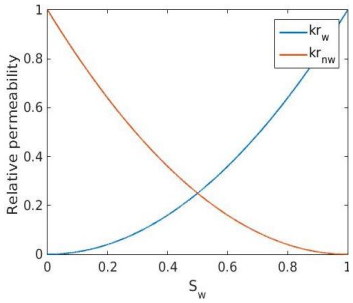


Figure 7.1: *Fluids relative permeability.*

We study the influence of capillary pressure and gravity terms on the performance of the methods, for which the relative permeability of the fluids are modeled with the Corey model.

We also investigate the influence of the capillary pressure, modeled with a linear relationship, $C_p = 10\,(1-S_w)$. The properties of the fluids are presented in Table 7.1, and the relative permeability curves are presented in Figure 7.1 for Corey coefficients of $\mathrm{n}_w = \mathrm{n}_{nw} = 2$.

The pressure's linear system and the solution of the transport equation are obtained with the Matlab Reservoir Simulation Toolbox (MRST, [7]).

Transport solver. To solve the transport equation, we use an implicit solver combined with an aggregation-based algebraic multigrid (AGMG) method [50–52] implemented in MRST .

Pressure solver. For the solution of the linear pressure equation, we implement the POD-based Deflated Preconditioned Conjugate Gradient method, with the incomplete Cholesky (IC) decomposition as preconditioner (DICCG) and we compare the results with the non-deflated method, ICCG.

As deflation vectors, we use a set of snapshots and a POD basis obtained from them. The stopping criterion is the relative preconditioned residual $\epsilon = M^{-1}r_k$, as presented in Table 5.1.

Snapshots collection and studied cases. For the collection of the snapshots, we use a moving window (MW), and a training phase (TP) approaches, as described in Section 4.2. More details are given below:

MW The p previously-computed time steps are used as snapshots. Note that, the first p time steps are computed with the ICCG method. For this problem, we study one (2D) and the 85 layers (3D) of the SPE 10 benchmark. We study cases with and without capillary pressure.

TP1 A full pre-simulation is run, randomly changing the bhp in the production wells every two time steps within a range of P = 137.5 and 275 bar (see Figure 7.11). The variations in the bhp are different for each well, but it is in the same range. The basis obtained from these snapshots is used to solve different problems:

TP1.1 *Equal bhp in the producers.* The bhp of the producers is the same and takes the values: $P = 275$ bar, an extreme value; $P = 200$ bar, an intermediate value; and $P = 400$ bar, a value outside the training phase pressure range.

TP1.2 *Different bhp in the producers.* One well has a bhp of $P_i = 20$ bar, and the rest have the same pressure as the reservoir $P_{j \neq i} = 500$ bar.

TP2 We also perform a full pre-simulation, but in this case, one well is activated during a quarter of the time steps, i.e., one well has a pressure P = 200 bar, and the rest have the same pressure as the reservoir $P_0 = 500$ bar.

TP3 Same as TP2, but the simulation is run for a shorter time, i.e., the training phase is shorter and therefore, requires less work.

The problems solved with the MW, TP2 and TP3 approaches have equal bhp in the producers. For the MW approach, the bhp is the same as for the snapshots, while for the TP2 and TP3 it is a bhp of $P = 275$ bar.

Heterogeneous permeability layers, MW approach

In this section, we study water injection into an 'academic' problem consisting of equal-sized layers with a constant porosity field $\phi = 0.2$ and different permeability values (see Figure 7.2). A set of layers with permeability $\sigma_1 = 10^1\ mD$ is followed by layers with permeability $\sigma_2 = 10^2$ or $10^7\ mD$. The contrast between permeability coefficients is given by $c_\sigma = \sigma_1/\sigma_2$.

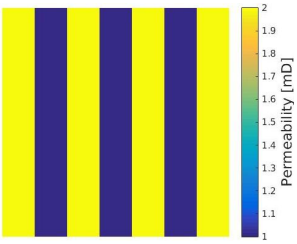


Figure 7.2: *Rock permeability*

The domain consists of a Cartesian grid of 35 x 35 cells for a 2D case, and 25 x 25 x 25 cells for a 3D case. The fluid properties are presented in Table 7.1. The first set of experiments does not consider gravity and capillary pressure terms. Later, we include capillary pressure and the 3D problem includes gravity terms. The studied test cases are:

TC1. $c_\sigma = 10^1$, no capillary pressure terms included.

TC2. $c_\sigma = 10^6$, no capillary pressure terms included.

TC3. $c_\sigma = 10^1$, capillary pressure terms included.

TC4. $c_\sigma = 10^6$, capillary pressure terms included.

We study water flooding with injection through the boundary and wells. When using wells, the setup consists of four producers, P_i, on the corners and one injector, I, in the center. The wells are controlled by prescribing the bottom hole pressure in the wells *bhp*. For the solution of the pressure equation, we implement the moving window approach.

Injection through the left boundary

Injection is performed through the left boundary at a rate of 0.4 m^3/day for the 2D case and 4 m^3/day for the 3D case. The pressure is set to zero at the right boundary and 100 bar inside the reservoir (See Table 7.2). The simulation is run for 240 days with a step size of 1 day (see Table 7.2). To collect the snapshots, we use the MW approach. We study the deflation procedure using ten previous solutions, $p = 10$, and five POD basis vectors, $p = 5$, obtained from these solutions.

Temporal parameters			Boundary conditions		
T_{steps}	240		$P_{0,x\neq(0,L_x)}$	100	bar
dT	1	day	$P_{x=L_x}$	0	bar
T_{total}	240	days	$Q_{x=0}$	0.4 (2D), 4 (3D)	m^3/day

Table 7.2: *Boundary conditions and temporal parameters.*

The pressure field and the water saturation are presented in Figure 7.3 and Figure 7.4 for various cases. We observe that the pressure is higher at the boundary where the water is injected, and it decreases towards the right boundary. We also observe that the layered pattern influences the pressure and saturation fields.

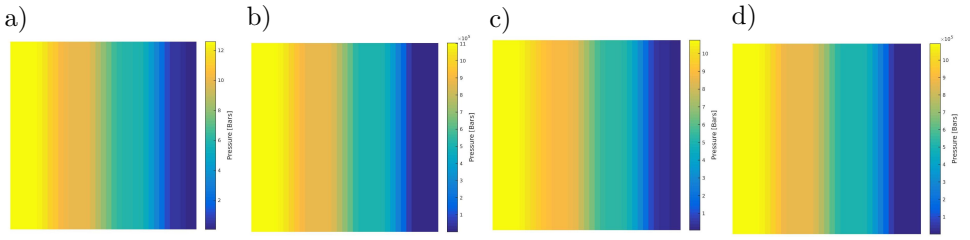


Figure 7.3: *Pressure field for the last time step, contrast between permeability values of a) TC1, b) TC2, c) TC3, d) TC4.*

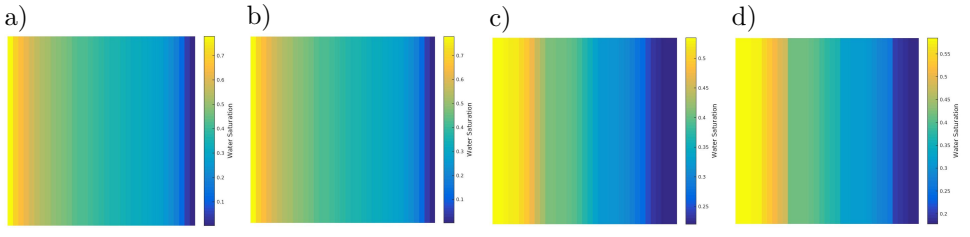


Figure 7.4: *Water saturation during the last time step, contrast between permeability values of a) TC1, b) TC2, c) TC3, d) TC4.*

The number of iterations necessary to achieve convergence is summarized in Table 7.3 for a tolerance value of $\epsilon = 5 \cdot 10^{-7}$, where the first column is the studied test case. In the second column, we present the number of deflation vectors used, p. The third column shows the number of iterations necessary to achieve convergence with the ICCG method only.

The number of iterations necessary to achieve convergence with the deflated method is presented in the sixth column. For this method, the first p time steps are computed with ICCG (fourth column), and the rest of the time steps are computed with DICCG (fifth column). In the last (seventh) column, we compute the percentage of total DICCG iterations concerning the total number of ICCG iterations.

	p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG Iterations	% of ICCG Work
2D Case: No capillary pressure included							
TC1	10	13185	447	2265	2712	21	43
TC1	5			2797	3244	25	38
TC2	10	22210	232	7763	7995	36	75
TC2	5			3887	4119	19	29
2D Case: Capillary pressure included							
TC3	10	12903	514	3842	4356	34	70
TC3	5			4035	4549	35	54
TC4	10	21918	784	3955	4739	22	45
TC4	5			4440	5224	24	37
3D Case: No capillary pressure included							
TC1	10	13128	535	2180	2715	21	43
TC1	5			2987	3522	27	41
TC2	10	32659	390	4827	5217	16	33
TC2	5			4452	4842	15	23
3D Case: Capillary pressure included							
TC3	10	12455	547	5108	5655	45	94
TC3	5			5817	6364	51	79
TC4	10	32233	1177	5821	6998	22	45
TC4	5			6833	8010	25	38

Table 7.3: *Number of iterations for various methods,* $\epsilon = 5 \cdot 10^{-7}$.

We observe a significant reduction in the number of iterations when using the DICCG, with similar trends for the 2D and 3D cases, the last one including gravity terms. For the 2D case with a stopping criterion of $\epsilon = 5 \cdot 10^{-7}$, we note a reduction to 21% and 25% of the number of ICCG iterations for $c_\sigma = 10^1$ (TC1), and to 36% and 19% for $c_\sigma = 10^6$ (TC2) and no capillary pressure included. These results are comparable if we use ten of five deflation vectors. If we include capillary pressure, this percentage increases (TC3 and TC4).

Using ten deflation vectors, we achieve reductions in the ICCG work from 94% to 33%, while using five the reduction was from 23% to 79%. Thus, better performance is achieved when using 5 deflation vectors. The reductions in the work are also larger if capillary pressure is not included.

	p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG Iterations	% of ICCG Work
2D Case: No capillary pressure included							
TC1	10	7228	93	815	908	13	29
TC1	5			959	1052	15	24
TC2	10	18481	120	1486	1606	9	20
TC2	5			1999	2119	11	19
2D Case: Capillary pressure included							
TC3	10	7207	230	1011	1241	17	41
TC3	5			1060	1290	18	29
TC4	10	19191	254	2238	2492	13	30
TC4	5			1756	2010	10	18
3D Case: No capillary pressure included							
TC1	10	6503	144	620	764	12	24
TC1	5			671	815	13	19
TC2	10	25592	215	1555	1770	7	14
TC2	5			1782	1997	8	12
3D Case: Capillary pressure included							
TC3	10	6405	206	805	1011	16	32
TC3	5			927	1133	18	27
TC4	10	25100	255	2537	2792	11	23
TC4	5			2278	2533	10	16

Table 7.4: *Number of iterations for various methods,* $\epsilon = 5 \cdot 10^{-4}$.

In Table 7.4 we present the same results as in Table 7.3 for a less strict tolerance $\epsilon = 5 \cdot 10^{-4}$, sufficient for engineering purposes. We also include in the last column of this table the percentage of ICCG work during the iteration process, also plotted in Figure 7.5. The work of a method is the number of iterations multiplied by the number of flops per iteration.

We observe that the total iteration work is reduced to less than 40% in all cases. However, the best performance is achieved when using five deflation for the TC2. When using a larger tolerance $\epsilon = 5 \cdot 10^{-4}$, the percentage of iterations is further reduced (see Table 7.4).

In Figure 7.6, we present the preconditioned relative residual $M^{-1}r_k$ and the true relative error e_k of the studied methods for the studied cases during the 50-th time step when $c_\sigma = 10^6$, i.e., TC2 and TC4. We note that, after a few iterations, $M^{-1}r_k$ of the deflated method is smaller than 10^{-4}, which results in the observed better

performance when requiring this tolerance.

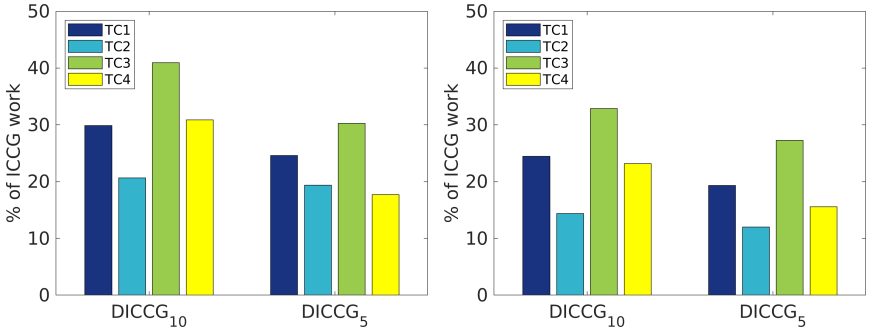


Figure 7.5: *% of ICCG work, left: 2D case, right: 3D case,* $\epsilon = 5 \cdot 10^{-4}$.

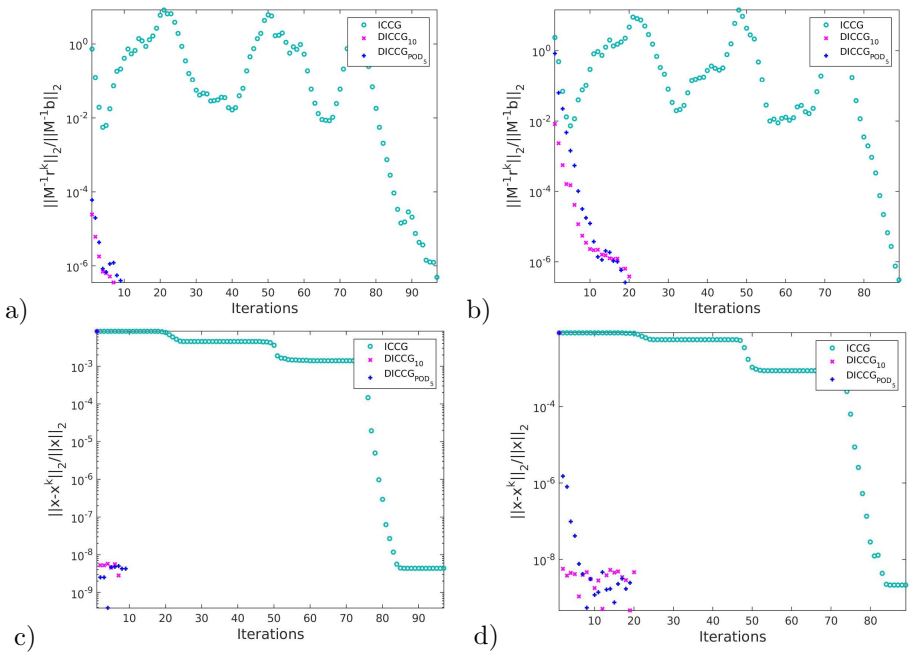


Figure 7.6: *Relative residual a) TC2, b) TC4, and true relative error c) TC2, d) TC4 for various methods.*

Comparing $M^{-1}r_k$ with e_k, we note that for the DICCG method, e_k is smaller than $M^{-1}r_k$, i.e., the solution is as accurate as expected. Furthermore, after the first iteration, e_k is already smaller than 10^{-5}.

The eigenvalues of the correlation matrix are presented in Figure 7.7 for an intermediate time step, t = 50 days, with and without capillary pressure terms for the 2D case a) $c_\sigma = 10^1$ and b) $c_\sigma = 10^6$ and the 3D case c) $c_\sigma = 10^1$. We note that about five or six eigenvalues are larger than the rest, which implies that most of the information is contained in the eigenvectors corresponding to these eigenvalues.

Thus, the performance of the deflation method with five or ten deflation vectors

is similar, which can be observed in Table 7.3 and Table 7.4. The rest of the eigenvalues are smaller than 10^{-6}; therefore, they do not have a strong influence on the performance of the method.

For the cases containing capillary pressure and $c_\sigma = 10^1$, we note that the eigenvalues are larger than for the case without capillary terms, which gives rise to the faster convergence and the detected better performance for the former case.

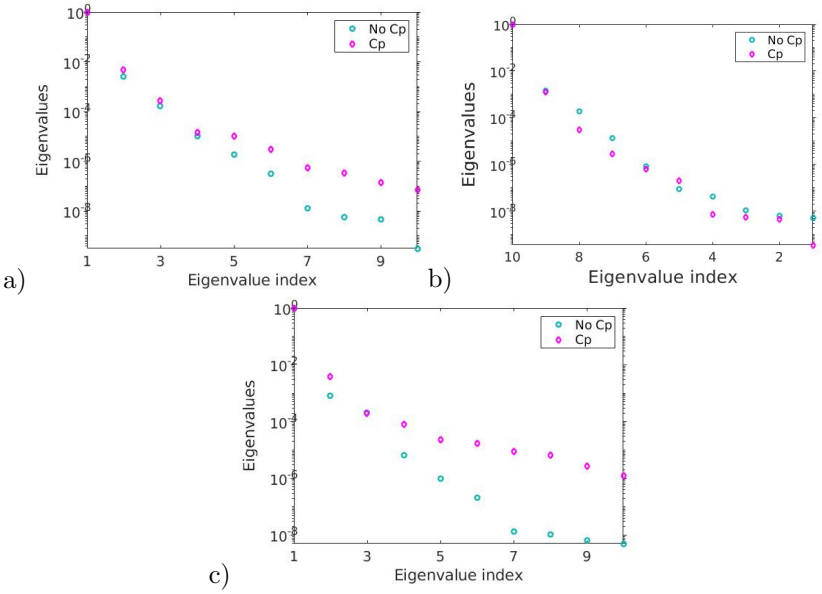


Figure 7.7: *Normalized eigenvalues of* $\mathbf{R}$*, a)* $c_\sigma = 10^1$ *(TC1 and TC3), b)* $c_\sigma = 10^6$ *(TC2 and TC4), 2D case and c)* $c_\sigma = 10^1$ *for the 3D case.*

SPE 10 benchmark

In this section, we simulate waterflooding for one layer and the full SPE 10 benchmark including gravity terms. We consider injection through the boundary and injection through wells for the 2D and 3D cases.

The wells setup consist of one injector and four producers (see Figure 5.8). The fluid properties and the capillary function used for this problems are the same as in Section 7.1 (see Table 7.1).

Injection through the left boundary, MW approach

For this test case, water is injected through the left boundary at a constant rate of 600 m^3/day to a reservoir initially filled with oil. The initial reservoir's pressure is 100 bar, and the pressure at the right boundary is set to zero bar. We simulate 240 time steps with a step size of 100 days.

For the DICCG method, we select ten snapshots (previously computed time steps) and five POD basis vectors as deflation vectors for the 2D case, and 30 snapshots and 10 POD basis vectors for the 3D case (MW approach). The number of iterations

required with the ICCG and DICCG methods is presented in Table 7.5 for a tolerance of $\epsilon = 5 \cdot 10^{-7}$ and Table 7.6 for $\epsilon = 5 \cdot 10^{-4}$.

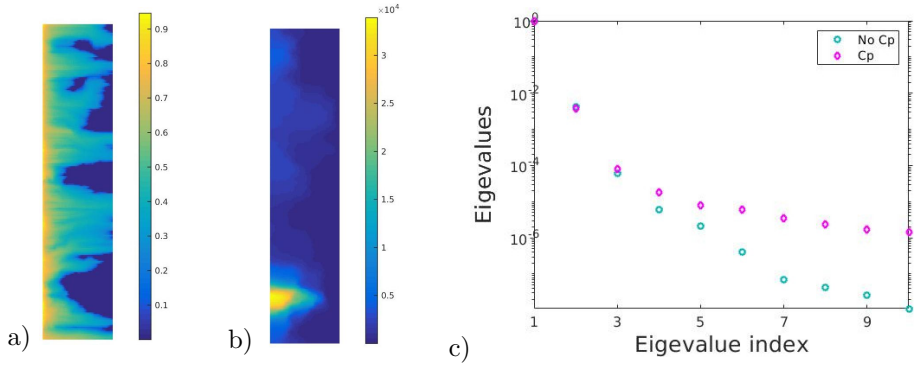


Figure 7.8: *a) Water saturation, b) Pressure field, c) Normalized eigenvalues of* $\mathbf{R}$.

For the 2D case, we observe a reduction to 33% of the number of ICCG iterations when using ten deflation vectors and to 46% when using five vectors for a tolerance value of $\epsilon = 5 \cdot 10^{-7}$. If the stopping criterion is $\epsilon = 5 \cdot 10^{-4}$, the number of DICCG iterations is reduced to around 16% of the number of ICCG iterations in both cases.

If we include capillary terms, we observe a noticeable increment in the number of iterations for the most accurate problem $\epsilon = 5 \cdot 10^{-7}$, and a small increment for the less accurate problem, $\epsilon = 5 \cdot 10^{-4}$. For the 3D case, the DICCG method requires only 25% of the number of ICCG iterations with 30 deflation vectors and to 31% with 10.

p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG
2D Case, no capillary pressure included					
10	34226	1244	10220	11464	33
5	34226	1244	14431	15675	46
2D Case, capillary pressure included					
10	34121	1249	15118	16367	48
5	34121	1249	17330	18579	54
3D Case, no capillary pressure included					
30	88502	6120	16274	22394	25
10	88502	2018	25129	27147	31

Table 7.5: *Number of iterations, various methods,* $\epsilon = 5 \cdot 10^{-7}$.

