Pontifícia Universidade Católica
do Rio de Janeiro

Ralph Engel Piazza

Performance Assessment of Linear Solvers for Fully Implicit Reservoir Simulation

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-Graduação em Engenharia Mecânica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Mecânica.

Advisor: Prof. Ivan Fábio Mota de Menezes
Co-advisor: Dr. Daniel Nunes de Miranda Filho

Rio de Janeiro
May 2019

PUC-Rio - Certificação Digital Nº 1612782/CA

Pontifícia Universidade Católica
do Rio de Janeiro

Ralph Engel Piazza

Performance Assessment of Linear Solvers for Fully Implicit Reservoir Simulation

Dissertation presented to the Programa de Pós-Graduação em Engenharia Mecânica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Mecânica. Approved by the Examination Committee.

Prof. Ivan Fábio Mota de Menezes
Advisor
Departamento de Engenharia Mecânica – PUC-Rio

Daniel Nunes de Miranda Filho
Co-advisor
Petróleo Brasileiro S.A.

Prof. Márcio da Silveira Carvalho
Departamento de Engenharia Mecânica – PUC-Rio

Luiz Otávio Schmall dos Santos
Petróleo Brasileiro S.A.

Rio de Janeiro, May 14th, 2019

PUC-Rio - Certificação Digital Nº 1612782/CA

All rights reserved.

Ralph Engel Piazza

The author graduated from Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2009 with a major in Electrical Engineering, with specializations in the areas of Electronics and Decision-Supporting Methods, as well as a minor in Risk Analysis. Concluded a *lato sensu* graduate degree in Petroleum Engineering in 2010, also from PUC-Rio. Joined Petrobras in 2010, as an Electrical Engineer, and in 2011 concluded a *lato sensu* graduate degree in Electric Engineering from Petrobras University. Exchanged positions within Petrobras in 2011, to act as a Petroleum Engineer, and has been dedicated to the area of Reservoir Evaluation since 2012.

PUC-Rio - Certificação Digital Nº 1612782/CA

Bibliographic data

Piazza, Ralph Engel

Performance assessment of linear solvers for fully implicit reservoir simulation / Ralph Engel Piazza ; advisor: Ivan Fábio Mota de Menezes ; co-advisor: Daniel Nunes de Miranda Filho. – 2019.

162 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica, 2019.

Inclui bibliografia

1. Engenharia Mecânica – Teses. 2. Solvers numéricos para sistemas lineares. 3. Métodos iterativos. 4. Precondicionadores. 5. Simuladores de reservatórios. 6. Métodos no subespaço de Krylov. I. Menezes, Ivan Fábio Mota de. II. Miranda Filho, Daniel Nunes de. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. IV. Título.

CDD: 621

PUC-Rio - Certificação Digital Nº 1612782/CA

To my parents, Idelso and Elizabeth,
and to my girlfriend Sarah.

Acknowledgements

I would like to thank everyone that, in different ways, helped me successfully conclude my master's degree in Mechanical Engineering, and that supported me along this strenuous journey.

A special thanks to my parents, for their incredible motivation and assistance during this period, and to my girlfriend Sarah, for being so supportive and understanding of the long hours dedicated to this project.

I am very grateful to my advisors Ivan Menezes and Daniel Miranda, for all their guidance, solicitude and time dedicated to assist my research. Your enthusiasm towards this project and your professionalism have served as inspiration.

My sincere gratitude to Leonardo Duarte, who developed the reservoir simulator with which I worked. This research would not have been possible without your dedication and expertise. All the support with programming matters (and there were many!) and the counsels given throughout the work were truly invaluable.

I am thankful to Pontifical Catholic University of Rio de Janeiro – PUC-Rio, for granting me a scholarship opportunity and making it possible for me to enroll in this master's program. In addition, this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, to whom I also thank.

I would also like to recognize my professors at Pontifical Catholic University of Rio de Janeiro – PUC-Rio, who shared their knowledge and experience so dedicatedly, as well as my classmates, who through various discussions helped enrich this journey. It has been a pleasure from the very first day to undertake this course.

I would like to thank the company for which I work, Petrobras, for allowing me to pursue this opportunity, conceding time for me to be dedicated to it. My earnest gratitude to my manager, Ana Paula Martins, for her unwavering confidence and incentive during these years. I would further like to acknowledge the support offered by my colleagues at my department, who not only encouraged me, but also assisted in several work-related matters so that I could dedicate time to this project.

PUC-Rio - Certificação Digital Nº 1612782/CA

Abstract

Piazza, Ralph Engel; Menezes, Ivan Fábio Mota de (Advisor); Miranda Filho, Daniel Nunes de (Co-advisor). **Performance assessment of linear solvers for fully implicit reservoir simulation.** Rio de Janeiro, 2019. 162p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Petroleum companies investing in the development of hydrocarbon fields rely upon a variety of reservoir studies to perform production forecasts and quantify the risks associated with the economics of their projects. Integral to these studies is the discipline of reservoir modeling, responsible for predicting future reservoir performance under various operational conditions. Considering that the most time-demanding aspect of reservoir simulations is the solution of the systems of equations that arise within the linearization cycles at each time-step, this research focuses on analyzing different numerical *solver* techniques to be applied to a simulator, in order to assess their performance. The numerical *solvers* most suited for the solution of very large systems of equations, such as those encountered in reservoir simulations, are the so-called iterative *solvers*, which gradually approach the solution to a problem by combining an iterative strategy with a preconditioning method. The iterative methods examined in this research were the Stabilized Biconjugate Gradient (BiCGSTAB), the Generalized Minimum Residual (GMRES), and the Orthogonal Minimization (ORTHOMIN) methods. Furthermore, three preconditioning techniques were implemented to aid the iterative methods, namely the Incomplete LU Factorization (ILU), the Nested Factorization (NF), and the Constrained Pressure Residual (CPR) methods. The combination of these different iterative methods and preconditioners enables the appraisal of several distinct *solver* configurations, in terms of their performance in a simulator. The numerical tests conducted in this work made use of a new reservoir simulator currently under development at Pontifical Catholic University of Rio de Janeiro (PUC-Rio), as part of a joint project with Petrobras. The objective of these tests was to assess the robustness and efficiency of each *solver* in the solution of the multiphase flow equations in porous media, and support the selection of the *solver* most suited for the simulator.