The percentage of ICCG work for the iteration process is presented in the last column of Table 7.6 and Figure 7.9, using a tolerance for the deflation method of $\epsilon = 5 \cdot 10^{-4}$. Here, we observe that the total work is reduced to less than 40%, and using fewer deflation vectors results in a better performance.

The pressure field and the water saturation during the last time step, together

with the eigenvalues of the correlation matrix are presented in Figure 7.8 for the 2D case. We observe a decrease in the eigenvalues that goes to 10^{-7} for the case without capillary pressure. When including capillary pressure, after the fifth eigenvalue it is constant at around 10^{-5}. This indicates that more information is contained in the corresponding eigenvectors when no capillary pressure is included, which is reflected in a better DICCG performance.

p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG Iter	% of ICCG Work
No capillary pressure included						
10	12220	418	1395	1813	15	34
5			1657	2075	17	28
Capillary pressure included						
10	12160	419	1532	1951	16	37
5			1712	2131	18	29

Table 7.6: *Number of iterations, various methods, 2D case,* $\epsilon = 5 \cdot 10^{-4}$.

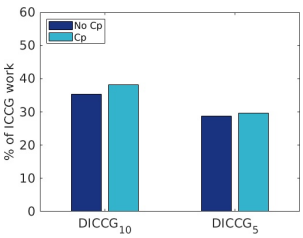


Figure 7.9: *% of ICCG work, 2D case,* $\epsilon = 5 \cdot 10^{-4}$.

The relative preconditioned residual $M^{-1}r_k$ and the true relative error e_k are presented in Figure 7.10 for the 2D case. The overall behavior of the methods is similar when computing $M^{-1}r_k$ or e_k. We note that the first DICCG iteration gives already a significant reduction, which is larger than the one achieved with ICCG for both, $M^{-1}r_k$ and e_k. For the latter one, the first iteration is smaller than $\epsilon = 5 \cdot 10^{-4}$.

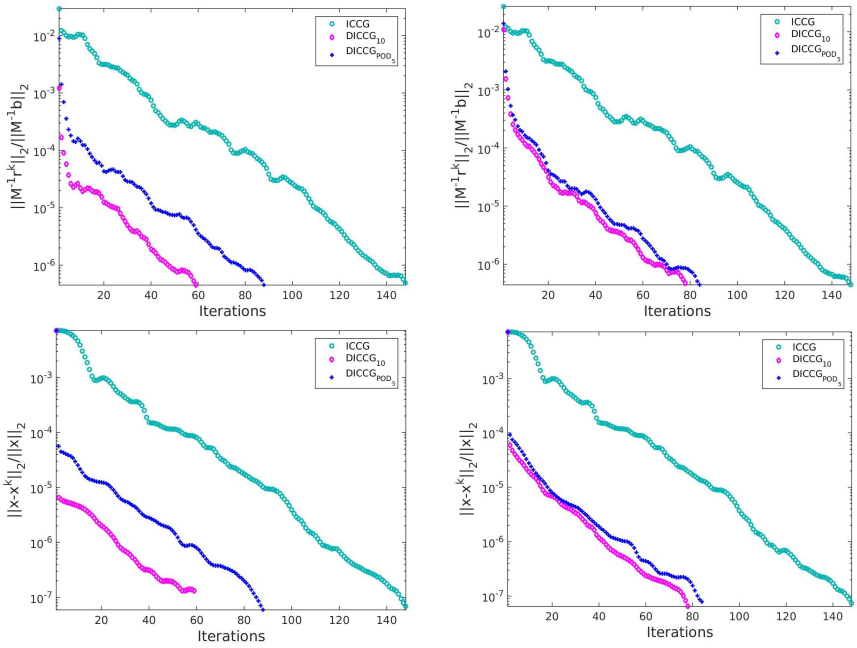


Figure 7.10: *Relative residual (upper) and true relative error (lower), left: no capillary pressure, right: capillary pressure included, 2D case.*

Injection through wells

In this section, we perform a series of experiments injecting water through wells with a prescribed bottom hole pressure (bhp). We solve the first layer of the SPE 10 benchmark and the full model containing 85 layers. We compare the DICCG and the ICCG methods, selecting a stopping criterion of $\epsilon = 10^{-7}$.

For the 2D case we select the ten and five largest POD basis vectors as deflation vectors, and for the 3D case, we use 20 and 15 vectors. In the first set of experiments we do not include capillary pressure terms, but for the 3D case, gravity terms are included.

The simulation is run during 500 time steps, with a step size of 25 days for the 2D case. For the 3D case, the simulation is run during 200 time steps with a step size of 1.5 days. We study the moving window (MW), and the training phase approaches (TP) introduced before. The bhp of the production wells for the TP cases is presented in Figure 7.11 for the first time steps. The pressure of the injection well is $\mathrm{P}_I = 1100$ bar for all cases.

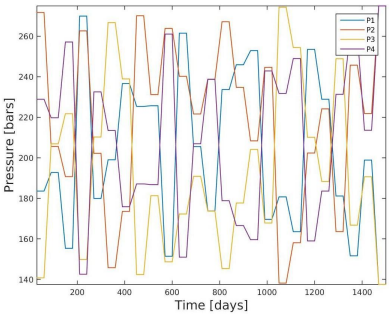


Figure 7.11: *Production well pressures, TP1.*

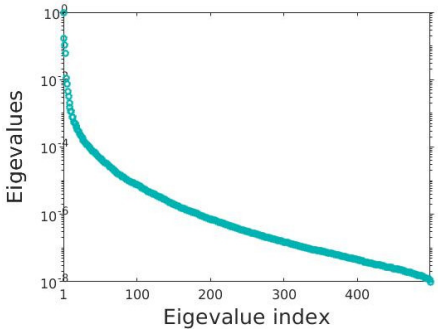


Figure 7.12: *Normalized eigenvalues of* $\mathbf{R}$*, 3D case TP1.*

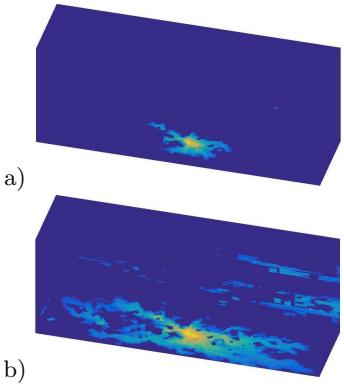


Figure 7.13: *Water saturation of the full SPE 10 benchmark, a) 50th time step, b) last time step.*

Table 7.7 shows the number of iterations required to perform the full simulation with the ICCG method (3rd column) and the DICCG method (4th column) with 5 or 10 POD basis vectors as deflation vectors p (2nd column). The percentage of the ICCG iterations required with the DICCG method is presented in the last column.

The pressure and saturation fields for the last time step and the saturation for the 200-th time step are shown in Figure 7.14 for the 2D case using the TP1 approach. The water saturation of the 3D case is presented in Figure 7.13. The eigenvalues of the training simulation are presented in Figure 7.12 for the same case.

We note that we require from 11-37 % of the ICCG iterations when using the DICCG method for the 2D case, and from 26-31% for the 3D case (see Table 7.7). Furthermore, if we use more deflation vectors, this percentage is further reduced.

	p	ICCG	DICCG	% of ICCG
MW	$P_{1:4} = 275$ bar			
	10	85477	9786	11
	5		10766	13
TP1.1	$P_{bhp} = 275$ bar			
2D	10	85477	13141	15
	5		20000	23
3D	20	96468	25376	26
	15		29658	31
	$P_{bhp} = 200$ bar			
2D	10	87204	13726	16
	5		21426	25
3D	20	96468	26730	28
	15		31146	32
	$P_{bhp} = 400$ bar			
2D	10	80591	11451	14
	5		17741	22
3D	20	96468	26730	28
	15		27429	28
TP1.2	$P_{2,3,4} = 275$ bar, $P_1 = 20$ bar			
	10	85477	12966	15
	5	85477	20034	23
	$P_{1,3,4} = 275$ bar, $P_2 = 20$ bar			
	10	90130	10518	12
	5		20926	23
	$P_{1,2,4} = 275$ bar, $P_3 = 20$ bar			
	10	90130	10518	12
	5		17325	19
	$P_{1,2,3} = 275$ bar, $P_4 = 20$ bar			
	10	90130	11636	13
	5		18693	21
	$P_{1:4} = 275$ bar			
TP2	10	85477	16202	19
	5		20585	24
TP3	10	85477	26334	31
	5		31833	37

Table 7.7: *Number of iterations, various methods and test cases.*

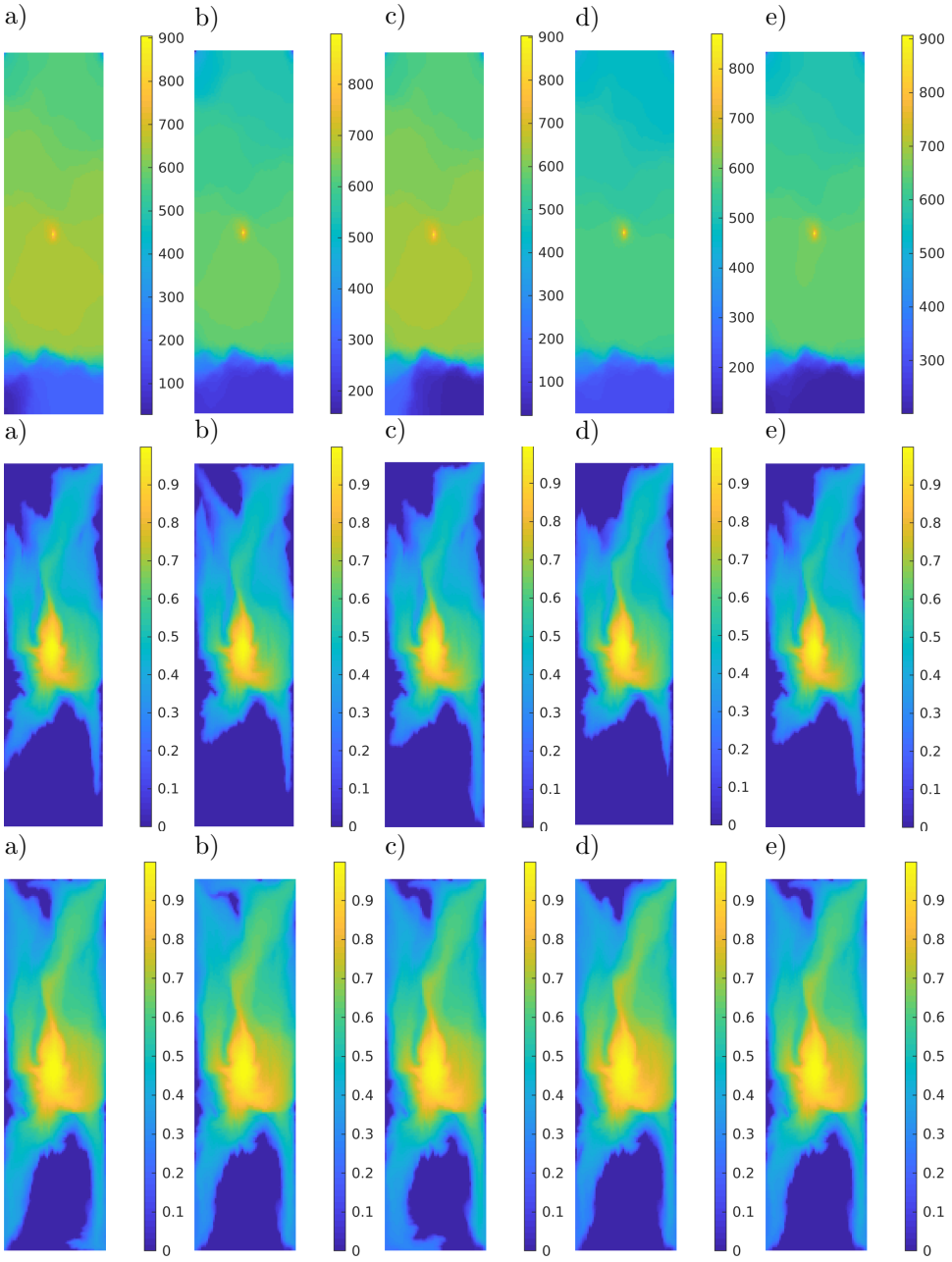


Figure 7.14: *First layer of the SPE 10 benchmark. Upper: Pressure field for the last time step, center: water saturation during the the 200-th time step, lower: water saturation during last time step. a) TP1.2: $P_1 = 20$ bar, b) TP1.2: $P_2 = 20$ bar, c) TP1.2: $P_3 = 20$ bar, d) TP1.2: $P_4 = 20$ bar, e) TP1.1: $P_{1:4} = 200$ bar.*

From the results of TP1, we can conclude that from a training phase computed in a range of pressures we can solve problems even outside this range. Furthermore, from TP1.2, we note that we can solve problems with diverse flow patterns, as showed in Figure 7.14.

For the eigenvalues of the correlation matrix (Figure 7.12), we observe a widely spread spectrum, which indicates the need for more deflation vectors to achieve a better performance for the 3D case.

We compute the amount of work of a method as the number of iterations multiplied by the work per iteration. The percentage of ICCG work required with the DICCG method, and the work required to obtain the POD basis for all methods are presented in Figure 7.16 and Figure 7.15.

The work required to compute the POD basis for the MW approach is the work required to compute the POD basis (see Appendix 2) with 10 vectors times the number of time steps (the basis is computed every time step). For the TP approach, the basis if obtainde from the whole simulation, i.e. 500 steps, the computation of the solutions of the simulation is not taken into account for this plot.

We note that almost all methods require less than 50% of the ICCG work, and using five or ten deflation vectors give a similar result. The TP3 require around 70% of the ICCG work and decreases when the number of deflation vectors decrease. We also observe that the moving window approach requires less work than the rest of the methods.

The initial work for the MW and the TP3 approaches is smaller than most of the other methods; however, the basis is computed online, which implies that this work is performed for every simulation. By contrast, in the training phase approaches, the pre-simulation is run only once, and the obtained basis can be used to solve many different problems.

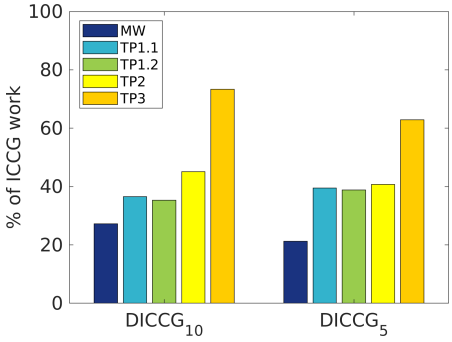


Figure 7.15: *% of ICCG work, various methods.*

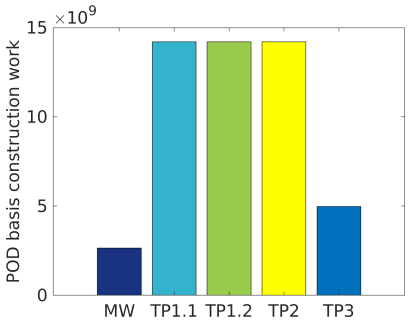


Figure 7.16: *Initial work, various methods.*

Regarding the training-phase approaches, TP2 and TP3 show an alternative way of collecting snapshots by activating only one well at the time, as presented by [11]. From these approaches, the TP3 case requires the less amount of work for the pre-simulation, but the reduction of ICCG iterations is smaller.

The preconditioned relative residual $M^{-1}r_k$ and the true relative error e_k are presented in Figure 7.17 for the MW approach and in Figure 7.18 for the TP1 approach for the 250-th time step. We note that, after the first DICCG iteration, the accuracy

of the approximation is around 10^{-4} in both cases.

We also note a faster convergence rate for DICCG methods when compared with ICCG during the first iterations. With the TP approach the behavior of the DICCG method is slightly (Figure 7.17) we used 10 snapshots and 5 POD basis vectors as deflation vectors. and only after a large number of iterations ICCG reaches the super linear convergence phase

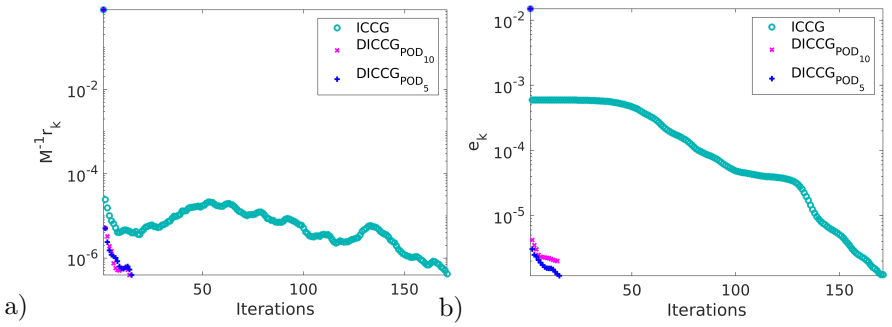


Figure 7.17: *a) Relative residual and b) true relative error, MW approach.*

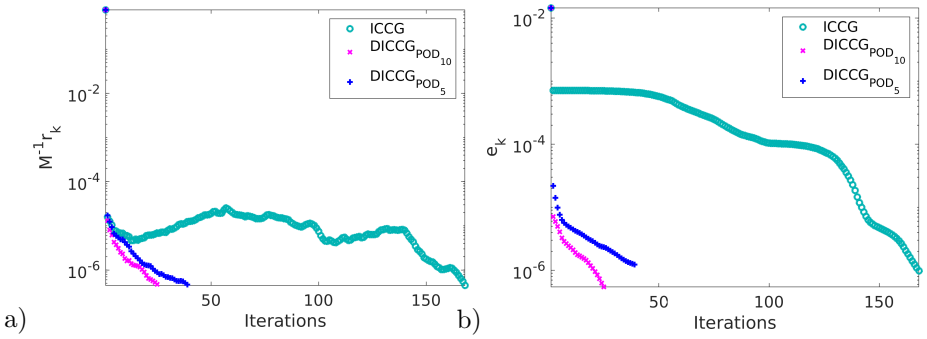


Figure 7.18: *a) Relative residual and b) true relative error, TP1 approach.*

In this section, we tested the POD-based methodology for the simulation of two-phase flow through a highly heterogeneous porous media. We achieve reductions up to 12% of the number of iterations and up to 20% of the work per iteration.

We studied various test cases that resulted in a small increment on the number of iterations if capillary pressure and gravity were included. In the next section, we perform a series of experiments to gain more insight into the influence of gravity and capillary pressure terms on the performance of the method.

7.2 Gravity-driven flow.

For this set of experiments, we simulate gravity driven two-phase flow in a reservoir containing water in the top part and oil in the bottom.

Model. We model a reservoir with a constant porosity field $\phi = 1$, i.e., an empty reservoir, and a permeability of 0.1 [D]; this example is taken from MRST [7]. The reservoir contains 20 x 20 x 40 cells, 2 m long in the x- and y-directions and 1, 2 and 4 meters long in the z-direction. The simulation is run for 800 steps, with a step size of 75 days. The implemented linear solvers are the same as in Section 7.1. For the DICCG method, we use the moving window (MW) approach. The stopping criterion is $\epsilon = 5 \cdot 10^{-7}$.

Results. The water saturation is presented in Figure 7.19 for the initial, an intermediate and the last time steps. We observe that the water is at the top of the reservoir at the beginning of the simulation, whereas at the end of the simulation, it has completely gone to the bottom.

The eigenvalues of the covariance matrix are presented in Figure 7.20 for all cases during the 200-th time step. From this plot, we can observe that they are similar; however, as the reservoir height increases, the eigenvalues become slightly smaller.

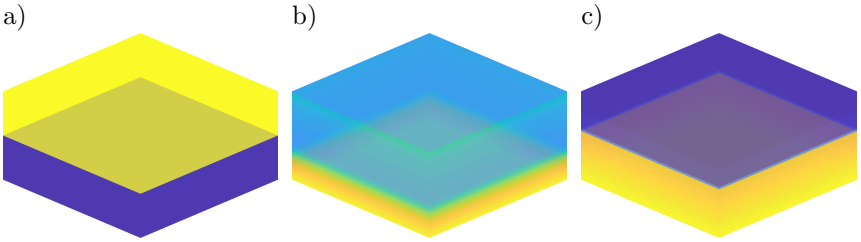


Figure 7.19: *Water saturation during a) the 1st time step, b) the 400-th time step, and c) the 800-th time step.*

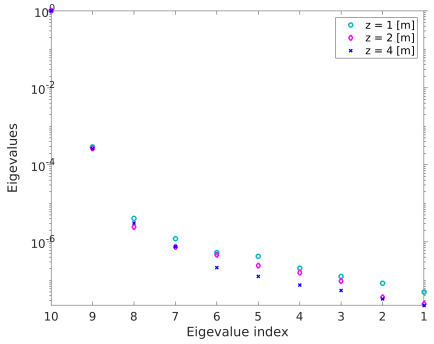


Figure 7.20: *Normalized eigenvalues of* $\mathbf{R}$*, various reservoir heights.*

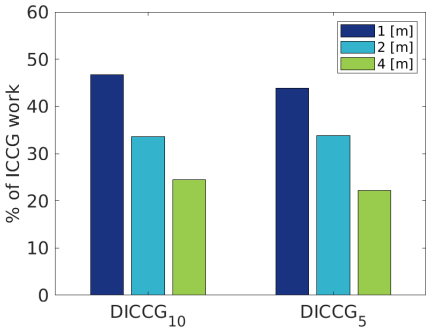


Figure 7.21: *% of ICCG work, various reservoir heights.*

The number of iterations required to achieve convergence is presented in Table 7.8. Implementing the DICCG method, we observe an important reduction in the number of iterations when compared with the ICCG method. For the ICCG method, the number of iterations varies one order of magnitude more $\mathcal{O}(10^4)$ than the variation observed for the DICCG method $\mathcal{O}(10^3)$ when changing the reservoir size.