PUC-Rio - Certificação Digital Nº 1612782/CA

Keywords

Linear system *solvers*; iterative methods; preconditioners; reservoir simulators; Krylov subspace methods.

PUC-Rio - Certificação Digital Nº 1612782/CA

Resumo

Piazza, Ralph Engel; Menezes, Ivan Fábio Mota de (Orientador); Miranda Filho, Daniel Nunes de (Coorientador). **Avaliação de desempenho de solvers lineares para simuladores de reservatório com formulação totalmente implícita.** Rio de Janeiro, 2019. 162p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Companhias de petróleo investindo no desenvolvimento de campos de hidrocarboneto dependem de estudos de reservatórios para realizarem previsões de produção e quantificarem os riscos associados à economicidade dos projetos. Neste sentido, a área de modelagem de reservatórios é de suma importância, sendo responsável por prever o desempenho futuro do reservatório sob diversas condições operacionais. Considerando que a solução dos sistemas de equações construídos a cada passo de tempo de uma simulação, durante o ciclo de linearização, é a parte que apresenta a maior demanda computacional, esta dissertação foca na análise de diferentes técnicas de *solvers* numéricos que podem ser aplicadas a simuladores, para mensurar seus desempenhos. Os *solvers* numéricos mais adequados para a solução de grandes sistemas de equações, tais como os encontrados em simulações de reservatórios, são os denominados *solvers* iterativos, que gradativamente aproximam a solução de um dado problema por meio da combinação de um método iterativo e um precondicionador. Os métodos iterativos avaliados nesta pesquisa foram o Gradiente Biconjugado Estabilizado (BiCGSTAB), Mínimos Resíduos Generalizado (GMRES) e Minimização Ortogonal (ORTHOMIN). Além disso, três técnicas de precondicionamento foram implementadas para auxiliar os métodos iterativos, sendo estas a Decomposição LU Incompleta (ILU), Fatoração Aninhada (NF) e Pressão Residual Restrita (CPR). A combinação destes diferentes métodos iterativos e precondicionadores permite a avaliação de diversas configurações distintas de *solvers*, em termos de seus desempenhos em um simulador. Os testes numéricos conduzidos neste trabalho utilizaram um novo simulador de reservatórios que está sendo desenvolvido pela Pontifícia Universidade Católica (PUC-Rio) em conjunto com a Petrobras. O objetivo dos testes foi analisar a robustez e eficiência de cada um dos *solvers* quanto à sua capacidade de resolver as

PUC-Rio - Certificação Digital Nº 1612782/CA

equações de escoamento multifásico no meio poroso, visando assim auxiliar na seleção do *solver* mais adequado para o simulador.

PUC-Rio - Certificação Digital Nº 1612782/CA

Palavras-chave

Solvers numéricos para sistemas lineares; métodos iterativos; precondicionadores; simuladores de reservatórios; métodos no subespaço de Krylov.

PUC-Rio - Certificação Digital Nº 1612782/CA

Contents

PUC-Rio - Certificação Digital Nº 1612782/CA

PUC-Rio - Certificação Digital Nº 1612782/CA

List of Figures

PUC-Rio - Certificação Digital Nº 1612782/CA

PUC-Rio - Certificação Digital Nº 1612782/CA

List of Tables

PUC-Rio - Certificação Digital Nº 1612782/CA

List of Algorithms

PUC-Rio - Certificação Digital Nº 1612782/CA

Acronyms

3D – Three Dimensional
ABF – Alternate-Block Factorization
ADIP – Alternating-Direction Implicit Procedure
AIM – Adaptative Implicit Method
AMG – Algebraic Multigrid
BCG – Bi-conjugate Gradient
BICGSTAB – Bi-conjugate Gradient Stabilized
CG – Conjugate Gradient
CGS – Conjugate Gradient Squared
CPR – Constrained Pressure Residual
CS – Correction Scheme
CSC – Compressed Sparse Column
CSR – Compressed Sparse Row
DRS – Dynamic Row Sum
EOS – Equation of State
FGMRES – Flexible Generalized Minimum Residual
FIM – Fully Implicit Method
FMG – Full Multigrid
FVF – Formation Volume Factor
GE – Gaussian Elimination
GM – Geometric Multigrid
GMRES – Generalized Minimum Residual
GS – Gauss-Seidel
IC – Incomplete Cholesky Factorization
ILU – Incomplete LU Factorization
ILUT – Incomplete LU Factorization with Threshold
IMPES – Implicit Pressure Explicit Saturation
IOC – International Oil Company
IPR – Inflow Performance Relationship
MILU – Modified Incomplete LU Factorization
MINRES – Minimum Residual
NF – Nested Factorization
NOC – National Oil Company
OOIP – Original Oil in Place
ORTHOMIN – Orthogonal Minimization
PDE – Partial Differential Equation
PETROBRAS – Petróleo Brasileiro S.A.
PUC – Pontifical Catholic University
PVT – Pressure-Volume-Temperature
QI – Quasi-IMPES
QMR – Quasi-Minimal Residual
SC – Standard Conditions
SCE – Saturation Column Elimination
SIP – Strongly Implicit Procedure
SOR – Successive Overrelaxation
SVD – Single Value Decomposition
TPR – Tubing Performance Relationship

PUC-Rio - Certificação Digital Nº 1612782/CA

Nomenclature

Iterative Method – A component of the numerical *solver* responsible for finding approximate intermediate solutions at each iteration level, measuring the error associated with those solutions, and improving them with the aid of a search vector. Also referred to as an accelerator in the literature.

Model Formulation – The reservoir model used in a simulator is a mathematical model comprised of several equations which attempt to describe the physical processes occurring within the reservoir. These equations are formulated – that is, the model components are put together through appropriate relationships – according to a set of assumptions that involve different aspects of the modeling process. As such, the term formulation may refer to the degree of detail attributed to its fluid components (i.e. black-oil, compositional or thermal formulations), to the degree of detail attributed to its rock components (i.e. dual-porosity, dual-permeability formulations), to the manner with which the differential equations are discretized in time (i.e. fully implicit, IMPES, AIM), as well as others characteristics and combinations thereof (SPE – Reservoir Simulation).

Numerical *Solver* – Relates to the part of the simulator responsible for solving linear or nonlinear systems of equations. May be comprised of one or several algorithms, as well as subroutines within those algorithms. The *solver* receives the coefficients of the system of equations and then returns the appropriate solution after the necessary computations have been performed.

Preconditioner – A component of the numerical *solver* responsible for aiding the iterative method in its search for improved solutions to the problem. It strives to reduce the condition number of the coefficient matrix and, consequently, lead to a more easily obtainable solution. These algorithms may be of such complexity that often in the literature the *solver* is named after this component.

PUC-Rio - Certificação Digital Nº 1612782/CA

1
Introduction

The discipline of reservoir engineering is an integral part in the study of petroleum accumulations. Investigating the capability of a reservoir to deliver hydrocarbon production is of the utmost importance for the petroleum companies[1] investing in a field's development. It will serve as one of the foundations for the economic analysis forecast of the asset and be used to determine whether its explotation is profitable, as well as to assess the risks associated with the necessary investments.

The fundamental objectives of reservoir engineering involve estimating the amount of hydrocarbon volume originally present within the reservoir rocks (OOIP), and the production curve that may be expected to be delivered by the reservoir (Dake, 1978). The investments required to develop and operate a petroleum field are quite high, reaching the order of billions of dollars in certain offshore scenarios. Being capable of accurately predicting future production is vital for defining the optimal number of wells to be constructed, for commissioning production units[2] and flow lines of adequate capacity, and to decide on an acceptable risk premium. Failure to do so may have a significant negative impact on the financial wellbeing of a company.

To maximize the value of the reservoir, it is essential to be capable of optimizing its production, recovering the greatest fraction possible of the hydrocarbon in place. This entails placing wells in advantageous positions, controlling their operating condition intelligently and applying effective enhanced recovery techniques to the field. These activities are related to the proper

PUC-Rio - Certificação Digital Nº 1612782/CA

[1] **Petroleum Companies** – These companies may either be Operators, responsible for leading the exploration and explotation of the field or, eventually, its Partners, who may have a partial stake in the asset.

[2] **Production Units** – Facilities where the produced hydrocarbon undergoes a primary set of treatments and is prepared for *midstream* transport into a refinery or gas treatment plant. In some instances, it may also involve the temporary storage of hydrocarbons for future collection by tanker ships or vehicles.

development and management of the field, and are the concern of the multi-disciplinary team responsible for its study.

1.1 Motivation

There are many tools available for studying a reservoir and attempting to forecast its future production (Ertekin et al., 2001), such as (i) Decline-Curve analysis; (ii) Material-Balance analysis; (iii) Statistical analysis based on analog reservoirs; and (iv) Reservoir simulation. However, of these, only reservoir simulation is truly capable of understanding the fluid movements occurring within the porous media, and of capturing the effects that variations in wells and field operating conditions will have on the production forecast. For instance, it can measure the impact of constructing a new well, of shutting off a producing zone in an existing well, or of starting a water or gas injection campaign. These variations would otherwise not be captured in a realistic fashion by any other method available (Mattax and Dalton, 1990). Furthermore, understanding the paths that the fluids are undertaking in the reservoir permits drilling infill wells in prime positions, in locations where there is still sufficient mobile hydrocarbon left in place to justify the investment of constructing the well.

Albeit being a powerful tool, a simulator is not assured to provide accurate, reliable information in every situation. As per any modeling process, the quality of the output, in this case the simulation results (i.e. fluid production over time), depends strongly on the quality of the model formulated. For a reservoir, this pertains to how closely the flow model simulation mimics the performance of the actual reservoir in question. To be able to adequately represent the reservoir, the model must be able to capture its architecture and the properties of the reservoir rock, including the variations occurring within its domain, as well as the properties of the fluids therein. This is achieved by discretizing the reservoir into small grid blocks and assigning rock and fluid properties to each block, relative to its position in space. It should be apparent that the greater the number of blocks into which the reservoir is partitioned, the more representative the model ought to be. Ultimately, to properly approximate the continuum, the reservoir must be discretized into

PUC-Rio - Certificação Digital Nº 1612782/CA

reasonably fine grids (Stüben, 2007). This has led to flow models with millions, and even tens of millions of grid blocks in modern reservoir studies.

However, as model size expands, the simulation becomes more dispendious, consuming ever more processing time and computer memory. In practice, such computational constraints are what limit model sizes from becoming even larger. The evolution in the number of maximum practicable flow model sizes is depicted in Figure 1.1. It shows that, historically, the tendency is for the model size to double every three years.

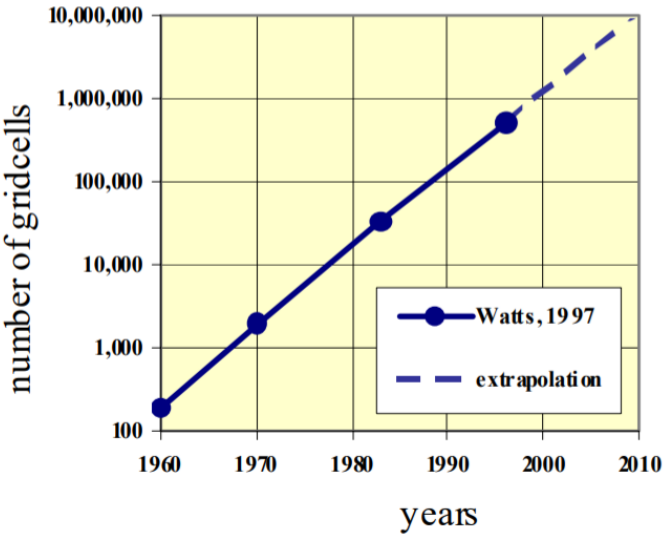

Figure 1.1 – Model size evolution through time (Extracted from Romeu et al., 2005).

To overcome these obstacles, either more powerful computers must be used or, alternatively, the simulator must be improved so as to reduce the memory consumption and processing time required for a simulation run. This second approach is the focus of this research, which will essentially involve the quest for algorithms capable of delivering faster simulation results.

1.2 Objectives

The objective of this thesis is to analyze possible solutions to reduce the simulation time in a fully implicit multiphase reservoir simulator that is being developed at Pontifical Catholic University of Rio de Janeiro (PUC-Rio) for a joint project with Petrobras. More specifically, it aims at implementing different iterative

PUC-Rio - Certificação Digital Nº 1612782/CA

numerical methods and preconditioning algorithms, which together function as a *numerical linear solver*, and examining their performances, with respect to their robustness and efficiency.