For the cells with the smallest reservoir height, 1 meter, we require more than

the other cases. However, the number of iterations is reduced to 20% and 26% of the ICCG iterations for ten and five deflation vectors. The most significant gain is obtained for a cell size of 4 [m], where we reduce the number of iterations to $\sim 12\%$.

p	Total ICCG	DICCG ICCG	DICCG DICCG	Total DICCG	% of ICCG Iter	% of ICCG Work
Size of the z cells: 1 [m]						
10	25659	529	4506	5035	20	47
5	25659	529	6127	6656	26	44
Size of the z cells: 2 [m]						
10	36835	530	4665	5195	14	34
5	36835	530	6843	7373	20	34
Size of the z cells: 4 [m]						
10	45043	664	3974	4638	10	24
5	45043	664	5257	5921	13	22

Table 7.8: *Number of iterations for the DICCG method,* $\epsilon = 5 \cdot 10^{-7}$.

The percentage of ICCG work is also presented in the last column of Table 7.8 and in Figure 7.21. We observe that the work decreases when increasing the reservoir height, the reduction is almost half of the work when the reservoir increases four times the size. Therefore, the performance improves for higher reservoirs which have more influence from the gravity terms.

7.3 Influence of capillary pressure terms.

In this section, we perform a set of experiments varying the capillary forces to investigate their influence on the performance of the method.

Model. We model waterflooding for the first layer of the SPE 10 benchmark, with the same well configuration and reservoir properties as the previous examples containing wells. We compare a case without capillary pressure with three cases presenting diverse Corey coefficients for the wetting phase, n_w =[2,3,4], and $n_{nw} = 2$, see Table 7.1. The relative permeability curves are presented in Figure 7.1 for $n_w = n_{nw} = 2$, and Figure 7.22 for the other cases.

We use the same linear solvers as before, and we implement the training phase scheme TP1. We simulate 600 time steps, with a step size of 10 days. The pressure field and water saturation are presented in Figure 7.23 for all cases during the last time step.

Table 7.9 shows the number of iterations required to achieve convergence for a stopping criterion of $\epsilon = 5 \cdot 10^{-7}$. For the cases with capillary pressure we note similar results and a small increment in the number of iterations when we increase the Corey coefficients to $n_w = 4$, $n_{nw} = 2$. However, in most of the cases we require $\sim$6% of the ICCG iterations when using 10 deflation vectors and $\sim$27% with 5.

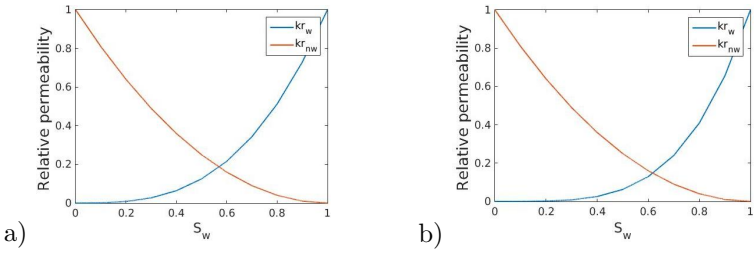


Figure 7.22: *Relative permeability curves* $n_{nw} = 2$, *a)* $n_w = 3$, *b)* $n_w = 4$.

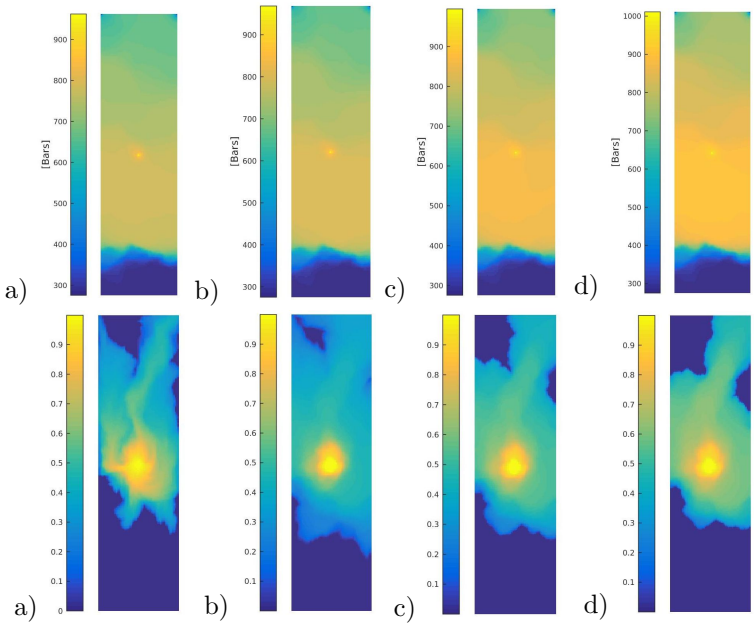


Figure 7.23: *Pressure field and water saturation for the last time step, first layer of the SPE 10 benchmark, a) No capillary pressure, b)*$n_w = n_{nw} = 2$, *c)* $n_w = 3$, $n_{nw} = 2$, *d)* $n_w = 4$, $n_{nw} = 2$.

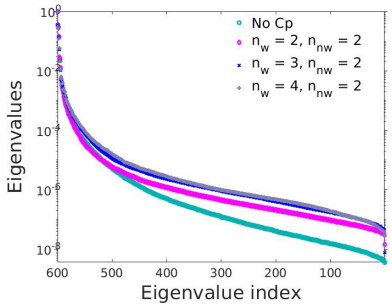


Figure 7.24: *Eigenvalues of the covariance matrix, diverse Corey coefficients*

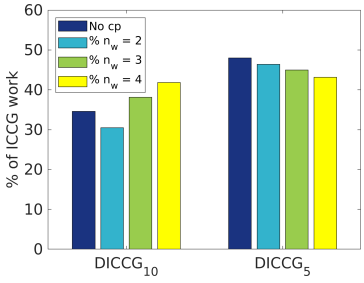


Figure 7.25: *% of ICCG work, various cases.*

The eigenvalues of the covariance matrix are presented in Figure 7.24, we note that they are similar; however, they are smaller for the case without capillary pressure and for the smaller Corey coefficient. We also observe a few eigenvalues larger than 10^{-4}, which suggests that most of the information is contained in the corresponding eigenvectors; thus, if we use ten eigenvectors as deflation vectors, they contain enough information to achieve an important acceleration.

p	Total ICCG	Total DICCG	% of ICCG Iter	% of ICCG Work
No capillary pressure				
10	86647	12603	15	34
5		24605	28	47
$k_{rw} = (S_w)^2$, $k_{rnw} = (S_{nw})^2$				
10	87235	11169	13	29
5		23965	27	45
$k_{rw} = (S_w)^3$, $k_{rnw} = (S_{nw})^2$				
10	80835	12957	16	37
5		21503	27	44
$k_{rw} = (S_w)^4$, $k_{rnw} = (S_{nw})^2$				
10	84063	14722	18	40
5		21457	26	42

Table 7.9: *Number of iterations for diverse Corey coefficients, tolerance of* $\epsilon = 5 \cdot 10^{-7}$.

The percentage of ICCG work required with the DICCG method to converge is presented in Figure 7.25 and the last column of Table 7.9. We observe a small increment when we increase the water Corey coefficient. However, the results are similar in all cases. The best performance is achieved when using 10 deflation vectors, for which we achieve reductions up to 30% the ICCG work.

In Figure 7.26, we plot the residual and the true error for the case with capillary pressure and Corey coefficients $n_w = 3$, and $n_{nw} = 2$ during the time step 150.

For the first ICCG iteration, we note that even if the residual is smaller than 10^{-4}, the true error is slightly larger, and it takes more than 100 iterations to go lower. Hence, if we would like to compute an approximation with this accuracy, the ICCG method could give an incorrect approximation. Furthermore, the method reaches the superlinear convergence region after around 120 iterations, but it does not reach the required accuracy.

By contrast, the approximation obtained with the DICCG method has a true error smaller than 10^{-5} after the first iteration. Additionally, the residual presents a similar behavior as the true error. Hence, the DICCG method for problems involving capillary pressure is also more accurate and robust than the ICCG method.

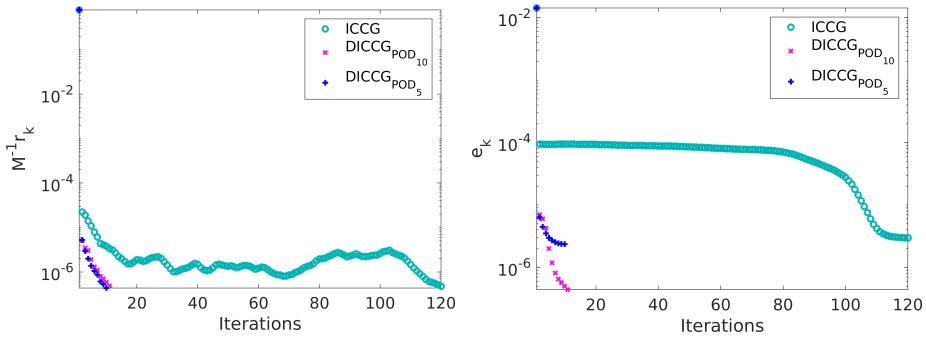


Figure 7.26: *Relative residual and true relative error for the ICCG and DICCG methods for a Corey coefficient of* $n_w = 3$.

Concluding remarks. In this chapter, we studied the performance of the POD based deflation methodology for two-phase flow problems. We studied injection of water through boundaries and wells for an 'academic' layered problem and the SPE 10 benchmark.

Results showed a correlation between the performance of the DICCG method and the eigenvalues of the correlation matrix. For the less favorable cases, the spectrum of the snapshots correlation matrix was more widely spread. Therefore, the information is distributed over more eigenvectors, and more deflation vectors are required to achieve good performance. On the contrary, if only a few eigenvalues are noticeably larger than the rest, they contain most of the system's information. Using them as deflation vectors leads to a better DICCG performance.

The first iteration of the deflated method resulted in a true solution of $\mathcal{O}(10^{-4})$ for most of the cases, which implies that if this accuracy is required, only one iteration is sufficient to have a good approximation. Furthermore, we noted that when the preconditioned residual reached the required accuracy for the ICCG method, the true error did not reach it, for some cases. By contrast, for the DICCG method, the true error and the relative residual were always of the same order of magnitude.

The DICCG method showed better performance for the higher contrast between permeability layers (c_σ). Without capillary pressure (cp) terms, for $c_\sigma = 1$, the ICCG work was reduced to 24-29%, and including cp to 29-41%. While for $c_\sigma = 6$, the reduction was to 19-29% without cp terms, and to 18-30% including cp terms. The largest reductions were achieved when using five deflation vectors. Finally, results showed that including capillary pressure terms resulted in more work of the DICCG method.

The POD basis was obtained with a moving window (MW) approach, which required an update at every time step, and a training phase (TP) approach where the basis was obtained in a pre-simulation. The work required during the iteration process was reduced to $\sim 27\%$ with ten deflation vectors and $\sim 21\%$ with 5 for the MW approach and from 35% to 73% (10 deflation vectors) and 38% to 63% (5 deflation vectors) for the TP approach. Thus, performing the MW window approach is less expensive; however the computation of the POD basis is performed for every simulation using this approach. By contrast, the basis computed with the TP approach can be to solve different problems, with different bhp in the wells.

Chapter 8

Comparison of 2L-PCG using deflation techniques.

In Chapter 3, we introduced the preconditioning techniques that help us to cluster the spectrum of the system matrix, accelerating in this way an iterative solver. For the implementation of these techniques, the original system is multiplied by a traditional preconditioner $\mathbf{M}^{-1}$, improving in this way the conditioning of this matrix.

Examples of traditional preconditioners are the inverse of the matrices $\mathbf{M}$ used for basic iterative methods, presented in Table 3.1, and the Incomplete Cholesky factorization with zero fill in $\mathbf{M} = \mathbf{IC}_0$, used throughout this thesis.

The conditioning of the matrix can be further improved by incorporating a second kind of preconditioner. The combination of a first- (traditional) and a second-level preconditioner is known as two-level preconditioning. Examples of the second-level preconditioners are Multigrid (MG) and domain decomposition (DDM) methods.

A two-level preconditioner applied to the PCG method gives rise to the two-level PCG method (2L-PCG). By making use of this method, the influence of the large eigenvalues of the system matrix is reduced in a first level, and the remaining small eigenvalues are treated in a second level.

The most basic 2L-PCG method is constructed by combining the PCG and a two-grid method. Here, an approximate solution is obtained on a fine grid making use of a first-level preconditioner. However, after this procedure, many of the low-frequency components of the error are not effectively treated. Then, coarser meshes are build to eliminate the influence of these components [16].

Multi-level methods have a geometrical relationship with the fine grid; if avoiding this relation is beneficial, a more general second-level problem is built by only using the system matrix $\mathbf{A}$. This method is called algebraic multigrid (AMG) and can be used for unstructured grids.

In the previous sections, we studied the deflation method, that can also be interpreted as a 2L-PCG method, where the second level preconditioner is the deflation subspace matrix $\mathbf{P}$. The performance of this method highly depends on the selection of the projection vectors, where, the eigenvectors corresponding to the unfavorable eigenvalues of the system matrix are a good choice. Contrary to MG methods, where

the projection vectors represent an interpolation between the fine and coarse grids.

However, the deflation method is similar to the multigrid method without pre- and post-smoothing, i.e., the deflation operator is the same as the coarse-grid operator of MG. The main difference between these methods is that the deflation method is effective for Krylov-subspace methods, contrary to the use of a coarse grid correction without smoothing, that does not lead to a successful MG method.

In this thesis, we introduced the use of a POD basis to construct a deflation-subspace matrix for the acceleration of a Krylov subspace method using standard deflation procedure [31–33]. Recently, Pasetto et al. [1] explore the POD-based deflation-subspace matrix for the construction of a two-level preconditioner.

In this Chapter, we make a comparison of different two-level preconditioners, including the deflation approach used throughout this work, referred to as DEF1, and the two-level preconditioners introduced by Pasetto et al. [1], referred to as ROM, and SROM. We study this method from a theoretical point of view and with a series of computational experiments.

8.1 Two-Level Preconditioned Conjugate Gradient (2L-PCG)

The two-level preconditioning techniques consist of an arbitrary first-level traditional preconditioner $\mathbf{M}^{-1}$, combined with one or more second-level projection matrices. Therefore, MG, DDM, and deflation methods can be seen as 2L-PCG methods. In this section, we introduce the two-level PCG method together with some properties. Furthermore, we introduce the methods compared in this chapter from a two-level method perspective.

The 2L-PCG method is defined as

$$\mathcal{P}\mathcal{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}, \quad \mathcal{P}, \mathcal{A} \in \mathbb{R}^{n\times n} \tag{8.1}$$

where, the matrix $\mathcal{A}$ is the system matrix $\mathbf{A}$, that can also be a combination of $\mathbf{A}$ and the deflation matrix $\mathbf{P}$ (see Definition 3.4.1). The operator $\mathcal{P}$ is a two-level preconditioner, wich can consist of a traditional preconditioner $\mathbf{M}^{-1}$ or a combination of traditional preconditioners and/or projectors.

There are multiple ways to construct the second-level operator $\mathcal{P}$. A special case is using the identity matrix $\mathcal{P} = \mathbf{I}$, thus $\mathcal{A} = \mathbf{A}$, $\underline{\mathbf{x}} = \mathbf{x}$ and $\underline{\mathbf{b}} = \mathbf{b}$, which results in the standard CG method. When the operator is a traditional preconditioner, $\mathcal{P} = \mathbf{M}^{-1}$, and $\underline{\mathbf{b}} = \mathbf{M}^{-1}\mathbf{b}$, Equation 8.1 reduces to the PCG method.

This chapter is based on:

J. Tjan. Study on deflation techniques and POD methods for the acceleration of Krylov subspace methods. Master's thesis, TU Delft, 8 2018. `web:http://ta.twi.tudelft.nl/nw/users/vuik/numanal/tjan_scriptie.pdf`.

Besides of traditional preconditioners, the operator $\mathcal{P}$ can be constructed using single-level and two-level preconditioners in an additive or in a multiplicative way [53], this methodology is explained below.

8.1.1 Additive preconditioner

Let $\mathbf{C}_1, \mathbf{C}_2$ be two arbitrary symmetric positive semi-definite (SPSD) preconditioners, and $c_i > 0 \in \mathbb{R}$, then the additive combination

$$\mathcal{P}_{a_2} = c_1\mathbf{C}_1 + c_2\mathbf{C}_2 \tag{8.2}$$

is also an SPSD preconditioner. Therefore, a linear combination of different SPSD preconditioners $\mathbf{C}_i$ with different weights c_i is also a preconditioner. We can generalize this result as:

$$\mathcal{P}_{a_k} = \sum_{i=1}^{k} c_i\mathbf{C}_i. \tag{8.3}$$

8.1.2 Multiplicative preconditioner

Let $\mathbf{C}_1, \mathbf{C}_2$ be two arbitrary SPSD preconditioners. We consider the combined iterations induced by these preconditioners as follows:

$$\begin{aligned} \mathbf{x}^{i+\frac{1}{2}} &= \mathbf{x}^i + \mathbf{C}_1(\mathbf{b} - \mathbf{A}\mathbf{x}^i), \\ \mathbf{x}^{i+1} &= \mathbf{x}^{i+\frac{1}{2}} + \mathbf{C}_2(\mathbf{b} - \mathbf{A}\mathbf{x}^{i+\frac{1}{2}}), \end{aligned} \tag{8.4}$$

then, we can combine the two preconditioners in the following way:

$$\mathbf{x}^{i+1} = \mathbf{x}^i + (\mathbf{C}_1 + \mathbf{C}_2 - \mathbf{C}_2\mathbf{A}\mathbf{C}_1)(\mathbf{b} - \mathbf{A}\mathbf{x}^i),$$

which can be interpreted as a multiplicative operator $\mathcal{P}_{m_2}$ consisting on two preconditioners [45, 53],

$$\mathcal{P}_{m_2} = \mathbf{C}_1 + \mathbf{C}_2 - \mathbf{C}_2\mathbf{A}\mathbf{C}_1. \tag{8.5}$$

This method can also be generalized to $\mathcal{P}_{m_k}$ by using k preconditioners. In particular, using three preconditioners, $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$, results in

$$\mathcal{P}_{m_3} = \mathbf{C}_1 + \mathbf{C}_2 + \mathbf{C}_3 - \mathbf{C}_2\mathbf{A}\mathbf{C}_1 - \mathbf{C}_3\mathbf{A}\mathbf{C}_2 - \mathbf{C}_3\mathbf{A}\mathbf{C}_1 + \mathbf{C}_3\mathbf{A}\mathbf{C}_2\mathbf{A}\mathbf{C}_1. \tag{8.6}$$

In the next sections, we introduce the methods studied throughout this chapter, seeing them as two-level additive or multiplicative preconditioners.

8.1.3 Deflation method

The Deflated Preconditioned Conjugated Gradient (DPCG), defined in Section 3.4.4, can be written as:

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

where $\hat{\mathbf{x}}$ is a non unique solution to this system, and the complete solution to the original system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is obtained by using

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

This method can be seen as a 2L-PCG method, where

$$\mathcal{P}_{DEF1} = \mathbf{M}^{-1}\mathbf{P}. \tag{8.7}$$

This formulation is referred to as "Deflation Variant 1" or in short DEF1.

An alternative way to describe the deflation technique was proposed by [45, 53] as follows:

Definition 8.1.1. Let $\overline{\mathbf{x}}$ be an arbitrary initial guess vector, we define a special starting vector ($\mathcal{V}_{start}$) as

$$\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}. \tag{8.8}$$

It can be shown (see [45]), that the solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is:

$$\mathbf{x} = \mathbf{x}^0 + \mathbf{P}^\top\mathbf{y}, \tag{8.9}$$

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

$$\mathbf{A}\mathbf{P}^\top\mathbf{y} = \mathbf{r}_0, \quad \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0. \tag{8.10}$$

The latter expression can be solved using a single-level preconditioner $\mathbf{M}^{-1}$, leading to

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{P}^\top\mathbf{y} = \mathbf{M}^{-1}\mathbf{r}_0. \tag{8.11}$$

Multiplying the previous equation by $\mathbf{P}^\top$, and using Equation (8.9) results in

$$\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^\top\mathbf{M}^{-1}\mathbf{b}. \tag{8.12}$$

Therefore, the projection operator of the resulting 2L-PCG algorithm is given by

$$\mathcal{P}_{DEF2} = \mathbf{P}^T\mathbf{M}^{-1}, \tag{8.13}$$

and it is referred to as "Deflation Variant 2" (DEF2).

The difference between DEF1 and DEF2 is the flipped two-level preconditioner. Furthermore, when using the DEF1 method, the unique solution $\mathbf{x}$ is found by projecting back the deflated solution via the operation $\mathbf{x} = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$, executed after the iterations. Contrary to the DEF2 method, for which this operation is executed before. Therefore, the methods have different robustness properties. More details can be found in [45, 53].

8.1.4 Adapted deflation methods

Selecting $\mathbf{C}_1 = \mathbf{Q}$ and $\mathbf{C}_2 = \mathbf{M}^{-1}$ to construct an additive preconditioner (Equation (8.5)), where $\mathbf{M}$ is a traditional preconditioner, the resulting two-level preconditioner is given by:

$$\mathcal{P}_{\text{A-DEF1}} = \mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}. \tag{8.14}$$

This method is referred to as Adapted Deflation Variant 1 (A-DEF1).
Similarly, an Adapted Deflation Variant 2 (A-DEF2) can be constructed by using $\mathbf{C}_1 = \mathbf{M}^{-1}$ and $\mathbf{C}_2 = \mathbf{Q}$ in Equation (8.5) to obtain a second-lever preconditioner:

$$\mathcal{P}_{\text{A-DEF2}} = \mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{Q}. \tag{8.15}$$

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

8.1.5 ROM-based preconditioner

The operator $\mathcal{P}_{\text{ROM}}$, proposed by Pasetto et al. [1], is an approximation of the inverse of $\mathbf{A}$ based on algebraic multigrid methods (AMG). For this method, the preconditioner is given by:

$$\mathcal{P}_{\text{ROM}} = \mathbf{M}^{-1} + \mathbf{Q}(1 - \mathbf{A}\mathbf{M}^{-1}), \tag{8.16}$$

using $\mathbf{Q} = \mathbf{Z}^\top \mathbf{E}^{-1}\mathbf{Z}$ and $\mathbf{E} = \mathbf{Z}^\top \mathbf{A}\mathbf{Z}$ from Definition 3.4.1.
Note that $\mathcal{P}_{\text{ROM}}$ is not always symmetric. However, an SPD variant is constructed as follows:

$$\mathcal{P}_{\text{SROM}} = \frac{\mathcal{P}_{\text{ROM}} + \mathcal{P}_{\text{ROM}}^\top}{2} = \mathbf{M}^{-1} + \mathbf{Q} - \frac{1}{2}\big(\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q}\big). \tag{8.17}$$

Later, in Lemma 8.2.1 of Section 8.2.1 we prove that the ROM method has the same operator as the A-DEF2 method. Furthermore, we show that $\mathcal{P}_{\text{SROM}}$ can be seen as an additive preconditioner consisting of the operator $\mathcal{P}_{\text{A-DEF1}}$ and $\mathcal{P}_{\text{A-DEF2}}$; thus, it can be written as:

$$\mathcal{P}_{\text{SROM}} = \frac{1}{2}(\mathcal{P}_{\text{A-DEF1}} + \mathcal{P}_{\text{A-DEF2}}). \tag{8.18}$$

8.1.6 Abstract balancing methods

Sometimes, symmetric operators are required. From the operators presented above, we mentioned that $\mathcal{P}_{\text{A-DEF1}}$, $\mathcal{P}_{\text{A-DEF2}}$ are not always symmetric, and we introduced a symmetric version of the ROM operator. In this section, we present some balancing methods, that are symmetric operators constructed by using the multiplicative preconditioner methodology with three preconditioners.

Combining $\mathbf{C}_1 = \mathbf{Q}$, $\mathbf{C}_2 = \mathbf{M}^{-1}$ and $\mathbf{C}_3 = \mathbf{Q}$ in a multiplicative way (see Equation 8.6), we obtain

$$\mathcal{P}_{\text{BNN}} = \mathbf{P}^\top \mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}. \tag{8.19}$$

The operator $\mathcal{P}_{\text{BNN}}$ is known as the Balancing-Neumann-Neumann (BNN) operator and it is an operator typically used for Domain Decomposition Methods (DDM). This operator is an SPD preconditioner. A reduced version of the BNN operator is obtained by removing the correction matrix $\mathbf{Q}$, leading to:

$$\mathcal{P}_{\text{R-BNN1}} = \mathbf{P}^\top \mathbf{M}^{-1}\mathbf{P}, \tag{8.20}$$

which is still a symmetric preconditioner. We can further reduce the preconditioner by removing the matrix $\mathbf{P}$, obtaining

$$\mathcal{P}_{\text{R-BNN2}} = \mathbf{P}^\top \mathbf{M}^{-1}. \tag{8.21}$$

Which similar to DEF2, and the only difference lies on the implementation of the method. Both $\mathcal{P}_{\text{R-BNN1}}$ and $\mathcal{P}_{\text{R-BNN2}}$ have the same properties as $\mathcal{P}_{\text{BNN}}$ if a correct initial condition is used. More details can be found in [45, 53].

Summary of the methods. In Equation 8.1, we presented the two-level Preconditioned Conjugated Gradient method (2L-PCG), given by:

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

with $\mathcal{A} = \mathbf{A}$, $\mathcal{P}$ an operator, $\underline{\mathbf{x}} = \mathbf{x}$ and $\underline{\mathbf{b}} = \mathcal{P}\mathbf{b}$, the previously mentioned methods can be written in terms of the 2L-PCG method. Some of them need a special starting vector $\mathcal{V}_{\text{start}}$ for a correct performance [45, 53]. From now on, we will consider $\mathbf{x}^0$ as the starting vector, but note that, it depends on the initial guess ($\overline{\mathbf{x}}$), and it is consider a special starting vector if it is given by $\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$.

A summary of these methods together with their required starting vectors is presented in Table 8.1. Note that, the original starting vector proposed by Pasetto et al. [1] for the ROM and SROM methods was an initial guess solution given by $\mathbf{x}^0_{(ROM/SROM)} = \overline{\mathbf{x}}$. However, we show in this work (see section 8.3) that the correct starting vector is the same as the starting vector of the A-DEF2 method, $\mathbf{x}^0_{(ROM/SROM)} = \mathbf{x}^0_{(A-DEF2)} = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$, as presented in Table 8.1.

Algorithm 10 shows the implementation of the 2L-PCG methods, where the corresponding matrices, and the required starting $\mathcal{V}_{\text{start}} = \mathbf{x}^0$ and ending $\mathcal{V}_{\text{end}}$ vectors are given in Table 8.2. This implementation is used for the numerical experiments presented in Section 8.3. Next, we present a theoretical comparison of some of the two-level preconditioners introduced in this section.

Name	Method	$\mathcal{V}_{\text{start}} = \mathbf{x}^0$	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	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q}(1 - \mathbf{A}\mathbf{M}^{-1})$
SROM	SROM-based preconditioner	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q} - \frac{(\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q})}{2}$

Table 8.1: *Overview of the various 2L-PCG methods*

Algorithm 10 Generalized two-level preconditioner Method

Required: a starting vector ($\mathcal{V}_{\text{start}} = \mathbf{x}^0$) depending on an initial guess ($\overline{\mathbf{x}}$), preconditioners: $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$, and a final vector: $\mathcal{V}_{\text{end}}$
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 $k = 0, \ldots$
  while $\mathbf{r}^k > \varepsilon$
    $\mathbf{w}^k = \mathcal{M}_3\mathbf{A}\mathbf{p}$
    $\alpha^k = \frac{\langle \mathbf{r}^k, \mathbf{z}^k \rangle}{\langle \mathbf{p}^k, \mathbf{w}^k \rangle}$
    $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k$
    $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha^k \mathbf{A}\mathbf{p}^k$
    $\mathbf{z}^{k+1} = \mathcal{M}_1\mathbf{r}^{k+1}$
    $\beta^k = \frac{\langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{z}^k, \mathbf{r}^k \rangle}$
    $\mathbf{p}^{k+1} = \mathcal{M}_2\mathbf{z}^{k+1} + \beta^k \mathbf{p}^k$
  end while
end for
$\mathcal{V}_{\text{end}}$ %Computed as in Table 8.2 from the approximate solution $\mathbf{x}^{k+1}$

Name	$\mathcal{V}_{\text{start}} = \mathbf{x}^0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{V}_{\text{end}}$
PREC	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
DEF1	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}$	$\mathbf{I}$	$\mathbf{P}$	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top \mathbf{x}_{k+1}$
DEF2	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$	$\mathbf{M}^{-1}$	$\mathbf{P}^\top$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
A-DEF1	$\overline{\mathbf{x}}$	$\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
A-DEF2	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{Q}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
BNN	$\overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
R-BNN1	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
R-BNN2	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$	$\mathbf{P}^\top\mathbf{M}^{-1}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
ROM	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q}(1 - \mathbf{A}\mathbf{M}^{-1})$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$
SROM	$\mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$	$\mathbf{M}^{-1} + \mathbf{Q} - \frac{(\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q})}{2}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{x}_{k+1}$

Table 8.2: *Parameters of the 2L-PCG methods computed with Algorithm 10.*

8.2 Theoretical comparison of two-level preconditioners

In this section, we compare the different two-level preconditioners discussed above. For the comparison, we present the computational complexity of the methods in terms of the number of operations required for the initialization of the method, and to perform one iteration.

Furthermore, we study the memory requirements of the methods. The results are given in terms of the size of the matrix (n), the sparsity (m) and the number of deflation vectors (p) used. Next, we study the spectrum of the preconditioned systems

and we prove some equivalence lemmas.

Computational complexity. In this section we compute the number of flops per iteration and for the initialization of the methods, more details can be found in Appendix 9.2 and [48].

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

Table 8.3: *Memory storage of the 2L-PCG methods.*

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

Table 8.4: *Number of operations required to implement the 2L-PCG methods.*

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a matrix containing m non-zero diagonals, and let $\mathbf{Z} \in \mathbb{R}^{n\times p}$ be a full deflation subspace matrix containing p deflation vectors. The number of operations required to perform the studied methods is presented in Table 8.4, and the memory storage in Table 8.3.

Regarding the flops per iteration, the cheapest 2L-PCG methods are: DEF1, DEF2, ROM, and R-BNN2, all of them requiring $\mathcal{O}(4pn)$ arithmetic work per iteration, and the most expensive is the BNN method requiring $\mathcal{O}(12pn)$ flops per iteration.

All deflation methods require similar amount of memory.

8.2.1 Spectral comparison

In Chapter 3, we showed that the deflation methodology improves the condition number of the system matrix, leading to a further CG acceleration.

In this section, we analyze the behavior of the eigenvalues of the two-level preconditioners applied to the matrix $\mathbf{A}$. In particular, we study the change of the condition number when using the operator $\mathcal{P}\mathbf{A}$ for the different methods. Some basic eigenvalue properties used in this section can be found in Appendix 9.2 and [48].

Theoretical comparison of the A-DEF2 method and the ROM method

In this section, we compare the A-DEF2 and the ROM methods. Lemma 8.2.1 shows that the operators $\mathcal{P}_{\text{A-DEF2}}$ and the $\mathcal{P}_{\text{ROM}}$ are the same.

Lemma 8.2.1. The A-DEF2 and the ROM methods have the same operator.

Proof. Recall that the operators are defined as

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

$$\begin{aligned} \mathcal{P}_{\text{A-DEF2}} &= \mathbf{P}^\top \mathbf{M}^{-1} + \mathbf{Q} \\ &= (\mathbf{I} - \mathbf{Q}\mathbf{A})\mathbf{M}^{-1} + \mathbf{Q} && \text{Lemma 3.4.2 k)} \\ &= \mathbf{M}^{-1} - \mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{Q} \\ &= \mathbf{M}^{-1} + \mathbf{Q}(\mathbf{I} - \mathbf{A}\mathbf{M}^{-1}) \\ &= \mathcal{P}_{\text{ROM}}. \end{aligned}$$

□

The only difference of these methods lies in the selection of the starting vector. For the A-DEF2 method, the starting vector is given by $\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top \overline{\mathbf{x}}$, whereas previous applications of ROM [1] did not considered this initial operation. The following numerical results show that this starting operation is needed also for ROM and SROM.

Spectra analysis

In Section 4.1.1, we showed that, when using eigenvectors of the system matrix as deflation vectors, the eigenvalues corresponding to these eigenvectors are set to zero. However, this is not the case for all operators presented in this work (see Table 8.1).

In this section, we study the spectrum resulting after application of the preconditioners defined in Table 8.2. For a better understanding, we divide them into two different classes depending on their properties. The operators of Class 0 consist of the preconditioners DEF1, DEF2, R-BNN1 and R-BNN2; and the Class 1 consists of BNN, A-DEF1, A-DEF2, and ROM.

We begin with two lemmas, presenting the spectral properties of each class. Then, we make a connection between the two classes. Finally, we focus the study on the

SROM operator, investigating the resulting preconditioned spectra for a specific choice of deflation vectors.

For the results here presented, unless stated otherwise, we consider $\mathbf{P} \in \mathbb{R}^{n\times n}$, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{Z} \in \mathbb{R}^{n\times p}$ an arbitrary full rank matrix with p deflation vectors as column vectors (see Definition 3.4.1). And $\mathbf{M} \in \mathbb{R}^{n\times n}$ is an SPD traditional preconditioner.

Lemma 8.2.2. The spectrum corresponding to the preconditioned matrix $\mathcal{P}\mathbf{A}$ for the Class 0 operators $\mathcal{P}_{DEF1/2}$, $\mathcal{P}_{R-BNN1/2}$ is the same, i.e.,

$$\sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A}) = \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}) = \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}). \tag{8.22}$$

Proof. From Table 8.2, we note that DEF2 and R-BNN2 have the same operator, $\mathcal{P}_{DEF2} = \mathcal{P}_{R-BNN2}$. Hence, the spectrum of these methods is the same. Therefore, it is only required to prove that the spectrum of $\mathcal{P}_{DEF1}\mathbf{A}$, $\mathcal{P}_{DEF2}\mathbf{A}$, and $\mathcal{P}_{R-BNN1}\mathbf{A}$ are the same.

First, we show that the spectrum of $\mathcal{P}_{DEF1}\mathbf{A} = \mathbf{M}^{-1}\mathbf{P}\mathbf{A}$ and $\mathcal{P}_{DEF2}\mathbf{A} = \mathbf{P}^T\mathbf{M}^{-1}\mathbf{A}$ are the same.

$$\begin{aligned}\sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A}) &= \sigma(\mathbf{A}\mathbf{M}^{-1}\mathbf{P}) && \text{Lemma 3.1.1 a)}\\ &= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}). && \text{Lemma 3.1.1 c)}\end{aligned}$$

Now, we showed that the spectrum of $\mathcal{P}_{DEF1}\mathbf{A} = \mathbf{M}^{-1}\mathbf{P}\mathbf{A}$ and $\mathcal{P}_{R-BNN1}\mathbf{A} = \mathbf{P}^T\mathbf{M}^{-1}\mathbf{A}$ are the same.

$$\begin{aligned}\sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A}) &= \sigma(\mathbf{M}^{-1}\mathbf{P}^2\mathbf{A}) && \text{Lemma 3.4.2 e)}\\ &= \sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{P}^\top) && \text{Lemma 3.4.2 f)}\\ &= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}). && \text{Lemma 3.1.1 a)}\end{aligned}$$

□

In Lemma 8.2.2, we show that the Class 0 operators: DEF1/2 and R-BNN1/2 applied to the system matrix $\mathbf{A}$ produce the same spectrum. In Lemma 8.2.3 we prove the same for the Class 1 operators: A-DEF1, A-DEF2, ROM and BNN.

Lemma 8.2.3. The spectrum corresponding to the preconditioned matrix $\mathcal{P}\mathbf{A}$ for the Class 1 operators: $\mathcal{P}_{A-DEF1/2}$, $\mathcal{P}_{ROM}$ and $\mathcal{P}_{BNN}$ is the same, i.e.,

$$\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A} + \mathbf{Q}\mathbf{A}) = \sigma(\mathbf{M}^{-1}\mathbf{P}\mathbf{A} + \mathbf{Q}\mathbf{A}) = \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A} + \mathbf{Q}\mathbf{A}). \tag{8.23}$$

Proof. From Lemma 8.2.1, we know that the operators $\mathcal{P}_{\text{A-DEF2}}$ and the $\mathcal{P}_{\text{ROM}}$ are the same; hence, the spectrum of the matrices preconditioned with these operators is the same. Now, we prove that the application of the operators A-DEF1, A-DEF2, and BNN to the matrix $\mathbf{A}$ produce the same spectrum.

First, we show that the spectrum of the preconditioned systems $\mathcal{P}_{A-DEF1}\mathbf{A} =$

$\mathbf{M}^{-1}\mathbf{PA}+\mathbf{QA}$ and $\mathcal{P}_{A-DEF2}\mathbf{A}=\mathbf{P}^T\mathbf{M}^{-1}\mathbf{A}+\mathbf{QA}$ are the same.

$$\begin{aligned}
\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}+\mathbf{QA}) &= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}+\mathbf{I}-\mathbf{P}^\top) && \text{Lemma 3.4.2 k)}\\
&= \sigma(\mathbf{P}^\top(\mathbf{M}^{-1}\mathbf{A}-\mathbf{I}))+\sigma(\mathbf{I}) && \text{Lemma 3.1.1 b)}\\
&= \sigma((\mathbf{M}^{-1}\mathbf{A}-\mathbf{I})\mathbf{P}^\top)+\sigma(\mathbf{I}) && \text{Lemma 3.1.1 a)}\\
&= \sigma(\mathbf{M}^{-1}\mathbf{AP}^\top-\mathbf{P}^\top\mathbf{I})+\sigma(\mathbf{I}) && \text{Lemma 3.1.1 b)}\\
&= \sigma(\mathbf{M}^{-1}\mathbf{PA}-\mathbf{P}^\top+\mathbf{I}) && \text{Lemma 3.4.2 f)}\\
&= \sigma(\mathbf{M}^{-1}\mathbf{PA}+\mathbf{QA}). && \text{Lemma 3.4.2 k)}
\end{aligned}$$

Then, we prove that $\mathcal{P}_{BNN}\mathbf{A}=\mathbf{P}^T\mathbf{M}^{-1}\mathbf{PA}+\mathbf{QA}$ and $\mathcal{P}_{A-DEF1}\mathbf{A}=\mathbf{M}^{-1}\mathbf{PA}+\mathbf{QA}$ produce the same spectrum.

$$\begin{aligned}
\sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{PA}+\mathbf{QA}) &= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{PA}+\mathbf{I}-\mathbf{P}^\top) && \text{Lemma 3.4.2 k)}\\
&= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{PA}-\mathbf{P}^\top)+\sigma(\mathbf{I}) && \text{Lemma 3.1.1 b)}\\
&= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{AP}^\top-\mathbf{P}^\top)+\sigma(\mathbf{I}) && \text{Lemma 3.4.2 f)}\\
&= \sigma(\mathbf{PAM}^{-1}\mathbf{P}-\mathbf{P})+\sigma(\mathbf{I}) && \text{Lemma 3.1.1 c)}\\
&= \sigma(\mathbf{P}(\mathbf{AM}^{-1}\mathbf{P}-\mathbf{I}))+\sigma(\mathbf{I})\\
&= \sigma((\mathbf{AM}^{-1}\mathbf{P}-\mathbf{I})\mathbf{P})+\sigma(\mathbf{I}) && \text{Lemma 3.1.1 a)}\\
&= \sigma(\mathbf{AM}^{-1}\mathbf{P}^2-\mathbf{P})+\sigma(\mathbf{I})\\
&= \sigma(\mathbf{AM}^{-1}\mathbf{P}-\mathbf{P})+\sigma(\mathbf{I}) && \text{Lemma 3.4.2 e)}\\
&= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}-\mathbf{P}^\top)+\sigma(\mathbf{I}) && \text{Lemma 3.1.1 c)}\\
&= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}-\mathbf{P}^\top+\mathbf{I}) && \text{Lemma 3.1.1 b)}\\
&= \sigma(\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}-\mathbf{QA}). && \text{Lemma 3.4.2 k)}
\end{aligned}$$

□

The previous lemmas show that for each class of operators, the spectrum of the preconditioned system $\mathcal{P}\mathbf{A}$ is the same. Therefore, it suffices to study the spectral behavior of only one method of each class. In Lemma 8.2.4, we present the relation between the preconditioned spectra when using a Class 0 (DEF1) and a Class 1 (BNN) operators.

Lemma 8.2.4. Let the spectrum of the preconditioned system $\mathcal{P}\mathbf{A}$ for the operators $\mathcal{P}_{DEF1}=\mathbf{M}^{-1}\mathbf{P}$ of Class 0, and $\mathcal{P}_{BNN}=\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}+\mathbf{Q}$ of Class 1, be given by

$$\sigma_{DEF1}(\mathbf{M}^{-1}\mathbf{PA})=\{\lambda_1,\ldots,\lambda_n\},\quad \sigma_{BNN}(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{PA}+\mathbf{QA})=\{\mu_1,\ldots,\mu_n\}$$

respectively.