The simulator in question is being developed based on a plug-in approach, where each part of the code functions as a black-box and may be called upon or substituted through a control script. The advantages of such configuration have been described by Duarte et al. (2015). Moreover, it uses a state-of-the-art data structure to handle and store the reservoir information, which has the potential to make it very fast. As such, the *solvers* developed were structured as plug-ins, to be incorporated into the simulator architecture.

1.3 Thesis Organization

This thesis is partitioned into six chapters, the first one being this introduction. The second one aims at detailing the fundamental concepts behind a reservoir simulator. Simulators are complex computer programs that utilize mathematical equations to model the physical processes governing the flow of fluids in porous reservoir rocks, and strive to predict the reservoir response under different operational conditions. The equations upon which they are constructed are partial differential equations with second order spatial derivatives and first order time derivatives. To be adequately handled, these equations must first be discretized, for example via finite difference approximations, and then linearized via a nonlinear *solver*. Finally, the resulting equations must be solved via a linear system *solver* (Ertekin et al., 2001). This final part is normally the most time-consuming one, and where significant gains in simulation time are possible, if optimized.

The third chapter describes the main ideas concerning linear systems *solvers* and presents an overview of the methods historically used in reservoir simulation applications. Furthermore, three particular methods are explained in greater detail, ORTHOMIN, GMRES and BiCGSTAB, because they were the algorithms implemented for this research. Besides their partial derivations, the pseudocode of the versions implemented in C++ programming language are also presented (Schildt, 1998; Celes et al., 2004; Goldberg, 1991).

PUC-Rio - Certificação Digital Nº 1612782/CA

The fourth chapter delves into the preconditioners applied in conjunction with the iterative methods of the previous chapter. Those methods are generally unable to solve the large linear systems of equations that arise in the simulator independently, without the aid of preconditioning strategies. The most common options of preconditioners are listed in this chapter. Moreover, three preconditioners that were implemented for this research, ILU, Nested Factorization (NF) and Constrained Pressure Residuals (CPR), are also described in greater detail, with their main concepts explained. Once again, the pseudocodes for some of the implemented versions are provided.

The fifth chapter presents the results obtained with the different simulations that were run during this work. It compares the performance of the various combinations of iterative methods and preconditioners, to examine which *solvers* are the most consistent at delivering faster results, for the synthetic test cases involved.

The final chapter lists the conclusions of the thesis and some final remarks. It also offers suggestions of future works that could be undertaken to further developed this line of research into linear system *solvers* intended for multiphase flow in porous media.

PUC-Rio - Certificação Digital Nº 1612782/CA

1.4 Contribution

The primary contribution of this work is its presentation of a wide range of possible numerical *solvers* applicable to fully implicit multiphase reservoir simulation. It will analyze the most relevant characteristics of the different methods, based on numerical tests, and assess which *solver* configurations seem most suited for a reservoir simulator.

2
Fundamentals of Reservoir Simulation

The intention of this chapter is to provide an overview of the mathematical modeling behind a reservoir simulator and contextualize the motivation for studying numerical methods directed towards solving linear system of equations, such as the ones encountered in the simulator. The approach taken to analyze the problem, the organization of the topics and the deduction of the equations presented here are all based upon the content of Ertekin et al.'s Basic Applied Reservoir Simulation (2001). Complementary references are also listed where appropriate.

The primary objective of a reservoir simulator is to be capable of accurately predicting reservoir response under various field development scenarios, such as different well placement configurations, well operating conditions and hydrocarbon recovery techniques. For this purpose, it should be designed to be as efficient as possible in delivering results that are sufficiently precise to be of practical use to the reservoir engineer. This accuracy is pursued by emulating as realistically as possible the underlying physics behind the flow of fluids through porous media, while maintaining the computational effort required to perform the reservoir study at a reasonable level in terms of processing power, computer memory and simulation time duration.

The fundamental physical principles applied to reservoir simulation are (i) generalized mass conservation; (ii) a governing law of fluid flow through porous media – Darcy's Law; and (iii) reservoir fluid and rock properties. These principles are then combined to form multiphase flow equations, which are a set of partial differential equations (PDEs). In addition to the multiphase flow equations, complementary equations, such as those expressing relationships between multiple phases, may also need to be considered, so as to constitute a comprehensive multiphase flow model.

The resulting mathematical formulation has the potential to describe fluid pressure and saturation, as well as production and injection rates, the main variables of a reservoir study, everywhere in the reservoir, at any time period. However, such

PUC-Rio - Certificação Digital Nº 1612782/CA

exact, analytical results are not feasibly achieved in practice, due to the nonlinear nature of the equations and to the inherent heterogeneities of the rock and fluid properties. Therefore, to solve the equations while still preserving a representative description of the reservoir, it becomes necessary to apply a numerical method to the problem at hand through the discretization of the multiphase flow equations, so as to transform the PDEs into algebraic form. These algebraic equations are, nevertheless, still nonlinear equations, whose solutions remain very challenging to obtain. To overcome this issue, a linearization technique must be applied to the equations, resulting in a set of linearized discretized multiphase flow equations. These linear algebraic equations are now able to provide values of pressure and saturation only at discrete moments of time and locations in the reservoir, delivering only an approximate overview of the reservoir behavior. However, depending on how well the reservoir parameters are known and on the degree of discretization applied to the reservoir, their results will have honored the physics of the problem to a degree not attainable by traditional analytical solutions.

PUC-Rio - Certificação Digital Nº 1612782/CA

2.1
Physical Principles

General Mass Conservation Equation

The principle of conservation of mass states that the total mass of fluid entering a control volume must equal the sum of the mass leaving the control volume and the mass accumulated within the volume. The partial differential equation that describes the conservation of mass over a control volume element through which fluid flows is named the continuity equation. It can be expressed individually for each fluid phase or fluid component as

$$-\frac{\partial}{\partial x}(\dot{m}_x A_x)\cdot\Delta x-\frac{\partial}{\partial y}(\dot{m}_y A_y)\cdot\Delta y-\frac{\partial}{\partial z}(\dot{m}_z A_z)\cdot\Delta z = \forall_B\cdot\frac{\partial}{\partial t}(m_\forall)-q_m-q_{m_t} \tag{2.1}$$

where $\dot{m}$ is the mass flux in each direction; A is the surface area perpendicular to the flux; $m_\forall$ is the mass per unit volume of porous medium; $\forall_B$ is the bulk volume

of the control element; q_m is the net rate of mass accumulation through a source term; and q_{m_t} is the net rate of mass transfer between phases. Here and in every other equation presented, a consistent set of units is assumed; otherwise, unit conversion constants would have to be incorporated into the equations.