Then, the eigenvalues λ_i, and μ_i can be reordered such that

$$\lambda_i=0,\quad \mu_i=1,\quad i=1,\ldots,p$$

and

$$\lambda_i=\mu_i,\quad i=p+1,\ldots,n.$$

Proof. Considering the operator $\mathcal{P}_{BNN} = \mathbf{P}^T\mathbf{M}^{-1}\mathbf{P} + \mathbf{Q}$, if we apply this operator to $\mathbf{AZ}$, it follows from Lemma 3.4.2 h) and c) that

$$(\mathbf{P}^\top\mathbf{M^{-1}P} + \mathbf{Q})\mathbf{AZ} = \mathbf{P}^\top\mathbf{M^{-1}PAZ} + \mathbf{QAZ} = \mathbf{0} + \mathbf{QAZ} = \mathbf{Z}.$$

Therefore, the column vector $\mathbf{v}_i$ of $\mathbf{Z}$ is an eigenvector of $\mathcal{P}_{\text{BNN}}\mathbf{A}$, and it corresponds to the eigenvalue $\mu_i = 1$.
Similarly, for the operator $\mathcal{P}_{DEF1} = \mathbf{M}^{-1}\mathbf{P}$ we have (Lemma 3.4.2 h)

$$\mathbf{M^{-1}PAZ} = 0.$$

Then, the column vector $\mathbf{v}_i$ of $\mathbf{Z}$ is an eigenvector of $\mathcal{P}_{\text{DEF1}}\mathbf{A}$, corresponding to the eigenvalue $\lambda_i = 0$.

From [54, Th.2.8], it is sufficient to proof that if

$$\sigma_{BNN}(\mathbf{P}^\top\mathbf{M^{-1}PA} + \mathbf{QA}) = \{1, \ldots, 1, \mu_{p+1}, \ldots, \mu_n\}$$

holds, then

$$\sigma_{DEF1}(\mathbf{M^{-1}PA}) = \{0, \ldots, 0, \mu_{p+1}, \ldots, \mu_n\}.$$

Let μ_i be an eigenvalue corresponding to the eigenvector $\mathbf{v}_i$ of $\mathcal{P}_{\text{BNN}}\mathbf{A}$, where $i = p+1, \ldots n$, i.e., $\mathbf{v}_i \notin \text{Col}(\mathbf{Z})$. Then, if we multiply

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

by $\mathbf{P}^T$, we get

$$\mathbf{P}^\top(\mathbf{P}^\top\mathbf{M^{-1}P} + \mathbf{Q})\mathbf{A}\mathbf{v}_i = \mu_i\mathbf{P}^\top\mathbf{v}_i. \tag{8.24}$$

Now, we rewrite the left-hand side of Equation (8.24) to get

$$\begin{aligned}
\mathbf{P}^\top(\mathbf{P}^\top\mathbf{M^{-1}P} + \mathbf{Q})\mathbf{A} &= (\mathbf{P}^\top)^{\mathbf{2}}\mathbf{M^{-1}PA} + \mathbf{P}^\top\mathbf{QA} \\
&= \mathbf{P}^\top\mathbf{M^{-1}P^2A} + \mathbf{P}^\top\mathbf{QA} && \text{Lemma 3.4.2 e)} \\
&= \mathbf{P}^\top\mathbf{M^{-1}PAP}^\top + \mathbf{P}^\top\mathbf{QA} && \text{Lemma 3.4.2 f)} \\
&= \mathbf{P}^\top\mathbf{M^{-1}PAP}^\top. && \text{Lemma 3.4.2 k)}
\end{aligned}$$

Equation (8.24) can now be written as

$$\mathbf{P}^\top\mathbf{M^{-1}PA}\mathbf{w}_i = \mu_i\mathbf{w}_i, \tag{8.25}$$

where $\mathbf{w}_i := \mathbf{P}^\top\mathbf{v}_i$. Note that $\mathbf{w}_i \neq 0$ since $\mathbf{v}_i \notin \text{Col}(\mathbf{Z})$. Hence, μ_i is also an eigenvalue of $\mathbf{P}^\top\mathbf{M^{-1}PA}$. From Lemma 8.2.2 we have

$$\sigma(\mathbf{M^{-1}PA}) = \sigma(\mathbf{P}^\top\mathbf{M^{-1}PA}) \tag{8.26}$$

thus, μ_i is an eigenvalue of DEF1. □

In Lemma 8.2.4 we showed that the operators of Class 0 set the eigenvalues to zero and that the operators of Class 1 set them to one. Furthermore, the rest of the eigenvalues are the same for the two classes of operators. Note that, these results hold for an arbitrary choice of $\mathbf{Z}$.

Most of the methods presented in Table 8.1 belong to one of the above mentioned classes and we know their spectral behaviour, except for the $\mathcal{P}_{\text{SROM}}$ operator. Hence, to have a complete picture of the methods, we investigate the spectrum of $\mathcal{P}_{\text{SROM}}\mathbf{A}$ in Lemma 8.2.5.

Lemma 8.2.5. Assume that $\mathbf{M}^{-1}\mathbf{A} \in \mathbb{R}^{n\times n}$ has eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$ with λ_i corresponding to the eigenvector $\mathbf{v}_i$ of the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$, and let $\mathbf{M}$ be a traditional SPD preconditioner. If the deflation-subspace matrix is defined as $\mathbf{Z} = [\mathbf{v}_1 \ldots \mathbf{v}_p]$, with $\mathbf{M}^{-1}\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i$, then

$$\sigma(\mathcal{P}_{\text{SROM}}\mathbf{A}) = \{1, \ldots, 1, \lambda_{p+1}, \ldots, \lambda_n\}.$$

Proof. First, we assume that the spectrum of $\mathcal{P}_{\text{SROM}}\mathbf{A}$ is defined by

$$\sigma(\mathcal{P}_{\text{SROM}}\mathbf{A}) = \sigma\left(\mathbf{M}^{-1} + \mathbf{Q} - \frac{\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q}}{2}\right) = \{\mu_i, \ldots, \mu_n\},$$

and the spectrum of $\mathcal{P}_{\text{DEF2}}\mathbf{A}$ is defined by

$$\sigma(\mathcal{P}_{\text{DEF2}}\mathbf{A}) = \sigma(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{A}) = \{\nu_1, \ldots, \nu_n\},$$

where the eigenvalues are ordered such that $\nu_i = 0$ for $i = 1, \ldots, p$ and $\nu_i = \lambda_i$ for $i = p+1, \ldots, n$. Using Lemma 3.4.2 k) and Definition 3.4.1, we obtain the following expression:

$$\begin{aligned}\mathcal{P}_{\text{SROM}} =& \mathbf{M}^{-1} + \mathbf{Q} - \frac{1}{2}\big(\mathbf{Q}\mathbf{A}\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{A}\mathbf{Q}\big)\\ =& \mathbf{Q} + \frac{1}{2}\big(\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{P}\big).\end{aligned}$$

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

$$\begin{aligned}\mathcal{P}_{\text{SROM}}\mathbf{A}\mathbf{v}_i &= \left[\mathbf{Q} + \frac{1}{2}\big(\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{P}\big)\right]\mathbf{A}\mathbf{v}_i\\ &= \mathbf{v}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}\mathbf{v}_i && \text{Lemma 3.4.2 k)}\\ &= \mathbf{v}_i + \frac{1}{2}\lambda_i\mathbf{P}^\top\mathbf{v}_i = \mathbf{v}_i. && \text{Lemma 3.4.2 k)}\end{aligned}$$

Then $\mu_i = 1$ for $\mathbf{v}_i \in \text{Col}(\mathbf{Z})$, i.e., for $i = 1, \ldots, p$.

Let $\mathbf{v}_i \notin \text{Col}(\mathbf{Z})$ be an eigenvector corresponding to the eigenvalue μ_i, i.e., $i = p+1, \ldots, n$; then

$$\begin{aligned}&\left[\mathbf{Q} + \frac{1}{2}\big(\mathbf{P}^\top\mathbf{M}^{-1} + \mathbf{M}^{-1}\mathbf{P}\big)\right]\mathbf{A}\mathbf{v}_i = \mu_i\mathbf{v}_i\\ \Rightarrow\quad &\mathbf{P}^\top\mathbf{Q}\mathbf{A}\mathbf{v}_i + \frac{1}{2}(\mathbf{P}^\top)^2\mathbf{M}^{-1}\mathbf{A}\mathbf{v}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{P}\mathbf{A}\mathbf{v}_i = && \times\mathbf{P}^\top\\ &= \mu_i\mathbf{P}^\top\mathbf{v}_i\\ \Rightarrow\quad &\frac{1}{2}\lambda_i\mathbf{P}^\top\mathbf{v}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}\mathbf{P}^\top\mathbf{v}_i = \mu_i\mathbf{P}^\top\mathbf{v}_i && \text{Lemma 3.4.2 f), e) and k)}\\ \Rightarrow\quad &\frac{1}{2}\lambda_i\mathbf{w}_i + \frac{1}{2}\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}\mathbf{w}_i = \mu_i\mathbf{w}_i && \mathbf{w}_i := \mathbf{P}^\top\mathbf{v}_i\\ \Rightarrow\quad &\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}\mathbf{w}_i = (2\mu_i - \lambda_i)\mathbf{w}_i.\end{aligned}$$

Recall that $\mathbf{w}_i \neq 0$ since $\mathbf{v}_i \notin \text{Col}(\mathbf{Z})$. Then, an eigenvector of $\mathcal{P}_{\text{SROM}}\mathbf{A}$ is an eigenvector of $\mathbf{P}^\top\mathbf{M}^{-1}\mathbf{A}$, i.e., it is an eigenvector of the Class 1 operators, in particular of $\mathcal{P}_{\text{DEF2}}\mathbf{A}$.

The eigenvectors $\mathbf{v}_i$ correspond to the eigenvalues $2\mu_i - \lambda_i = \nu_i$, thus $\mu_i = \frac{1}{2}(\nu_i + \lambda_i)$. Recall that the spectrum is ordered such that $\nu_i = 0$ for $i = 1, \ldots, p$ and $\nu_i = \lambda_i$ for $i = p+1, \ldots, n$. This implies that, for $i = p+1, \ldots, n$, we have $\mu_i = \frac{1}{2}(\lambda_i + \lambda_i) = \lambda_i$, then, the spectrum of $\mathcal{P}_{\text{SROM}}\mathbf{A}$ is given by

$$\sigma(\mathcal{P}_{\text{SROM}}\mathbf{A}) = \{1, \ldots, 1, \lambda_{p+1}, \ldots, \lambda_n\}. \tag{8.27}$$

□

From Lemma 8.2.5 we can conclude that the two-level preconditioner of SROM belongs to the Class 1 when using eigenvectors of the preconditioned system as deflation vectors.

Next, we present some numerical results to test the theoretical results presented in this section.

8.3 Numerical comparison of the two-level preconditioners

In Section 8.2.1 we presented a theoretical comparison of the ROM and the A-DEF2 methods. In Lemma 8.2.1 we showed that the operators of both methods are the same $\mathcal{P}_{ROM} = \mathcal{P}_{A-DEF2}$, where the only difference is the starting or initial vector of the methods: $\mathbf{x}^0 = \overline{\mathbf{x}}$ for ROM, $\mathbf{x}^0 = \mathbf{Q}\mathbf{b} + \mathbf{P}^\top\overline{\mathbf{x}}$ for A-DEF2 (see [1]).

In this section, we present some numerical experiments that study the behaviour of these two methods, and to illustrate the theoretical results. The studied cases are a layered problem and the SPE 10 benchmark.

Layered problem. We consider a layered problem consisting of four layers with two different permeability values, $\sigma_1 = 0.510^{-3}$ D and $\sigma_2 = 100$ D, where the contrast in permeability between the layers is $c_\sigma = \sigma_1/\sigma_2 = \mathcal{O}(10^6)$. The permeability and the pressure fields are presented in Figure 8.1.

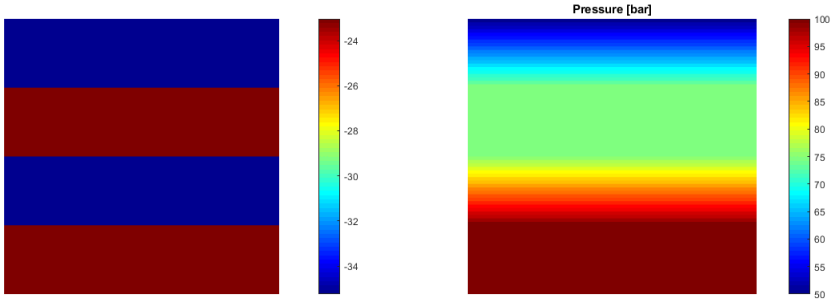


Figure 8.1: *Left: Permeability field, Right: Pressure field of the layered problem.*

Model. We study a Cartesian grid containing 40 x 40 cells. Therefore, the size of $\mathbf{A}$ is 1600×1600, and it has a condition number of $\kappa(\mathbf{A}) \approx \mathcal{O}(10^8)$. The pressures in the upper and lower boundaries are set as $p_{bc_u} = 50$ bars and $p_{bc_l} = 100$ bars.

Deflation vectors. We select 4 subdomain vectors as deflation vectors, where each deflation vector corresponds with one layer.

Solvers. In the first experiment, we compare the ROM method and the A-DEF2 method. As stopping criterion we use the true relative residual $r^t = 10^{-12}$, and the maximum number of iterations is 200. The initial guess is a random vector $\mathbf{x}^0 = \overline{\mathbf{p}}$, the same for all problems studied in this section.

In the previous section, we mentioned the need of a special starting vector before the iterative process for some 2L-PCG methods (see Table 8.2). In particular, for the A-DEF2 method, it is required to compute the starting vector $\mathbf{x}^0 = \mathbf{Q}\mathbf{q} + \mathbf{P}^\top\overline{\mathbf{p}}$. We also showed that the ROM and the A-DEF2 methods have the same operator, and as such, we expect the ROM method to require the same starting vector. Therefore, we study the influence of a special starting vector $\mathbf{x}^0$ on the ROM method.

Results. The residual is presented in Figure 8.2a for the two studied methods with different starting vector $\mathbf{x}^0_{(ROM)} = \overline{\mathbf{p}}$ and $\mathbf{x}^0_{(A-DEF2)} = \mathbf{Q}\mathbf{q}+\mathbf{P}^\top\overline{\mathbf{p}}$. In Figure 8.2b we present the results using the same special starting vector $\mathbf{x}^0 = \mathbf{Q}\mathbf{q}+\mathbf{P}^\top\overline{\mathbf{p}}$. The error, residual, flops and number of iterations are presented in Table 8.5 for both cases.

From Figure 8.2a we note that the ROM method does not converge when using a random starting vector $\mathbf{x}^0 = \overline{\mathbf{p}}$. Furthermore, we note in Figure 8.2b and in Table 8.5 that the result for A-DEF2 and ROM are the same when using the special starting vector $\mathbf{x}^0 = \mathbf{Q}\mathbf{q} + \mathbf{P}^\top\overline{\mathbf{p}}$, as expected from Section 8.2.1.

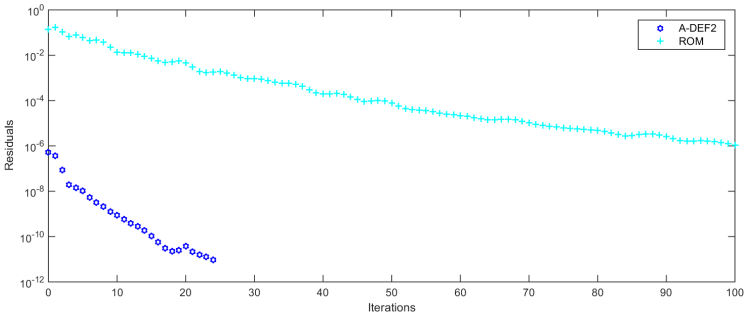
We note that a good starting vector is important for the ROM method to converge, as the SROM method is built from the ROM method, it is expected that the choice of the starting vector also influences the behaviour of the SROM method. In the next experiments, we investigate the influence of the starting vector on the ROM and SROM methods for the same test case.

Method	Error	Residual	Flops		# Iterations
			Initial	Iteration	
No special starting vector, $\mathbf{x}^0 = \overline{\mathbf{p}}$					
A-DEF2	8.10e-09	9.17e-12	$217n$	$62n$	24
ROM	1.58e-06	1.10e-06	$151n$	$56n$	NC
Special starting vector, $\mathbf{x}^0 = \mathbf{Q}\mathbf{q} + \mathbf{P}^\top\overline{\mathbf{p}}$					
A-DEF2	8.10e-09	9.17e-12	$217n$	$62n$	24
ROM	8.10e-09	9.17e-12	$211n$	$56n$	24

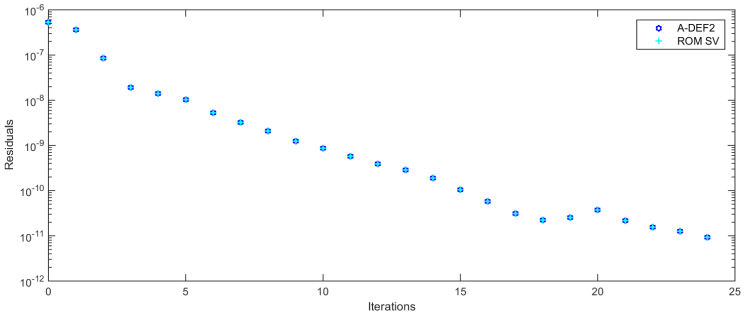
Table 8.5: *Comparison of A-DEF2 and ROM method using different starting vector.*

Special starting vector. As we mentioned before, we used a random vector as starting vector $\mathbf{x}^0 = \overline{\mathbf{p}}$, the same for all cases. However, we noted that it could lead to a strange behaviour of the ROM method. This issue can be overcome by using a special starting vector given by:

$$\mathbf{x}^0 = \mathbf{Q}\mathbf{q} + \mathbf{P}^\top\overline{\mathbf{p}}, \tag{8.28}$$



(a) $\mathbf{x}^0_{(ROM)} = \overline{\mathbf{p}}$, $\mathbf{x}^0_{(A-DEF2)} = \mathbf{Qq} + \mathbf{P}^\top \overline{\mathbf{p}}$.



(b) $\mathbf{x}^0_{(ROM,A-DEF2)} = \mathbf{Qq} + \mathbf{P}^\top \overline{\mathbf{p}}$.

Figure 8.2: *Comparison of the relative residuals of the A-DEF2 and ROM methods.*

where $\overline{\mathbf{p}}$ is an initial guess. In these experiments, $\overline{\mathbf{p}}$ is a random vector, the same for all cases. In practice, the test cases are time-dependent, and the previously computed solution is used as the initial guess $\overline{\mathbf{p}}$.

We perform a set of experiments to illustrate the behavior of the ROM and SROM methods using as starting vector $\mathbf{x}^0 = \overline{\mathbf{p}}$ and the special starting vector is given in Equation (8.28). The relative error, relative residual, flops and number of iterations are presented in Table 8.6, where the SV refers to the use of the special starting vector.

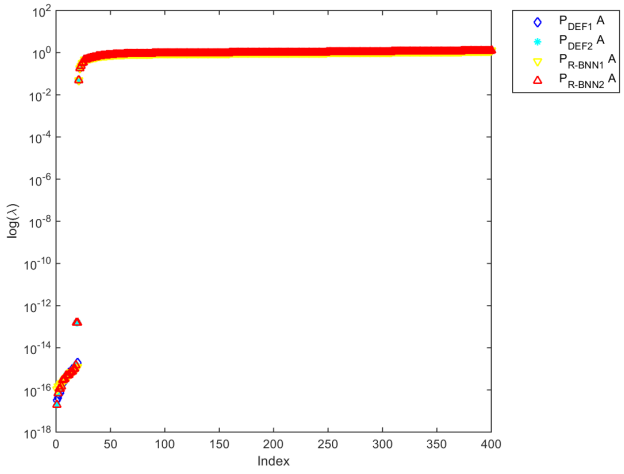
Method	Error	Residual	Flops		# Iterations
			Initial	Iteration	
ROM	1.58e-06	1.10e-06	$151n$	$56n$	NC
ROM SV	8.12e-09	9.17e-12	$211n$	$56n$	24
SROM	2.84e-09	8.66e-12	$175n$	$64n$	41
SROM SV	7.45e-09	9.05e-12	$299n$	$64n$	24

Table 8.6: *Comparison of the ROM and SROM method using different starting vectors.*

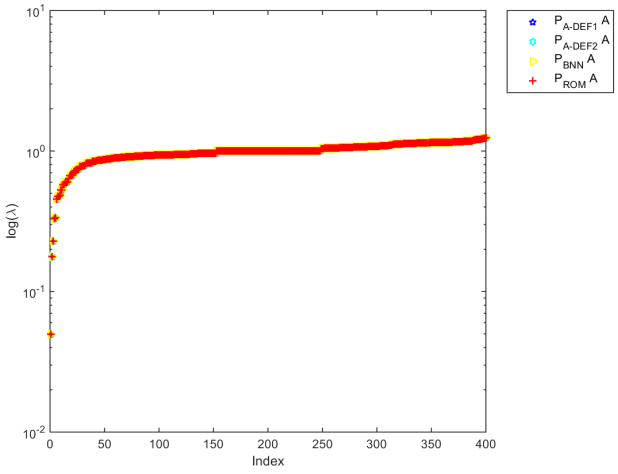
Note that, if a especial starting vector is used (see Equation (8.28)), convergence

of the ROM method is guaranteed. Furthermore, for the SROM method, the number of iterations decreases to almost half, with only a small increment of the initial flops. Thus, a especial starting vector leads to a better performance of the ROM and SROM methods. In the next section we study the spectral behaviour of the 2L-PCG methods, comparing subdomain and eigenvectors as deflation vectors.

Spectra analysis. In this section, we present some experiments to verify the theory presented in Section 8.2. The size of the problem is $\mathbf{A} \in \mathbb{R}^{400\times 400}$, and as traditional preconditioner we use IC_0 and 20 subdomain vectors as deflation vectors.



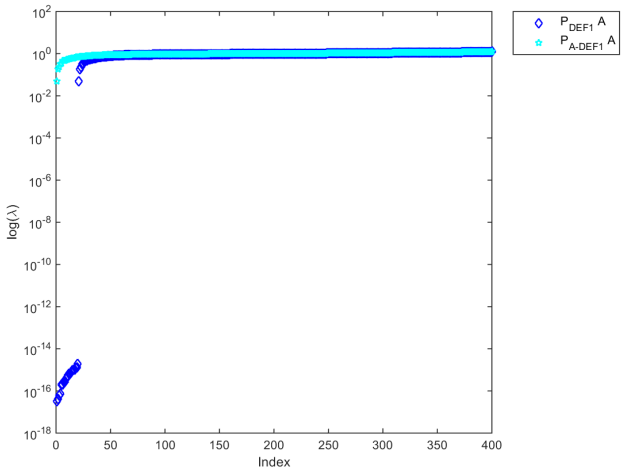
(a) *Spectrum of 2L-PCCG methods of Class 0.*



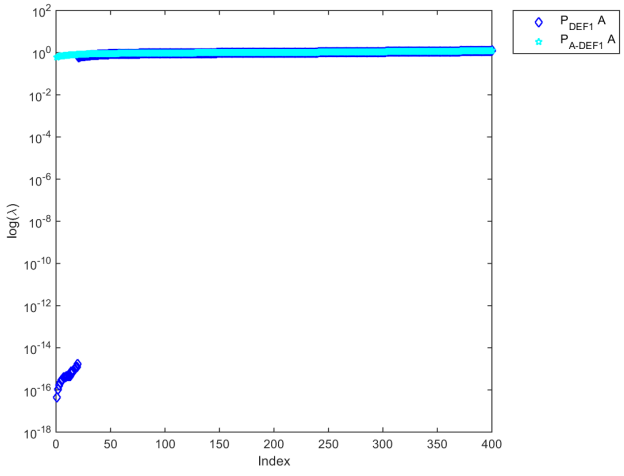
(b) *Spectrum of 2L-PCCG methods of Class 1.*

Figure 8.3: *Spectrum of different methods, 20 subdomain vectors as deflation vectors.*

We observe that the spectrum of the methods within each class are the same. Additionally, we note that, the Class 0 methods set the first 20 eigenvalues to zero, whereas, the Class 1 methods set them to one, validating Lemma 8.2.2, and 8.2.3.



(a) *20 subdomain vectors as deflation vectors.*



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

Figure 8.4: *Comparison of the spectrum of DEF1 and A-DEF1 for different choices of deflation vectors.*

In Lemma 8.2.4, we showed that, besides the eigenvalues deflated to zero or one (see Figure 8.3), the remaining eigenvalues are the same for both classes after the deflation procedure. This behavior is presented in Figure 8.4, where we compare the behavior of DEF1 from Class 0 and A-DEF1 from Class 1, using a) 20 subdomain vectors and d) 20 eigenvectors of preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ as deflation vectors.

The spectrum of the two classes of methods, Class 0: DEF1/2, and R-BNN1/2, and Class 1: A-DEF1/2, BNN and ROM, are presented in Figure 8.3 for each class. We observe, as in the previous example, that some eigenvalues are set to zero for DEF1 and to one for A-DEF-1. Moreover, we note that the rest of the spectrum is similar for both methods, regardless of the selection of deflation vectors, as expected from Lemma 8.2.4.

Spectral analysis of the SROM method. We proved, in Lemma 8.2.5, that the SROM method applied to the matrix $\mathbf{A}$ belongs to the Class 1 methods when using eigenvectors as deflation vectors, and thus, it has the same spectrum as ROM, A-DEF1/2 and BNN. In this section we illustrate this behavior with a set of numerical experiments.

Model. We study the layered cased presented above for a matrix with size 400 x 400, i.e., $\mathbf{A} \in \mathbb{R}^{400\times 400}$.

Solvers We study the performance of the SROM and the A-DEF1 methods using the Incomplete Cholesky with zero fill in $\mathbf{M} = IC_0$ preconditioner and two different choices of deflation vectors: 50 eigenvectors of the preconditioned system $\mathbf{M}^{-1}\mathbf{A}$ and 50 subdomain vectors as deflation vectors.

Results The spectrum of the systems preconditioned with SROM and A-DEF1 are presented in Figure 8.6, for this plot the scale on the y-axis is the same; however not all of the eigenvalues can be seen. The full spectrum is presented in Figure 8.5, where we note that some of the eigenvalues are still small.

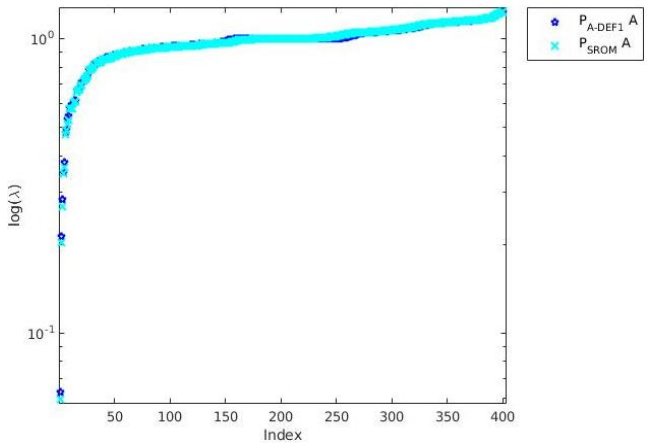
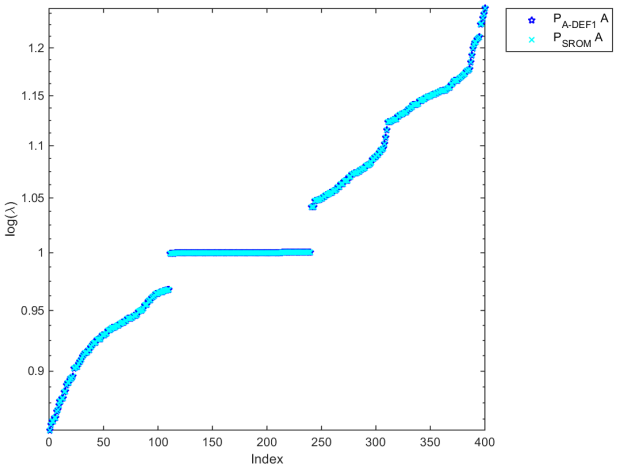


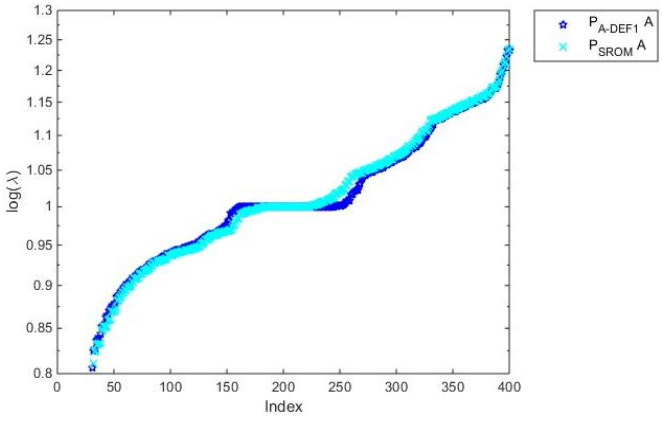
Figure 8.5: *Comparison of the spectrum of SROM and A-DEF1 for different choices of deflation vectors,* $\mathbf{M} = IC_0$*, full spectrum.*

Figure 8.6 shows that the spectrum of the $\mathcal{P}_{SROM}\mathbf{A}$ and the $\mathcal{P}_{A-DEF1}\mathbf{A}$ systems are similar. Moreover, we note that some eigenvalues are sent to one when using eigenvectors as deflation vectors, as expected from theory. Moreover, a similar behaviour occurs when using subdomain vectors as deflation vectors; however, for this case we

cannot select which eigenvalues are deflated, and some of the smallest eigenvalues remain unchanged, see Figure 8.5.



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



(b) *50 subdomain vectors as deflation vectors.*

Figure 8.6: *Comparison of the spectrum of SROM and A-DEF1 for different choices of deflation vectors,* $\mathbf{M} = IC_0$.

SPE 10 benchmark. In these experiments, we make a comparison of the methods when using POD basis vectors as deflation vectors. 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 ($P_{1:4}$). The permeability field and the location of the wells is the same as in Figure 5.9.

Model. We study the upper layer of the model with 60×220 cells. The condition number of the original matrix is $\kappa(\mathbf{A}) \approx 10^8$. Four production wells ($P_{1:4}$) are placed on the corners of the domain, and one injection well (I) is located in the center. Homogeneous Neumann boundary conditions are imposed in all boundaries.

We study waterflooding, and the properties of the oil and water are presented in Table 7.1. We use the fractional flow formulation defined in Chapter 2.

Snapshots. We run a full simulation, where the pressure at the injection well is maintained constant at $I = 1100$ bars, and the pressure of the production wells $P_{1:4}$ is varied between 137.5 - 275 bars every 2 time steps. The initial reservoir pressure is $P^0 = 500$ bars. The simulation is run during 600 time steps, with a time step of 100 days, until the water reaches all production wells.

Deflation vectors. A POD basis is obtained from the snapshots presented before, and five or ten POD basis vectors are used as deflation vectors.

Test case. We study the system matrix of an intermediate time step, 300-th time step, taking as initial guess the solution of the previous time step, 299-th.

Solvers. The training run is computed with the ICCG method with a tolerance of $r_k = 10^{-10}$. As stopping criterion, we use the true residual $r^t = 10^{-7}$, and the maximum number of iterations is 1500. For the test case, we use the 2L-PCG methods introduced in this chapter, and we compare their performance. We use the special starting vector introduced in Equation 8.28 for the ROM and SROM methods.

Results. The number of operations for the initialization of the methods and per iteration, together with the number of iterations required to converge are presented in Table 8.7. We present two cases with different number of deflation vectors $p = 5$ and $p = 10$.

In Figure 8.7, we plot the initial work and the total work per iteration, i.e., the number of iterations times the flops per iteration. We note that all the 2L-PCG methods perform better than the traditional PCG. However, the initialization work pf the PCG method is smaller than the 2L-PCG methods.

The percentage of ICCG work for all the methods is shown in Figure 8.8. The method that requires more work is the BNN method, 70% of the ICCG work. The DEF1, DEF2, and R-BNN2 methods are slightly better than the rest, as they reduce the percentage of ICCG work to around 40% when using five deflation vectors. If the number of deflation vectors increases, a further reduction of 0.3% is achieved for these methods.

Increasing the number of deflation vectors to ten reduces the 47% of ICCG work required with the ROM method in around 10%, and 47% of ICCG work of the A-DEF2 method in around 0.1%. The A-DEF2 and R-BNN1 methods behave similarly, reducing the ICCG work to 45%, and do not change significantly when increasing the number of deflation vectors. The work of the SROM method is also reduced from 56% to 54% the ICCG work when the number of deflation vectors increases.

Method	p	Flops		Iterations
		Initial	Iteration	
PCG		$28n$	$30n$	252
DEF1	5	$207n$	$49n$	55
	10	$646n$	$69n$	31
DEF2	5	$246n$	$49n$	55
	10	$646n$	$69n$	31
A-DEF1	5	$227n$	$69n$	55
	10	$607n$	$109n$	32
A-DEF2	5	$266n$	$69n$	55
	10	$686n$	$109n$	31
BNN	5	$247n$	$89n$	55
	10	$647n$	$149n$	31
R-BNN1	5	$266n$	$69n$	55
	10	$686n$	$109n$	31
R-BNN2	5	$246n$	$49n$	55
	10	$646n$	$69n$	31
ROM	5	$217n$	$59n$	55
	10	$432n$	$79n$	31
SROM	5	$229n$	$71n$	55
	10	$509n$	$111n$	31

Table 8.7: *Comparison of the performance of the methods when using 5 and 10 POD basis vectors as deflation vectors.*

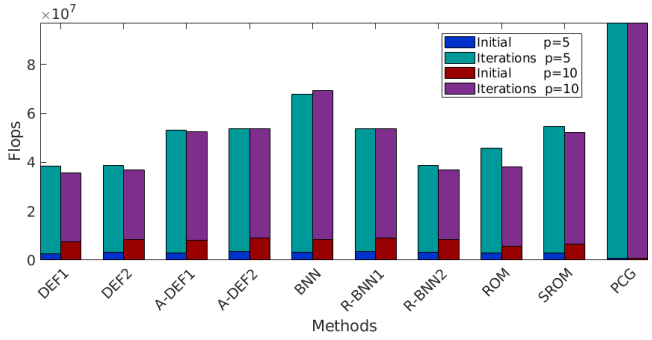


Figure 8.7: *Comparison of work of 2L-PCG methods, five and ten deflation vectors.*

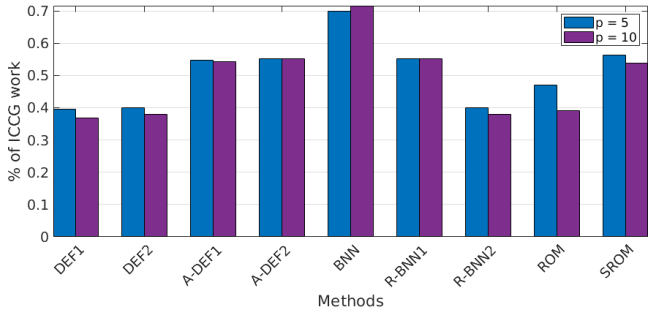


Figure 8.8: *Percentage of ICCG work of the 2L-PCG methods.*

Concluding remarks. In this chapter, we introduced the two-level Preconditioned Conjugate Gradient methods (2L-PCG). We described the deflation method used throughout this work (DEF1) in this context. Additionally, we introduce various 2L-PCG methods: PCG, DEF1/2, A-DEF1/2, BNN, R-BNN1/2, ROM and SROM, and we compared them theoretically and numerically.

We proved that the ROM method has the same operator as the A-DEF2 method; furthermore, we showed that the symmetric version of this method, the SROM method, consists on an additive version of the A-DEF1 and A-DEF2 methods.

We divided the methods into two classes: the ones that set the eigenvalues of the system matrix to zero (Class 0) and the ones that set them to one (Class 1). We showed that the DEF1 method belongs to the Class 0 and the ROM method belongs to the Class 1. The SROM method also belongs to Class 1 only if the eigenvalues of the preconditioned system matrix were selected as deflation vectors.

We illustrated the theoretical results with numerical experiments for a layered problem and the SPE 10 benchmark. The results of the numerical experiments showed that for the ROM and SROM method, the same special starting vector used for the A-DEF2 method is required to achieve optimal performance. We also observed that all methods have a similar performance, reducing the ICCG work to 37%-71%. The best performance was achieved with the DEF1, DEF2, and R-BNN2 methods requiring only from 37% to 40% of the ICCG work. The most expensive method was the BNN method requiring from 70% to 72% of the ICCG work.

We also studied the influence of the number of deflation vectors in the performance of the methods. For the DEF1, DEF2, A-DEF1, R-BNNN2, ROM and SROM methods the work was reduced if the number of deflation vectors was increased, for the A-DEF1 and R-BNN1 the change was unnoticeable, and for the BNN it was increased.

Chapter 9

Conclusions

9.1 Concluding remarks

POD-based deflation method. In this work, we developed and explored the possibilities of combining POD and deflation techniques for the acceleration of the solution of large and ill-conditioned linear systems. We proposed to obtain a set of deflation vectors containing relevant system information via recycling using a linearly independent (*l.i.*) set of snapshots with slightly different parameters that span the solution space, and from a POD basis obtained from an arbitrary set of snapshots. We also proposed various approaches for snapshots collection.

The convergence properties of the resulting POD-based deflation method were studied for incompressible and compressible single-phase, and incompressible two-phase flow problems. We studied heterogeneous porous media presenting a contrast in permeability coefficients up to $\mathcal{O}(10^7)$, leading to condition numbers of the system matrix of $\mathcal{O}(10^8)$. In particular, we studied an academic layered problem with various contrasts between permeability layers and the SPE 10 benchmark.

We showed that the deflation method converges in one iteration for a *l.i.* set of snapshots spanning the solution spaces, Lemma 4.1.3. To find the *l.i.* set spanning the solution space it is necessary to consider the number of sources, represented as non-zero elements in the right-hand side (rhs), and the boundary conditions of the studied system. For reservoirs presenting homogeneous Neumann boundary conditions and p wells, the *l.i.* set consisted on $p-1$ snapshots, Lemma 4.1.4. By contrast, if the reservoir has non-homogeneous Dirichlet boundary conditions and p wells, $p+1$ snapshots were required, Lemma 4.1.5.

It was also proved that to achieve the optimal performance of the deflation method, the snapshots have to be computed with certain accuracy. This accuracy depends on the condition number of the system under consideration, 4.1.6. Similar performance was observed if a POD basis obtained from a set of snapshots containing the *l.i.* set. Therefore, the information obtained from the *l.i.* set of snapshots, was captured in the POD basis. This result is especially useful when the selection of the *l.i.* set is not straightforward. Moreover, it can give us an idea of how the POD-deflation method works.

Incompressible single-phase problem. We studied the behavior of the deflation method simulating an incompressible fluid. We compared the performance of the method with different deflation vectors: eigenvalues of the system matrix and the preconditioned system matrix; subdomain deflation vectors; a *l.i.* set of snapshots spanning the solution space and a POD basis obtained from a set of system-related snapshots. The best performance was achieved when using the *l.i.* set and the POD basis, requiring only one iteration, as expected from theory.

The performance of the deflation method with these particular sets of deflation vectors was independent of the boundary conditions, the contrast between permeability coefficients, or the size of the problem. Our largest test case was the SPE 10 model, containing $\mathcal{O}(10^6)$ cells, for which we also achieved convergence in one iteration.

For the layered test case, we observed that after preconditioning, some system eigenvalues became small. Moreover, the largest contrast between permeability layers generated the smallest eigenvalues. These small eigenvalues were effectively removed after deflation, i.e., set to a value close to zero, improving the condition number and accelerating the solution.

Compressible single-phase problem. For a compressible fluid, we reduced the number of DICCG iterations up to 20% of the number of ICCG iterations when using the POD-based deflation method. The best performance was achieved using four deflation vectors. In this case, each DICCG iteration needs around 1.4 times the number of operations required with the ICCG method (see Table 4.1).

Incompressible two-phase problem. We implemented the POD-based method for the solution of two-phase flow simulation problems. Our primary focus was the solution of the pressure equation. With this method, we reduced the number of ICCG iterations up to 7% and the ICCG work during the iteration process was reduced to 11%, using a stopping criteria of 10^{-4}. Furthermore, for most of the cases, the true error of the approximation obtained after the first DICCG iteration was smaller than 10^{-4}. This implies that, if this accuracy is required, only one DICCG iteration is necessary.

Including capillary pressure terms slightly deteriorates the performance of the DICCG method, i.e., the number of iterations increased. The largest increment in the number of iterations was 5% the number of ICCG iterations. The largest increment in work during the iteration process was 11% of the ICCG work.

The slower convergence suggests that the eigenvectors of the correlation matrix capture less information of the process, which influences the performance of the deflation method. This behavior could be related to the step size of the simulation; it is possible that some phenomena cannot be captured accurately in this time-scale and therefore not all the information is contained in the basis.

Regarding the snapshots collection, for problems without capillary pressure and injection of water through wells, the moving window approach (MW) resulted in less work during the iteration process when compared with the training phase approach (TP). However, the computational cost of obtaining the basis was larger, and the basis obtained with the TP approach can be used to solve more than one problem.

Using 10 POD basis vectors as deflation vectors, the MW approach required 11% of the ICCG iterations while the TP approach required 13-16 % for a 2D case. These

percentages increased when reducing the number of deflation vectors to 13% for the MW approach and to 19-24% for the TP approach.

Results showed that the information acquired during the training phase could be used to solve problems with diverse flow patterns, even different than the ones used during the pre-simulation.