For the oil, water and free gas phases we may define the following additional equations

$$\dot{m}_c = \rho_c u_c \tag{2.2}$$

$$m_{\forall_c} = \phi \rho_c S_c \tag{2.3}$$

$$q_{m_c} = \rho_c q_c \tag{2.4}$$

where c represents the fluid phase under analysis (or fluid component); u_c is the phase velocity; ρ_c is the phase density; S_c is the phase saturation in the porous medium; and ϕ is the total porosity of the control volume.

For the dissolved gas present in the oil phase, the following definitions are applicable

$$\dot{m} = \left(\rho_{sc} \frac{R_s}{B_o}\right) \cdot u_o \tag{2.5}$$

$$m_{\forall} = \left(\rho_{sc} \frac{R_s}{B_o}\right) \cdot S_o \tag{2.6}$$

$$q_m = \left(\rho_{sc} \frac{R_s}{B_o}\right) \cdot q_o \tag{2.7}$$

where ρ_{sc} is the density of the oil at standard conditions (SC), referring to the standard pressure $(1\ \text{atm})$ and temperature $(20°\text{C})$ levels; R_s is the solution gas-oil ratio; and B_o is the formation volume factor (FVF) of the oil, which shall be detailed further ahead. It is assumed here that mass transfer occurs only between the oil and gas phases, but not between the hydrocarbon and water phases.

PUC-Rio - Certificação Digital Nº 1612782/CA

Darcy's Law

The constitutive equation that describes fluid flow through the control element is denoted as Darcy's law. This equation was developed empirically from experimental results and relates the areal velocity of a fluid flowing in a porous medium to a set of fluid and rock parameters. The energy available to a fluid for generating movement may be represented by its potential gradient, defined as

$$\overline{\nabla}\Phi = \overline{\nabla}P - \gamma\overline{\nabla}Z \tag{2.8}$$

where Φ represents the fluid potential; P is the fluid pressure; γ is the specific weight of the fluid; and Z is the vertical distance to a reference datum.

The flow stemming from a potential gradient within a fluid can then be expressed, for any given direction x, as

$$u_x = -\frac{k_x}{\mu} \cdot \frac{\partial \Phi}{\partial x} \tag{2.9}$$

where k_x is the effective permeability in that direction; and μ is the fluid viscosity.

Reservoir Fluid and Rock Properties

The relationship between fluid density and other thermodynamic state variables, such as temperature, pressure and internal energy, may be expressed by certain mathematical formulations, commonly called the equation of state (EOS) of the fluid (Ertekin et al., 2001; Dake, 1978). For example, the density of water may be given by

$$\rho_w = \rho_{wsc}[1 + c_w(P_w - P_{sc}) - c_{Tw}(T_w - T_{sc})] \tag{2.10}$$

where c_w is the water compressibility; P_{sc} and T_{sc} are the standard pressure and temperature values, respectively; c_{Tw} is the coefficient of thermal expansion of water; and T_w is the water temperature. The compressibility of a fluid is defined as

PUC-Rio - Certificação Digital Nº 1612782/CA

$$c_c = -\frac{1}{\forall_c}\frac{\partial \forall_c}{\partial P} = \frac{1}{\rho_c}\frac{\partial \rho_c}{\partial P} \tag{2.11}$$

where $\forall_c$ is the volume of an arbitrary quantity of the component's mass, measured at the same pressure level in which the $(\partial \forall_c / \partial P)$ derivative is calculated.

Likewise, the oil density may be expressed as

$$\rho_o = \frac{\left(\rho_{osc} + \rho_{gsc} R_s\right)}{B_o} \tag{2.12}$$

for saturated oil, whenever $P_o < P_b$; and as

$$\rho_o = \rho_{ob}[1 + c_o(P_o - P_b)] \tag{2.13}$$

for undersaturated oil, whenever $P_o > P_b$. Here, P_b is called the saturation or bubble-point pressure and ρ_{ob} is the density of the oil phase measured at that specific pressure level.

The free gas density can be derived from the real-gas law as being equal to

$$\rho_g = \frac{P_g M}{ZRT} \tag{2.14}$$

where M is the molar mass of the gas; Z is the real-gas compressibility factor, calculated from the pseudoreduced pressure and pseudoreduced temperature of the gas; and R is the universal gas constant.

Another important relationship describing a fluid is the ratio of volume it occupies at different pressure levels, defined as its formation volume factor

$$B_c = \frac{\forall_c}{\forall_{csc}} \tag{2.15}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

where $\forall_c$ is the volume occupied by an arbitrary quantity of mass of the component at in-situ conditions; and $\forall_{csc}$ is the volume occupied by this same amount of mass at standard conditions.

For water and free gas phases, this may be expressed more conveniently as

$$B_c = \frac{\rho_{sc}}{\rho_c} \tag{2.16}$$

while for an undersaturated oil phase as

$$B_o = B_{ob}[1 - c_o(P_o - P_b)] \tag{2.17}$$

where B_{ob} is the FVF at the bubble-point pressure.

Concerning the gas phase, an additional relationship is necessary to determine the amount of gas dissolved within the oil phase. This quantity is defined as the solution gas-oil ratio

$$R_s = \frac{\forall_{SG}}{\forall_o} \tag{2.18}$$

where $\forall_{SG}$ represents the volume of dissolved gas at standard conditions, per unit volume of oil at in-situ conditions; and $\forall_o$ is the volume of oil at standard conditions, per unit volume of oil at in-situ conditions. Since at pressures above the saturation point all of the solution gas is already contained within the oil phase, for pressures in this range the value of R_s remains constant and equal to its value at bubble-point pressure.

Moreover, the viscosities of the phases are also required for describing the flow processes in the reservoir. For an undersaturated oil, the equation to be used is

$$\mu_o = \mu_{ob}/\left[1 - c_\mu(P_o - P_b)\right] \tag{2.19}$$

where μ_{ob} is the oil viscosity at bubble-point pressure; and c_μ represents the fractional change of viscosity per unit change of pressure.

PUC-Rio - Certificação Digital Nº 1612782/CA

Finally, during the simulation, some of the necessary fluid properties are not calculated directly from mathematical formulas, but instead have its values at a given pressure interpolated from measurements taken at some other pre-established pressure levels, in what constitutes a PVT (Pressure-Volume-Temperature) table. These tabled values are determined via laboratory testing of fluid samples collected from the reservoir, or occasionally via correlation models, if representative samples are not available. This is the case of μ_w and μ_g, which are usually established from correlations, and of B_o, R_s and μ_o at pressures below the saturation point, which are usually established from experimental results.

It is also worth noting that the saturation pressure may not be constant for the entire reservoir, nor stay constant for the entire simulation period. This will depend on whether fluid compositions are equivalent in different regions and at different depths of the reservoir, as well as on whether there is gas injection into the reservoir (Ertekin et al., 2001; and Ponting et al., 1983).

In addition, in a black-oil model, such as the one considered for this work, only the three aforementioned fluid components will potentially be present inside the reservoir: oil, water and gas. However, in compositional models, several fluid components might be contemplated, and thus EOS PVT characterizations would be required to describe their behavior.

With respect to the rock properties, porosity may also be considered to be a function of pressure, according to the following relationship

$$\phi = \phi_{REF}\left[1 + c_\phi (P - P_{REF})\right] \tag{2.20}$$

where ϕ_{REF} is the porosity at a reference pressure level; P_{REF} is the reference pressure; and $c_\phi$is the compressibility of the rock, whose definition is equivalent to the one given for fluids. Also, the rock is considered here to be only a slightly compressible media, and thus its compressibility value is assumed constant.

Permeability, on the other hand, is usually assumed to be independent of any of the simulation variables. In theory, its value might depend on porosity, but for most practical applications this relation is neglected.

Porosity and permeability can be considered to be either homogeneous or heterogeneous in a reservoir, that is, to have constant or varying values,

PUC-Rio - Certificação Digital Nº 1612782/CA

respectively, throughout its domain. Additionally, permeability may also exhibit directional variations in a given point in space. This characteristic, called anisotropy, means that its value may differ for the x, y and z spatial directions. For simplicity, in the following derivations the coordinate system is assumed to be aligned with the principal axes of the permeability tensor, resulting in a diagonal tensor, without cross terms.

Furthermore, the absolute permeability of a rock represents the easiness with which single-phase fluid is capable of flowing through its pore network. In the case of multiphase flow, the effective permeability of the rock to the flow of each phase will be a fraction of the single-phase one, and is expressed, for a given direction, as being

$$k_{cx} = k_{rc} k_x \tag{2.21}$$

where k_x is the absolute permeability in that direction; k_{rc} is the relative permeability to the fluid component c; and k_{cx} is the effective permeability in the given direction.

Similar to the fluid properties, the relative permeability of a phase may be determined via laboratory testing using core or side-well core samples, or alternatively via correlation models, if such samples are not available. However, unlike the other properties discussed thus far, relative permeabilities are usually dependent on the fluid's saturation level, instead of its pressure and temperature. For three-phase flow, the most comprehensive models available are either those developed by Stone, or the one developed by Naar, Henderson and Wygal (Ertekin et al., 2001; and Chen et al., 2006).

2.2
Multiphase Flow Equations

Combining the various mathematical equations modeling these physical principles, and defining the total gas production rate as

$$q_{GSC} = q_{FGSC} + R_s q_{OSC} \tag{2.22}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

where is q_{GSC} the total gas production rate; and q_{FGSC} is the gas production rate pertaining just to the free gas inside the reservoir; it is possible to arrive in simultaneous phase equations that govern the flow of each individual component through a control element. For a black-oil model, the respective oil, water and gas equations are

$$\begin{aligned}\frac{\partial}{\partial x}\left[k_x A_x \cdot \frac{k_{ro}}{\mu_o B_o} \cdot \left(\frac{\partial P_o}{\partial x} - \gamma_o \frac{\partial Z}{\partial x}\right)\right] \cdot \Delta x \\ + \frac{\partial}{\partial y}\left[k_y A_y \cdot \frac{k_{ro}}{\mu_o B_o} \cdot \left(\frac{\partial P_o}{\partial y} - \gamma_o \frac{\partial Z}{\partial y}\right)\right] \cdot \Delta y \\ + \frac{\partial}{\partial z}\left[k_z A_z \cdot \frac{k_{ro}}{\mu_o B_o} \cdot \left(\frac{\partial P_o}{\partial z} - \gamma_o \frac{\partial Z}{\partial z}\right)\right] \cdot \Delta z \\ = \forall_B \cdot \frac{\partial}{\partial t}\left(\frac{\phi S_o}{B_o}\right) - q_{OSC}\end{aligned} \tag{2.23}$$

$$\begin{aligned}\frac{\partial}{\partial x}\left[k_x A_x \cdot \frac{k_{rw}}{\mu_w B_w} \cdot \left(\frac{\partial P_w}{\partial x} - \gamma_w \frac{\partial Z}{\partial x}\right)\right] \cdot \Delta x \\ + \frac{\partial}{\partial y}\left[k_y A_y \cdot \frac{k_{rw}}{\mu_w B_w} \cdot \left(\frac{\partial P_w}{\partial y} - \gamma_w \frac{\partial Z}{\partial y}\right)\right] \cdot \Delta y \\ + \frac{\partial}{\partial z}\left[k_z A_z \cdot \frac{k_{rw}}{\mu_w B_w} \cdot \left(\frac{\partial P_w}{\partial z} - \gamma_w \frac{\partial Z}{\partial z}\right)\right] \cdot \Delta z \\ = \forall_B \cdot \frac{\partial}{\partial t}\left(\frac{\phi S_w}{B_w}\right) - q_{WSC}\end{aligned} \tag{2.24}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\frac{\partial}{\partial x}\left[k_x A_x \cdot \frac{k_{rg}}{\mu_g B_g} \cdot \left(\frac{\partial P_g}{\partial x} - \gamma_g \frac{\partial Z}{\partial x}\right) + k_x A_x \cdot \frac{k_{ro} R_s}{\mu_o B_o} \cdot \left(\frac{\partial P_o}{\partial x} - \gamma_o \frac{\partial Z}{\partial x}\right)\right] \cdot \Delta x + \frac{\partial}{\partial y}\left[k_y A_y \cdot \frac{k_{rg}}{\mu_g B_g} \cdot \left(\frac{\partial P_g}{\partial y} - \gamma_g \frac{\partial Z}{\partial y}\right) + k_y A_y \cdot \frac{k_{ro} R_s}{\mu_o B_o} \cdot \left(\frac{\partial P_o}{\partial y} - \gamma_o \frac{\partial Z}{\partial y}\right)\right] \cdot \Delta y + \frac{\partial}{\partial z}\left[k_z A_z \cdot \frac{k_{rg}}{\mu_g B_g} \cdot \left(\frac{\partial P_g}{\partial z} - \gamma_g \frac{\partial Z}{\partial z}\right) + k_z A_z \cdot \frac{k_{ro} R_s}{\mu_o B_o} \cdot \left(\frac{\partial P_o}{\partial z} - \gamma_o \frac{\partial Z}{\partial z}\right)\right] \cdot \Delta z = \forall_B \cdot \frac{\partial}{\partial t}\left(\frac{\phi S_g}{B_g} + \frac{\phi R_s S_o}{B_o}\right) - q_{GSC} \quad (2.25)$$