When including gravity terms, if the size of the reservoir increases, the number of iterations is further reduced by the DICCG method. For a cell's length of 1 [m], the number of ICCG iterations was reduced to 20% and the ICCG work was reduced to 40%. When the length of the cells is 4 [m], the number of ICCG iterations was reduced to 10% and the ICCG work is reduced to 20%.

If capillary pressure terms were included, selecting a water's Corey coefficient of 2, the DICCG method reduced the number of ICCG iterations to 34%. Increasing this coefficient to 4, the reduction was 40%. This implies that the Corey coefficients influence the DICCG method, the smaller the Corey coefficients are, the better performance of the DICCG method was achieved. This behavior was observed for a stopping criterion of 10^{-7}. As in the previous cases, after one DICCG iteration, the error of the approximation is of order 10^{-4}.

Comparison of 2L-PCG methods. Besides the performance of the POD-based Preconditioned Conjugate Gradient (PCG) deflated method, we studied it as part of a family of preconditioners, referred to as 2L-PCG methods. They consist of a traditional single-level preconditioner, Incomplete Cholesky (IC) in our case; and a second-level preconditioner, the deflation matrix $\mathbf{P}$.

The deflation method used in this work has several variants, the one implemented in this work is referred to as DEF1 (see [53]). We compared the performance of various 2L-PCG methods including DEF1 and a recently-developed preconditioner based on AMG methods (the ROM method), introduced by Pasetto et al. [1]. The comparison was made regarding the operators, the influence on the system matrix's spectrum, and the work required to perform each method.

The ROM method was proved to be a 2L-PCG method. Furthermore, the operator of this method was equivalent to the operator of the A-DEF2 deflation variant, described in [53] and Chapter 8 of this work. The operator of the symmetric version of the ROM method, the SROM method, was also described in terms of the A-DEF1 and A-DEF2 deflation variants.

The only difference between the ROM methods and the deflation variants was the implementation. A particular starting vector was required for the implementation of the A-DEF1 method, which also improved the performance of the ROM methods.

We studied two classes of 2L-PCG methods. The Class 0 methods set the smallest eigenvalues to zero, and the Class 1 methods set them to one when using an arbitrary set of deflation vectors [53]. We showed that the ROM method belongs to the Class 1 methods, and the SROM methods set them to one only if the deflation vectors are eigenvalues. Otherwise, some arbitrary eigenvalues were set to one.

All the methods showed a similar performance in terms on the number of iterations when using a POD basis as deflation vectors, reducing the number of ICCG iterations to 20% when using five deflation vectors and to 12% when using ten.

Regarding the work, they showed similar results among them and also when changing the number of deflation vectors. However, the best performance was achieved with

the DEF1, DEF2, and R-BNN2 methods requiring only from 37% to 40% of the ICCG work. The most expensive method was the BNN method requiring from 70% to 72% of the ICCG work.

Applicability of the POD-based deflation method. This dissertation was devoted to the acceleration of the solution of Symmetric Positive Definite (SPD) linear systems. In particular, we based our work on the acceleration of the PCG method. For most of the experiments, we used the DEF 1 deflation variant introduced in Section 3.4.2 as acceleration strategy.

In Chapter 8, we show that this methodology can also apply to diverse 2L-PCG methods. In particular, the method resulted in a good performance for various 2L-PCG methods. Therefore, it could be implemented together with many other similar methodologies, e.g., Multigrid, Multilevel or Domain Decomposition strategies.

Moreover, it was tested for reservoir simulation problems, but its applicability does not depend on the problem under investigation; therefore, it can be used for any transient problem. Hence, it is a very versatile methodology.

9.2 Future research and recommendations

POD basis. In this work, we showed that using a POD basis as deflation vectors leads to a good acceleration. We suggest further theoretical and experimental research on this subject. The suggested studies are:

Theoretical study of the POD basis used as deflation vectors. Selecting the set of snapshots spanning the solution space as deflation vectors, we showed that the solution is achieved in one iteration (see Lemma 4.1.3). A similar study for the POD basis has to be performed.

Influence of capillary and gravity terms. We observed that changing the capillary pressure or the size of the reservoir changes the performance of the method. However, we do not know why this happens and what is the relation between these parameters and the performance of the method. More studies are required to understand this behavior.

The influence of perturbing the matrix and the right-hand side by a parameter ϵ has to be addressed to understand the behavior of the method under these changes. Such a study could give us an idea of the influence of gravity and capillary terms.

Influence of time stepping. Computation of the POD basis was obtained from a set of snapshots. These snapshots capture the main features of the system. However, some processes are slower than others, and different time stepping has to be done to capture the main system flow mechanisms accurately.

To understand this influence, studies with different time steps are required.

Influence of permeability field. In this dissertation, we implemented the POD-based deflation method for a layered academic problem and the SPE 10 method. However, we did not study the influence of the permeability field. To gain more

insight, studies with diverse permeability fields are required. Among others, using structured subdomains with different permeability coefficients and random fields can be tested.

Solution as initial guess. Results showed that the first DICCG iteration gives an approximation of $\mathcal{O}(10^{-4})$. Therefore, when requiring this accuracy (usual accuracy to solve engineering problems), this method obtained the results very efficiently. However, if a more accurate solution is required, this method can be used to obtain an initial guess, for a further acceleration with any other method.

Further development of the methodology. New and powerful methods have been recently studied to accelerate the solution of reservoir simulation problems. Among others, parallelization [55–58], and machine learning [59–61] can be implemented together with the POD-basis methodology developed in this work.

Furthermore, many state-of-the-art Reduced Order Model techniques (ROM) [62, 63] and various preconditioners like Multigrid methods can also be combined with deflation methods.

Bibliography

[1] D. Pasetto, M. Ferronato and M. Putti. A reduced order model-based preconditioner for the efficient solution of transient diffusion equations. *International Journal for Numerical Methods in Engineering*, 109(8):1159–1179, 2017.

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

[3] D.W. Peaceman. *Fundamentals of Numerical Reservoir Simulation.* Elsevier, 1977.

[4] L.W. Lake. *Enhanced Oil Recovery.* Prentice Hall, 1989.

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

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

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

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

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

[10] 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.

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

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

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

[14] P. Dechaumphai and N. Wansophark. *Numerical Methods in Engineering: Theories with MATLAB, Fortran, C and Pascal Programs.* Alpha Science International, 2011.

[15] R.L. Burden and J.D. Faires. *Numerical Analysis.* Brooks/Cole Cengage Learning, Ninth edition, 2010.

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

[17] J. van Kan, A. Segal, and F. Vermolen. *Numerical methods in scientific computing.* VSSD, 2008.

[18] G.B. Golub and C.F. Van Loan. Matrix Computations. *Johns Hopkins University*, 1983.

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

[20] K. Carlberg, V. Forstall and R. Tuminaro. Krylov-subspace recycling via the POD-augmented conjugate-gradient method. *SIAM Journal on Matrix Analysis and Applications*, 37(3):1304–1336, 2016.

[21] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid.* Elsevier, 2000.

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

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

[24] V. Dolean, P. Jolivet, and F. Nataf. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*, volume 144. SIAM, 2015.

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

[26] M.A. Cardoso, L.J. Durlofsky and P. Sarma. Development and application of reduced-order modeling procedures for subsurface flow simulation. *International journal for numerical methods in engineering*, 77(9):1322–1350, 2009.

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

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

[29] P.T.M. Vermeulen, A.W. Heemink and C.B.M. Te Stroet. Reduced models for linear groundwater flow models using empirical orthogonal functions. *Advances in Water Resources*, 27(1):57–69, 2004.

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

[31] C. Vuik, A. Segal and J.A. Meijerink. An Efficient Preconditioned CG Method for the Solution of a Class of Layered Problems with Extreme Contrasts in the Coefficients. *Journal of Computational Physics*, 152:385, 1999.

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

[33] G.B. Diaz Cortes, C. Vuik and J.D. Jansen. Physics-based pre-conditioners for large-scale subsurface flow simulation. Technical report, Delft University of Technology, Department of Applied Mathematics, 03 2016.

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

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

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

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

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

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

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

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

[42] I. Ipsen and C. Meyer. The idea behind Krylov methods. *The American mathematical monthly*, 105(10):889–899, 1998.

[43] L. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. SIAM, 1997.

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

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

[46] 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. Report 18-1, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2018.

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

[48] J. Tjan. Study on deflation techniques and POD methods for the acceleration of Krylov subspace methods. Master's thesis, TU Delft, 8 2018. `web:http://ta.twi.tudelft.nl/nw/users/vuik/numanal/tjan_scriptie.pdf`.

[49] G.B. Diaz Cortes, J.D. Jansen, and C. Vuik. On The Acceleration Of Ill-Conditioned Linear Systems: A Pod-Based Deflation Method For The Simulation Of Two-Phase Flow. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, 2018.

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

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

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

[53] J.M. Tang, R. Nabben, C. Vuik, and Y.A. Erlangga. Theoretical and numerical comparison of various projection methods derived from deflation, domain decomposition and multigrid methods. *Reports of the Department of Applied Mathematical Analysis, 07-04*, 2007.

[54] R. Nabben, and C. Vuik. A comparison of deflation and the balancing preconditioner. *SIAM Journal on Scientific Computing*, 27(5):1742–1759, 2006.

[55] J. E. Killough, D. and D. Camilleri, L. Bruce, J. Foster,et all. A parallel reservoir simulator based on local grid refinement. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 1997.

[56] J.H. van der Linden, T.B. Jönsthövel, A.Lukyanov, Alexander, and C. Vuik. The parallel subdomain-levelset deflation method in reservoir simulation. *Journal of Computational Physics*, 304:340–358, 2016.

[57] Frank, J. and Vuik, C. On the Construction of Deflation-Based Preconditioners. *SIAM Journal on Scientific Computing*, 23(2):442–462, 2001.

[58] S. Gries et al. On the Convergence of System-AMG in Reservoir Simulation. *SPE Journal*, 2018.

[59] P. Borrel, M. Kormaksson, C. Paz and C. N. Mena. Machine learning assisted reservoir simulation, 2018. US Patent App. 15/197,734.

[60] H. Zalavadia and E. and Gildin. Parametric Model Order Reduction For Adaptive Basis Selection Using Machine Learning Techniques During Well Location Opt. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, 2018.

[61] P. Tahmasebi, F. Javadpour, and M. Sahimi. Data mining and machine learning for identifying sweet spots in shale reservoirs. *Expert Systems with Applications*, 88:435–447, 2017.

[62] L. Iapichino, S. Ulbrich, and S. Volkwein. Multiobjective PDE-constrained optimization using the reduced-basis method. *Advances in Computational Mathematics*, 43(5):945–972, 2017.

[63] M.A. Cremon, M. Christie, M.G. Gerritsen. Monte Carlo Simulation For Uncertainty Quantification In Reservoir Simulation: A Convergence Study. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, 2018.

[64] G.E. Plassman. A survey of singular value decomposition methods and performance comparison of some available serial codes. (NASA/CR-2005-213500), 2005.

Appendix 1. List of notation

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

Table 1: *Notation*

Appendix 2. Stopping criteria

When implementing iterative methods, we obtain an approximate solution that is close enough to the exact solution. In other words, we want that the error [8, pag. 42]:

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

or the relative error:

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

is small.
Choosing the relative error a stopping criterion is the best way to know the real accuracy of our approximation. However, the exact solution has to be computed. Therefore, a new stopping criterion has to be selected.

A common choice is the residual

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

or the relative residual, that is actually computed at each iteration. The relation between the error and the residual is given by:

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

that depends on the condition number of the system matrix. Taking this relation into account, we can choose the stopping criteria as an ϵ for which

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

And the relative error will be bounded by:

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

Appendix 3. Complexity

This appendix is devoted to the computation of the number of operations necessary to implement the methods studied throughout this work. Table 2 presents the number of flops for the most common vectors and matrix operations and deflation operators.

For the implementation of the POD-based deflation method, first, we need to compute the snapshots with the ICCG method, the algorithm and the flop count of this method is presented in Algorithm 11.

Once the snapshots are computed, we obtain the eigenvalues (Λ) and eigenvectors ($\mathbf{V}$) of the covariance matrix $\overline{\mathbf{R}} \in \mathbb{R}^{n\times n}$ by computing the SVD of $\overline{\mathbf{R}}^T = \overline{\mathbf{X}}\overline{\mathbf{X}}^T \in \mathbb{R}^{p\times p}$. The eigenvalues of both matrices are the same and the eigenvectors of $\overline{\mathbf{R}}$ can be computed from:

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

where $\mathbf{V}$ are the eigenvectors of $\overline{\mathbf{R}}^T$ (see [11, 46]). The eigenvectors corresponding to the largest eigenvalues are selected as the POD basis. The operational cost of this process is presented in Table 2.

The POD basis recently computed is then used as deflation subspace matrix $\mathbf{Z}$ and, depending on the approach, the next time steps (moving window) or a new simulation (training phase) is solved with DICCG. Algorithm 12 shows the number of flops required for the implementation of this method. Table 2 presents a comparison of flops between the deflated and the non-deflated methods. The flops are computed for full matrices and sparse matrices with m non-zero diagonals, and p deflation vectors.

For the computation of the number of flops of the DICCG method, we assume that the matrix $\mathbf{Z}$ is already give, i.e. it does not change during the iteration process. Under this assumption, we can compute a set of matrices that will reduce the number of operations of the deflation procedure. These matrices are the following: $\mathbf{V}_Z = \mathbf{AZ}$, $\mathbf{E} = \mathbf{Z}^T\mathbf{AZ} = \mathbf{Z}^T\mathbf{V}_Z$, $\mathbf{E}^{-1}$ and $\mathbf{B} = \mathbf{AZE}^{-1} = \mathbf{V}_Z\mathbf{E}^{-1}$. The cost of computing these matrices is computed as initial work. The preconditioning process $\mathbf{y} = \mathbf{M}^{-1}\mathbf{x}$ is performed by factorizing the system matrix in his upper and lower parts $\mathbf{A} = \mathbf{LL}^T$ and solving the system $\mathbf{LL}^T\mathbf{y} = \mathbf{x}$. For this process, the factorization is considered as initial work, and the only work considered during the iteration is related to the solution of $\mathbf{y} = (\mathbf{LL}^T)^{-1}\mathbf{x}$ with backward and forward substitution.

Basic operations, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{L}$, $\in \mathbb{R}^{n\times n}$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$		
Operation	Flops	
	Full matrix	$\mathbf{A}$ sparse (m)
$\mathbf{x}^T\mathbf{y}$	$2n-1$	$2n-1$
$\mathbf{x}(+/-)\mathbf{y}$	n	n
$\alpha\mathbf{x}$	n	n
$\mathbf{Ax}$	$2n^2-n$	$(2m-1)n$
$\mathbf{AB}$	$2n^3-n^2$	$(2m-1)n^2$
$\mathbf{A}=\mathbf{LL}^T$	$1/3n^3$	nm^2
$(\mathbf{L}/\mathbf{L}^T)\mathbf{x}=\mathbf{y}$	n^2	nm
$\mathbf{x}=(\mathbf{LL}^T)^{-1}\mathbf{y}$	$2n^2$ (forward and backward subs)	$2nm$
$\mathbf{A}\in\mathbb{R}^{n\times n}$, $\mathbf{B}\in\mathbb{R}^{n\times p}$, $\mathbf{x}\in\mathbb{R}^p$		
$\mathbf{AB}$	$(2n-1)pn$	$(2m-1)pn$
$\mathbf{BB}^T=\mathbf{V}\Sigma\mathbf{V}^T$	$6n^3$[64]	$6n^3$[64]
Compute the POD basis from $\mathbf{X}^{1:p}=\{\mathbf{x}^1,\mathbf{x}^2,...,\mathbf{x}^p\}, \mathbf{x}\in\mathbb{R}^n$		
Computing $\mathbf{U}\in\mathbb{R}^{n\times n}$ from $\mathbf{V}\in\mathbb{R}^{s\times s}$. $\Lambda^T\in\mathbb{R}^{s\times s}$, $\mathbf{X}\in\mathbb{R}^{n\times s}$, s snapshots		
$(\Lambda^T)^{\frac{1}{2}}$		s
$\mathbf{V}_\Sigma=\mathbf{V}\Sigma$		$(2s-1)s^2$
$\mathbf{XV}_\Sigma$		$(2s-1)sn$
Total $\mathbf{U}^1=\mathbf{XV}(\Lambda^T)^{\frac{1}{2}}$		$[(2s-1)(n+s)+1]s$
$\overline{\mathbf{x}}=\frac{1}{s}\sum_{j=1}^{s}\mathbf{x}^j$		$n(s-1)+1$
$\overline{\mathbf{x}}^j=\mathbf{x}^j-\overline{\mathbf{x}}$		ns
$\overline{\mathbf{R}}^T=\overline{\mathbf{X}}^T\overline{\mathbf{X}}$		$(2n-1)s^2$
$\mathbf{V}\Lambda\mathbf{V}^T=\overline{\mathbf{R}}^T$		$6s^3$
$\mathbf{U}^1$ (from $\mathbf{V}$)		$[(2s-1)(n+s)+1]s$
Total		$(4s^2+s-1)n+$ $+(8s^2-2s+1)s+1$
Deflation operations, $\mathbf{Z},\mathbf{B},\mathbf{V}\in\mathbb{R}^{n\times p}$, $\mathbf{E}\in\mathbb{R}^{p\times p}$, $\mathbf{x}\in\mathbb{R}^n$, and $\mathbf{y}\in\mathbb{R}^p$ (If the matrix $\mathbf{Z}$ is not changing).		
$\mathbf{x}_Z=\mathbf{Z}\mathbf{y}_Z$		$(2p-1)n$
$\mathbf{y}_Z=\mathbf{Z}^T\mathbf{x}_Z$		$(2n-1)p$
$\mathbf{E}^{-1}$		$\frac{p^3}{3}$
$\mathbf{w}=\mathbf{E}^{-1}\mathbf{y}$, $\mathbf{E}^{-1}$ already computed		$(2p-1)p$
$\mathbf{V}=\mathbf{AZ}$, $\mathbf{A}$ sparse		$(2m-1)np$
$\mathbf{B}=\mathbf{AZE}^{-1}$, $\mathbf{AZ}$ and $\mathbf{E}^{-1}$ already computed		$(2p-1)np$

Table 2: *Flops of the operations*

Deflation operators, with $\mathbf{E}^{-1} \in \mathbb{R}^{p\times p}$, $\mathbf{ZE}^{-1} \in \mathbb{R}^{n}$, and $\mathbf{B} = \mathbf{AZE}^{-1} \in \mathbb{R}^{n\times p}$ already computed.	
$\mathbf{Qx} = (\mathbf{ZE}^{-1})(\mathbf{Z}^T\mathbf{x})$	$(2p-1)n + (2n-1)p = (4p-1)n - p$
$\mathbf{Px} = \mathbf{I} - \mathbf{B}(\mathbf{Z}^T\mathbf{x})$	$n + (2n-1)p + (2p-1)n = (4n-1)p$
$\mathbf{P}^T\mathbf{x} = \mathbf{I} - \mathbf{Z}(\mathbf{B}^T\mathbf{x})$	$n + (2n-1)p + (2n-1)n = (4n-1)p$

Table 3: *Flops of the deflation operators*

Compute true residual $\mathbf{r}_k = \frac{\|\mathbf{b}-\mathbf{Ax}^k\|_2}{\|\mathbf{b}\|_2}$		
Operation	Flops	
	Full matrix	**A** sparse (m)
$\mathbf{Ax}_k$	$(2n-1)n$	$(2m-1)n$
$\mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k$	n	n
$\|\|\mathbf{r}_k\|\|_2$	$2n-1$	$2n-1$
$\|\|\mathbf{b}\|\|_2$	$2n-1$	$2n-1$
Total	$(2n^2+4)n-2$	$(2m+4)n-2$

Table 4: *Work computing* $\mathbf{r}_k$.

Solving $\mathbf{Ax} = \mathbf{b}$; $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$.	
Initial work	
ICCG	$(m^2+4m+2)n$
DICCG	$(m^2+4m+2)n + (4p^2+2mp+2p)n + \frac{p^3}{3} - p^2 - p$
$\frac{\text{DICCG}}{\text{ICCG}}$	$1 + \frac{4p+2m+2}{m^2+4m+2} + \frac{(\frac{p^2}{3}-p-1)p}{(m^2+4m+2)n}$
SVD	$[(8s-1)n + 10s^2 - 2s + 1]s$, s is the No. of snapshots
Work per iteration	
ICCG	$(4m+9)n$
DICCG	$(4p+4m+9)n - p$
$\frac{\text{DICCG}}{\text{ICCG}}$	$\sim 1 + \frac{4p}{4m+9}$

Table 5: *Comparison of flops required with the ICCG and the DICCG methods.*

Algorithm 11 ICCG method, solving $\mathbf{Ax} = \mathbf{b}$; $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$.