where Δx, Δy, and Δz are the geometric dimensions of the control volume, considering a three-dimensional (3D) problem in the cartesian coordinate system.

The multiphase flow equations, as presented, attempt to encompass all of the noteworthy forces acting on the fluid; namely, viscous, capillary and gravitational forces. They also allow for the treatment of irregular reservoir boundaries and boundary conditions, as well as reservoir heterogeneities.

Furthermore, since the flow equations are based upon the mass conservation of each component separately and since they are referenced to surface conditions, part of the mass accounted for in the gas rate term was originally dissolved within other phases, while part was already in a free state inside the reservoir. Assuming that the water and gas phases are practically immiscible, the solution gas may be considered to originate exclusively from the oil phase. Therefore, in the case of gas mass conservation it is necessary to include the flow of this dissolved gas through the control element; this is the reason for introducing the terms related to oil flow into the gas equation.

Finally, the formulation presented in Equations (2.23) – (2.25) assumes that no chemical reactions are occurring between the rock and the reservoir fluids; nor between the different fluid components. It also considers that no physical dispersion

PUC-Rio - Certificação Digital Nº 1612782/CA

is occurring in the case of miscible recovery schemes, and presumes instantaneous local equilibrium.

2.3
Multiphase Flow Model

The mathematical formulation to be derived in the end will depend not only on the previous multiphase flow equations, but also on some additional relationships that correlate the variables of the problem, as well as on the initial and boundary conditions. This supplementary information is necessary for the complete description of the fluid flow inside the reservoir.

Additional Relationships

The first such relationship to be established is the relation between phases. For a black-oil model, the oil and water phases are deemed to be immiscible, as are the water and gas phases. Gas is considered to be miscible with oil, and can thus exist as free gas or as solution gas. Moreover, it is assumed that oil, water and gas are the only fluids present in the pore space, and that consequently they must fully occupy it

$$S_o + S_w + S_g = 1 \tag{2.26}$$

An alternative treatment may be applied to regions of the reservoir in which there is no free gas (Chen et al., 2006). In this case, the volume balance relationship could equate to

$$S_o + S_w = 1 \tag{2.27}$$

and the gas phase flow equation could be altered to a bubble-point equation, where P_b would be the primary variable. The transition from an undersaturated condition to a saturated condition inside any given control element would then occur whenever

PUC-Rio - Certificação Digital Nº 1612782/CA

$$P_b > P_o \tag{2.28}$$

while the transition in the opposite direction, from a saturated condition to an undersaturated one, would be given by the occurrence of

$$S_g < 0 \tag{2.29}$$

during the simulation. Nonetheless, this formulation with P_b as one of the problem variables will not be detailed further, with the derivation of the equations being restrained to the oil pressure and water and gas saturation format.

Moreover, in a compositional model, the sum of the saturation of the various components present would also be qual to unity, and the behavior of each component would be described by their own flow equation.

Also, in the black-oil model the fluid temperature is assumed to be constant throughout the reservoir, at all times. Conversely, in thermal models the temperature must also be considered and, therefore, an additional energy-balance equation will arise.

Yet additional models are available to the reservoir engineer, such as dual-porosity and dual-permeability models, that attempt to enrich the quality of the simulation, but hereafter only the standard three-phase black-oil model will be considered.

Finally, a relationship that must still be established is the one between the different phase pressures (Ertekin et al., 2001; and Aziz and Settari, 1979). These pressures are related through the capillary forces present in the small pore spaces, as follows

$$P_{cow} = P_o - P_w = f(S_w) \tag{2.30}$$

$$P_{cgo} = P_g - P_o = f(S_g) \tag{2.31}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

where P_{cow} is the oil-water capillary pressure, whose value is a function of the water saturation; and P_{cgo} is the gas-oil capillary pressure, whose value is a function of the gas saturation.

Equations (2.26) and (2.30) – (2.31) can then be used so as to eliminate the terms S_o, P_w and P_g from the multiphase flow equations (2.23) – (2.25), leaving only P_o, S_w and S_g as the variables of the simulation. Consequently, the problem may now be described by three equations with three unknowns.

Initial Conditions

The initial conditions required for the black-oil model vary for each zone in the reservoir. In an original gas-cap zone only gas is present as a free fluid, and connate water saturation is at its initial value S_{wi}. The two other variables can then be expressed as

$$S_g = 1 - S_{or} + S_{wi} \tag{2.32}$$

where S_{or} represents a residual oil saturation that may eventually be present; and

$$P_o = P_g - P_{cgo}(S_g) \tag{2.33}$$

where gas pressure P_g at each point may be calculated from its hydrostatic gradient

$$\frac{\partial P_g}{\partial z} = \gamma_g \tag{2.34}$$

In the oil-gas transition zone it is first necessary to calculate the oil and gas pressures separately, using their respective gradients; gas as presented in Equation (2.34) and oil as in Equation (2.35)

$$\frac{\partial P_o}{\partial z} = \gamma_o \tag{2.35}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

From the pressures levels it is possible to obtain the gas saturation from the capillary pressure relationship

$$P_{cgo}(S_g) = P_g - P_o \tag{2.36}$$

while the water saturation will be at its initial value S_{wi}.