Initial work

Operation	Flops	
	Full **A**	*Sparse* **A** *m non-zero diag*
Initial guess $\mathbf{x}_0$.		
$\mathbf{M} = \mathbf{L}_0\mathbf{L}_0^T$	$1/3n^3$	nm^2

Initialization of variables

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$	$2n^2$	$2nm$
$\mathbf{y}_0 = \mathbf{M}^{-1}\mathbf{r}_0$	$2n^2$	$2mn$
$\mathbf{r}_0 \cdot \mathbf{y}_0 = (\mathbf{r}_0, \mathbf{y}_0)$	$2n - 1$	$2n - 1$
$\mathbf{p}_0 = \mathbf{y}_0$		
Initialization of variables	$(4n + 2)n - 1$	$(4m + 2)n - 1$
Total initial work	$(\frac{1}{3}n^2 + 4n + 2)n - 1$	$[(m + 4)m + 2]n - 1$

Work per iteration

for $k = 0, ...,$ until convergence		
$\mathbf{w}_k = \mathbf{A}\mathbf{p}_k$	$2n^2 - n$	$(2m - 1)n$
$\mathbf{r}_k \cdot \mathbf{w}_k = (\mathbf{r}_k, \mathbf{w}_k)$	$2n - 1$	$2n - 1$
$\alpha_k = \frac{\mathbf{r}_k \cdot \mathbf{y}_k}{\mathbf{r}_k \cdot \mathbf{w}_k}$	1	1
$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$	$2n$	$2n$
$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\mathbf{w}_k$	$2n$	$2n$
$\mathbf{y}_{k+1} = \mathbf{M}^{-1}\mathbf{r}_{k+1}$	$2n^2$	$2mn$
$\mathbf{r}_{k+1} \cdot \mathbf{y}_{k+1} = (\mathbf{r}_{k+1}, \mathbf{y}_{k+1})$	$2n - 1$	$2n - 1$
$\beta_k = \frac{\mathbf{r}_{k+1} \cdot \mathbf{y}_{k+1}}{\mathbf{r}_k \cdot \mathbf{y}_k}$	1	1
$\mathbf{p}_{k+1} = \mathbf{y}_{k+1} + \beta_k\mathbf{p}_k$	$2n$	$2n$
end for		
Flops per iteration	$(4n + 9)n$	$(4m + 9)n$
Total flops	$(\frac{1}{3}n^2 + 8n + 11)n - 1$	$(m^2 + 8m + 11)n - 1$

Algorithm 12 DICCG method, solving $\mathbf{Ax} = \mathbf{b}$; $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{Z} \in \mathbb{R}^{n\times p}$, $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$

Initial Work

Operation	Flops	
	Full $\mathbf{A}$	Sparse $\mathbf{A}$
Given an initial guess $\mathbf{x}_0$		
$\mathbf{M} = \mathbf{L}_0\mathbf{L}_0^T$	$\frac{n^3}{3}$	nm^2
$\mathbf{V}_Z = \mathbf{AZ}, \in \mathbb{R}^{n\times p}$	$(2n-1)np$	$(2m-1)np$
$\mathbf{E} = \mathbf{Z}^T\mathbf{AZ} = \mathbf{Z}^T\mathbf{V}_Z, \in \mathbb{R}^{p\times p}$	$(2n-1)p^2$	$(2n-1)p^2$
$\mathbf{E}^{-1}$	$\frac{p^3}{3}$	$\frac{p^3}{3}$
$\mathbf{B} = \mathbf{AZE}^{-1} = \mathbf{V}_Z\mathbf{E}^{-1}, \in \mathbb{R}^{n\times p}$	$(2p-1)np$	$(2p-1)np$
Total flops auxiliar matrices	$\frac{n^3+p^3}{3}+$	$(4p^2+m^2+2mp-2p)n+$
	$+(n+2p-1)2np-p^2$	$+\frac{p^3}{3}-p^2$

Initialization of variables

$\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$	$2n^2$	$2nm$
$\hat{\mathbf{r}}_0 = \mathbf{Pr}_0$	$(4n-1)p$	$(4n-1)p$
$\mathbf{y}_0 = \mathbf{M}^{-1}\mathbf{r}_0$	$2n^2$	$2mn$
$\mathbf{r}^0 \cdot \mathbf{y}^0 = (\mathbf{r}^0, \mathbf{y}^0)$	$2n-1$	$2n-1$
$\mathbf{p}_0 = \mathbf{y}_0$		
Initialization of variables	$(2n+1)2n$	$(4m+4p+2)n-$
	$+(4n-1)p-1$	$-p-1$
Total initial work	$\frac{n^3+p^3}{3}+(2n+1)2n$	$(4p^2+m^2+2mp+2p)n+$
	$+(n+2p+1)2np+$	$+(4m+2)n+\frac{p^3}{3}-p^2-$
	$-(p+1)p-1$	$-p-1$

Work per iteration

for $k = 0, ...,$ until convergence		
$\mathbf{w}_k = \mathbf{Ap}_k$	$(2n-1)n$	$(2m-1)n$
$\hat{\mathbf{w}}_k = \mathbf{Pw}_k$	$(4n-1)p$	$(4n-1)p$
$\mathbf{p}_k \cdot \hat{\mathbf{w}}_k = (\mathbf{p}_k, \hat{\mathbf{w}}_k)$	$2n-1$	$2n-1$
$\alpha_k = \frac{\mathbf{r}_k\cdot\mathbf{y}_k}{\mathbf{p}_k\cdot\hat{\mathbf{w}}_k}$	1	1
$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k\mathbf{p}_k$	$2n$	$2n$
$\hat{\mathbf{r}}_{k+1} = \hat{\mathbf{r}}_k - \alpha_k\hat{\mathbf{w}}_k$	$2n$	$2n$
$\mathbf{y}_{k+1} = \mathbf{M}^{-1}\mathbf{r}_{k+1}$	$2n^2$	$2mn$
$\mathbf{r}_{k+1} \cdot \mathbf{y}_{k+1} = (\mathbf{r}_{k+1}, \mathbf{y}_{k+1})$	$2n-1$	$2n-1$
$\beta_k = \frac{\mathbf{r}_{k+1}\cdot\mathbf{y}_{k+1}}{\mathbf{r}_k\cdot\mathbf{y}_k}$	1	1
$\mathbf{p}_{k+1} = \mathbf{y}_{k+1} + \beta_k\mathbf{p}_k$	$2n$	$2n$
end for		
$\mathbf{x} = \mathbf{Qb} + \mathbf{P}^T\hat{\mathbf{x}}_{k+1}$	$8np-2p$	$8np-2p$
Flops per iteration	$(4n+4p+9)n-p$	$(4p+4m+9)n-p$
Total flops	$\frac{n^3+p^3}{3}+(8n+12p+11)n$	$(4p^2+m^2+2mp+14p)n+$
	$+(n+2p+1)2np+$	$+(8m+11)n+\frac{p^3}{3}-p^2-$
	$-(p+4)p-1$	$-4p-1$

Initial	
Methods	**Flops**
PCG	$nm^2 + 4mn + 2n - 1$
DEF1	$((m + 2p + 4)m + (4p + 2)p + 2)n + (\frac{p^2}{3} - p - 1)p - 1$
DEF2	$((m + 2p + 4)m + (4p + 10)p + 1)n + (\frac{p^2}{3} - p - 3)p - 1$
A-DEF1	$((m + 2p + 4)m + (4p + 6)p + 2)n + (\frac{p^2}{3} - p - 2)p - 1$
A-DEF2	$((m + 2p + 4)m + (4p + 14)p + 1)n + (\frac{p^2}{3} - p - 4)p - 1$
BNN	$((m + 2p + 4)m + (4p + 10)p + 2)n + (\frac{p^2}{3} - p - 3)p - 1$
R-BNN1	$((m + 2p + 4)m + (4p + 14)p + 1)n + (\frac{p^2}{3} - p - 4)p - 1$
R-BNN2	$((m + 2p + 4)m + (4p + 10)p + 1)n + (\frac{p^2}{3} - p - 3)p - 1$
ROM	$((3m + 2p + 5)m + (2p + 3)p + 2)n + (\frac{p^2}{3} - p - 1)p - 1$
SROM	$((m + 4p + 4)m + (2p + 6)p + 4)n + (\frac{p^2}{3} + 3p - 4)p + 1$
Iterations	
Methods	**Flops**
PCG	$(4m + 9)n$
DEF1	$(2m + 4p + 9)n - p$
DEF2	$(4m + 4p + 9)n - p$
A-DEF1	$(4m + 8p + 9)n - 2p$
A-DEF2	$(4m + 8p + 9)n - 2p$
BNN	$(9 + 4m + 12p)n - 3p$
R-BNN1	$(8p + 4m + 10)n - 2p$
R-BNN2	$(4m + 4p + 9)n - p$
ROM	$(6m + 4p + 9)n - p$
SROM	$(4m + 8p + 11)n + (4p - 4)p + 2$

Table 6: *Overview table of the flop count of the 2L PCG methods.*

Appendix 5. Eigenvalues properties

In this section, we present some properties of the eigenvalues. Assume $\mathbf{A} \in \mathbb{R}^{n\times n}$ has spectrum $\sigma(\mathbf{A}) = \{\lambda_1, \ldots, \lambda_n\}$.

Lemma .0.1. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ be arbitrary matrices. Now, the following equalities hold:

1. $\sigma(\mathbf{AB}) = \sigma(\mathbf{BA})$,
2. $\sigma(\mathbf{A} + \alpha\mathbf{I}) = \sigma(\mathbf{A}) + \alpha\sigma(\mathbf{I})$, where $\alpha \in \mathbb{R}$,
3. $\sigma(\mathbf{A}) = \sigma(\mathbf{A}^\top)$.

Proof. 1. Let $\mathbf{v}$ be an eigenvector corresponding to eigenvalue λ of matrix $\mathbf{AB}$. Two cases will be considered and proven separately.

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

$$\det(\mathbf{AB}) = \det(\mathbf{BA}) = 0. \tag{1}$$

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

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

$$\begin{aligned} \mathbf{ABv} &= \lambda\mathbf{v} && (2)\\ \mathbf{BABv} &= \lambda\mathbf{Bv} && (3)\\ \mathbf{BAw} &= \lambda\mathbf{w}, && (4) \end{aligned}$$

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

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

2. 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

$$\begin{aligned} (\mathbf{A} + \alpha\mathbf{I})\mathbf{v} &= \lambda\mathbf{v} && (5)\\ \mathbf{A} &= (\lambda - \alpha)\mathbf{v} && (6)\\ \mathbf{A} &= \mu\mathbf{v}, && (7) \end{aligned}$$

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

3. It follows from the definition of determinants that

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \det(\mathbf{A}^\top - \lambda\mathbf{I}) \tag{8}$$

for all λ.

□

Curriculum Vitae

Gabriela Berenice Diaz Cortes was born in Mexico City. Her early academic career started in the National Autonomous University of Mexico, UNAM for its Spanish name. She graduated from the Faculty of Sciences (UNAM) as Physicist in 2011, and from her masters in Material Sciences (*cum laude*) from the Institute of Materials Research (IIM, UNAM) in 2014.

During her bachelor and master thesis, she worked in the Center for Applied Sciences and Technological Development (CCADET) under the supervision of Dr. Tupak Garcia Fernandez. Her work focused on experimental optics. She was teacher assistant in the Laboratory of Electromagnetism from 2010 to 2014.

In 2014 she was awarded a scholarship from the Mexican Institute of Petroleum (IMP) to perform her PhD project at the Technical University of Delft (TU Delft). The same year she moved to the Netherlands to start a PhD in the Department of Applied Mathematics under the supervision of Prof. Cornelis Vuik and Prof. Jan Dirk Jansen from TU Delft, and Dr. Leonid Sheremetov from the IMP. The goal of her PhD project was to combine two *state-of-the-art* methods for the acceleration of the solution of liner systems with iterative methods.

During her PhD she attended several conferences, being the SIAM conference in Geosciences (2017) the most rewarding one. Together with her promotors, she organized a mini-symposium leading to a collaboration with Dr. Damiano Pasetto, from the EPFL University, Switzerland, and Prof. Mario Putti and Dr. Massimiliano Ferronato from the Padova University, Italy. This collaboration resulted in a Masters project, and an article. For this project, she supervised her first master student Jenny Tjang, together with Prof. Vuik.

She was teacher assistant at TU Delft in the practical work of numerical mathematics and partial differential equations. In 2017 she joined the SIAM student chapter as secretary and received the *SIAM certificate of recognition.*

In 2018 she started working at the Mexican Institute of Petroleum (IMP).

Publications and Academic Activities

Papers

Expanded use of a fast photography technique to characterize laser-induced plasma plumes. M. Valverde-Alva, T. García-Fernández, G.B. Díaz Cortés, J.L. Sánchez-Llamazares, E. Rodriguez-Gonzalez, C. Sánchez-Aké, A. Quintana-Nedelcos, and M. Villagrán-Muniz. *Revista mexicana de física*, 60:195-204, 2014.

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

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

On The Acceleration Of Ill-Conditioned Linear Systems: A POD-Based Deflation Method For The Simulation Of Two-Phase Flow. G.B. Diaz Cortes, J.D. Jansen, and C. Vuik. *16th European Conference on the Mathematics of Oil Recovery*, ECMOR XVI, Barcelona, Spain, 2018.

Technical Reports

Physics-based pre-conditioners for large-scale subsurface flow simulation. G.B. Diaz Cortes, C. Vuik and J.D. Jansen. Report 19-03, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2015.

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

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

Conferences

Woudschoten Conference on Scientific Computing, Dutch-Flemish Sicentific Computing Society, Zeist, The Netherlands, 2014 – 2018.

International Conference On Preconditioning Techniques For Scientific And Industrial Applications, Eindhoven University of Technology, The Netherlands, 2014), University of British Columbia, Canada, 2017.
Talk Title (2017): POD-based Deflation Methods for Two-Phase Reservoir Simulation.

European Conference on the Mathematics of Oil Recovery (EAGE), Amsterdam, the Netherlands, 2016, and Spain, 2018.

Talk Title (2016): Physics-based pre-conditioners for large-scale subsurface flow simulation.
Talk Title (2018): On The Acceleration Of Ill-Conditioned Linear Systems: A POD-Based Deflation Method For The Simulation Of Two-Phase Flow.

SIAM Conference on Mathematical and Computational Issues in the Geosciences, University of Erlangen–Nuremberg, Germany, 2017.
Minisymposium organization (title): Krylov-based Deflation Methods in Flow for Porous Media.
Talk Title: POD-deflation method for highly heterogeneous porous media.

15th Copper Mountain Conference On Iterative Methods, Front Range Scientific Computations, Colorado, USA, 2018.
Talk Title: Efficient POD-based deflation methods for the solution of large and ill-conditioned linear systems.

Computational Methods in Water Resources, University of Rennes, Saint-Malo, France, 2018.
Talk Title: Comparison of POD-based iterative solvers for the solution of porous media flow problems.

The 20th European Conference on Mathematics for Industry, ECMI, Budapest, Hungary, 2018.
Talk Title: ROM-based deflation methods to accelerate the solution of time-varying linear systems.

Previous Thesis.

Bachelor in Physics, Facultad de Ciencias, UNAM, Mexico City, Mexico, 2011.
Thesis Title: Laser ablation of NiMnSn and NiMnIn alloys for obtaining thin films. Supervisor: Dr. Tupak Ernesto García Fernández.

Master of Science, Material Sciences, Instituto de Investigaciones en Materiales, UNAM, 2014, Mexico City, Mexico.
Dynamic Speckle for Materials Characterization.
Supervisors: Dr. Tupak Ernesto García Fernández, Dr. Mayo Villagrán Muniz, and Dr. Francisco Manuel Sánchez Arévalo

Teaching experience.

Teacher assistant, Electromagnetism Laboratory. Facultad de Ciencias, UNAM, 2010–2014.

Teacher assistant, Practical work Numerical Mathematics. EEMCS, TU Delft, Delft, The Netherlands, 2014–2017.

Teacher assistant, Practical work Differential Equations. EEMCS, TU Delft, Delft, The Netherlands, 2015–2017.

Supervised works.

Masters Thesis: Study on deflation techniques and POD methods for the acceleration of Krylov subspace methods. J. Tjan, TU Delft, 2018.

Acknowledgements

First and foremost, I want to express my profound gratitude to Prof. Kees for opening this empty book for me, giving me his support through all my PhD, and so many great opportunities. As head of the department, he created a pleasant and friendly environment where it was a pleasure to work, encouraging team building activities like the SIAM student chapter and the Krylov tigers.

I greatly appreciate the support received from my second promotor, Prof. Jan Dirk, who always posed questions that lead us to good research lines, supported my research, and gave me ideas for solving the problems I encountered.

I also want to thank Dr. Leonid, for his supervision from the IMP, and joining the defense committee. Furthermore, I want to thank him and Dr. Erik for their support when starting my new chapter of life: IMP. Besides my supervisors, I would like to thank the rest of the committee members (Dr. D. Pasetto, Dr. M. Ferronato, Dr. H. Hajibeygi, Prof. C.W. Oosterlee and Prof. L.J. Sluys for accepting being part of my committee, reading and commenting on my work.

I would like to express my very great appreciation to the staff of the Numerical Analysis group for their kindness and for creating a friendly atmosphere in the department, Martin, Fred, Valia, Domenco, and Matthias, it was nice meeting you. I want to express my gratitude to Deborah and Kees Lemmens for the assistance provided during my PhD.

I gratefully acknowledge the funding received during my PhD from the Mexican Insitute of Petroleum (IMP), the 'Consejo Nacional de Ciencia y Tecnología (CONACYT)', and the 'Secretaría de Energía (SENER)' through the programs: 'PCTRES' and 'Formación de Recursos Humanos Especializados para el Sector Hidrocarburos'.

I also want to express my gratitude to the DIAM department for the financial support for conferences and courses.

My PhD was a 'there and back again' story with a book in between. It was a tale of high adventure, undertaken in the company of fellow PhDs and friends. Encounters with deadlines, bugs, non-converging solutions, and unexpected behaviors are some of the adventures that I had to befall. But they were overcome by lighter moments: good fellowship, welcome meals, laughter and sports. An unforgettable experience due to the people surrounding me.

Among these people, I want to thank Manuel, Chiara, Dave, Moritz and Reinaldo that welcomed me in Delft and always stayed by my side. Manuel, for taking care of me, making me part of his life, of course for faaaantastic Vancouver, and introducing me to his nice friends Mathias, Lenard, Ina, and Rafael to whom I also want to thank their friendship. Chiara and Dave, I enjoyed a lot the good soccer, and so many great moments together. Moritz, for so funny moments with you, Laura and beautiful

Mara. Reinaldo, for helping me whenever I needed it, and thank you, Slawek and Dominika for the movie nights and dinners.

I want to thank my office mates during my first two years, Jiao, it was really fun meeting you, Yue, Luis and Dr. Xe. I am also very grateful to my office mates from the *coolest office* : Mo, Alice, Merel, Zaza, Peyiao, and Shan Xiue, who made my office hours lighter, and my off-office time great. Mo, I really enjoyed the movie/fondue/swimming nights, thank you also for translating my summary. Alice, it was nice to work so many days and nights together, we managed to survive, in style, a very warm summer. Merel, thank you for your support and for helping me with the Dutch translations. I am particularly indebted with my 20% office mate Roel, who helped me to face all the ill-posed conditions I encountered during my last years, proofread one of my chapters, and shared with me so many nice moments.

This journey took me to unexpected places to present my work in. I want to thank Owen, Vandana, Mousa and Menel, for making the conferences we attended unforgettable. Each of them was particularly nice. Thank you Owen and Mousa for proofreading my chapters, Vandana for celebrating my birthday with me so many times, and Menel, for walking all the way together with me.

I also want to thank captain Baalja for forming the Krylov tigers, for many inspiring and interesting talks, and for helping me with my propositions. Thanks to Berouhz, Anne, Prajakta, Shuaiqiang, Jochen, and Lisa for nice moments in and out office hours. Thanks to my favorite master student, for her great work, good moments, and also for checking the chapter related to her work.

Big thanks to Noelia and Lola, it was super fun hanging out with you; Noelia, we will manage to achieve all our goals someday. Thanks to Baruch, Mario, Oso, Vin, Hrishikesh, Pascal, Jiaxiu, Lorenzo, Lucas and Rock, for so nice memories during the PhD. Si se puede Mario. Thanks a lot to Erika (Pandilli) for her unconditional support and for being always there for me to share my successes and failures.

I also want to thank my awesome Tutor friends: Maaike, Mareen, Samantha Haage, Kristin, Samantha Martinho, Sietske, Ruxi, Jeanette, Radu, Jose and mi amigo para siempre, I will miss the tutor days. Maaike, thank you for proofreading my work and helping me with the Dutch translations. Samantha Haage, it was a pleasure playing, training, and relaxing together with you. Thanks also to the Krylov Tigers who together with Tutor made soccer next level fun.

Mi mayor agradecimiento es a toda mi familia, que siempre esta ahí para mi, se preocupan por mi bienestar y me recibe con los brazos abiertos. En particular a mis padres, mamá, nunca hiciste nada mal y todo lo que he logrado es gracias a tu apoyo y al de mi papá. Y por supuesto a Omar, que siempre me anda ayudando y cumpliendo mis caprichos, aunque a veces no le parezcan. A Diego, Erika, Erik y Freddy por ser una gran familia con cuyo apoyo siempre puedo contar, a Erik por ser tan lindo también. A Ernesto por ayudarme con mi portada, por venir a visitarme y siempre preocuparse por mi.