In the oil zone, there is no free gas present and so the gas saturation S_g is null, while water saturation will be at its initial value S_{wi}. Since oil is a continuous phase, its pressure P_o may be calculated directly from the gradient, as was done previously.

In the oil-water transition zone the procedure is analogous to the oil-gas zone. Oil and water pressures are calculated through their respective gradients, which for water is

$$\frac{\partial P_w}{\partial z} = \gamma_w \tag{2.37}$$

Water saturation is then obtained from the capillary pressure relationship

$$P_{cow}(S_w) = P_o - P_w \tag{2.38}$$

while gas saturation S_g is assumed to be null.

In the water zone, water is the only phase present. Consequently, oil and gas saturations are null and water saturation equals unity. Water pressure will again be given by the hydrostatic gradient, while oil pressure may be found using the capillary pressure equation

$$P_o = P_w - P_{cow}(S_w) \tag{2.39}$$

Finally, in addition to the specific gravity of each fluid, the capillary pressure curves, and the initial water saturation distribution above the water zone, it is also necessary to establish a reference oil pressure at a reference datum. This will be used for calculating the different phase pressures throughout the reservoir interval, via the fluid gradients. In an undersaturated reservoir the reference datum may be

PUC-Rio - Certificação Digital Nº 1612782/CA

any arbitrary depth within the reservoir, but in the saturated case it should be taken to be the depth of the oil-gas contact Z_{ogc}.

Boundary Conditions

For the external boundary conditions three situations may arise, either a no-flow condition, a constant-pressure condition, or a specified-flux condition (Ertekin et al., 2001; Chen et al., 2006).

A no-flow condition, representing a sealed boundary, is modeled by simply defining the transmissibilities across the outer surface of the reservoir as having zero value. Transmissibility refers to the term that relates the fluid potential gradient to a corresponding fluid flow in a given direction, and is defined as

$$T_{cx} = \left(\frac{k_x A_x}{\Delta x}\right) \cdot \left(\frac{k_{rc}}{\mu_c B_c}\right) \tag{2.40}$$

The constant-pressure condition, with pressure P_E, represents a scenario in which the rate of fluids withdrawn from one side of the boundary equals the rate of fluids supplied to the opposite side. It can be modeled by an extra influx term present at that outer control element, as if it were a well

$$q_{CSC-B} = \left[\left(\frac{k_x A_x}{\Delta x/2}\right)\left(\frac{k_{ro}}{\mu_o B_o}\right)\right] \cdot \left(P_E - P_{c_n}\right) \tag{2.41}$$

where P_{c_n} is the pressure at the center of a control element n with a no-flow boundary in the x direction; and q_{CSC-B} is the pseudo-rate of component c across the external boundary, given in surface conditions.

The specified-flux condition occurs whenever there exists communication between the reservoir and an adjacent permeable rock body, such as an aquifer or another separate reservoir. The flux may occur inwards towards the reservoir, or outwards to the external body, depending on the potential gradient at the boundary. Moreover, the magnitude of the influx or efflux may be defined either by a specified flux rate or by a specified pressure gradient. In both cases this is represented

PUC-Rio - Certificação Digital Nº 1612782/CA

mathematically by setting the outer transmissibilities to zero and including an extra rate term inside the boundary control element.

For the specified rate q_{sp} situation, the term to be added to the equation of each component c is

$$q_{CSC-B} = \frac{T_{cx}}{\sum_c^N T_{cx}} \cdot q_{sp} \tag{2.42}$$

where the transmissibility T_{cx} of each component refers to the one on the boundary side in the x direction.

Alternatively, for the specified pressure gradient dP/dx situation, the term to be added to the equation of each component c becomes

$$q_{CSC-B} = (-T_{cx}\,\Delta x) \cdot \frac{dP}{dx} \tag{2.43}$$

The internal boundary conditions of the problem depend on the specifications defined for the various wells. There are several manners of specifying the operating condition of each well, especially in a multiphase reservoir with fluid production or injection occurring in multiple control elements, that is, with a well completed and perforated in multiple layers. The possible specifications may be wellbore bottom-hole pressure; oil, water, liquid, gas or total bottom-hole rate; oil, water, liquid, gas or total surface rate.

The relationship between the fluid pressure in a control element, the wellbore bottom-hole pressure and a component's surface rate q_{CSC_n} is designated as the inflow performance relationship (IPR). For a control element n in a multilayered vertical well, this relationship is defined as

$$q_{CSC_n} = -J_n \cdot \left(P_n - P_{wf_n}\right) \tag{2.44}$$

where P_{wf_n} is the flowing wellbore bottom-hole pressure, which here may be considered equal for all components; and J_n is the component productivity or injectivity index, given by

PUC-Rio - Certificação Digital Nº 1612782/CA

$$J_n = \frac{2\pi k_{H_n} k_{rc} h_n}{\mu_{c_n} B_{c_n} \cdot \left[\ln\left(r_{eq_n}/r_w\right) + s_n\right]} = \left(\frac{k_{rc}}{\mu_c B_c}\right)_n \cdot G_{w_n} \tag{2.45}$$

where G_{w_n} represents the constant terms of the equation; k_{H_n} is the equivalent permeability in the radial direction; k_{rc} is the relative permeability for the component c; h_n is the height of the control element; s_n is the formation damage, or wellbore skin; r_w is the well radius; and r_{eq_n} is the equivalent radius, defined by Peaceman (1978) as

$$r_{eq} = 0.28 \cdot \frac{\left\{\left[\left(\frac{k_y}{k_x}\right)^{1/2} \cdot (\Delta x)^2\right] + \left[\left(\frac{k_x}{k_y}\right)^{1/2} \cdot (\Delta y)^2\right]\right\}^{1/2}}{\left(\frac{k_y}{k_x}\right)^{1/4} + \left(\frac{k_x}{k_y}\right)^{1/4}} \tag{2.46}$$

which represents the position within the control element where the calculated pressure of the element will be equal to the actual flowing pressure.

As a note, in a black-oil model with gas production, the gas inflow performance relationship is altered slightly, to include the solution gas into the equation

$$q_{GSC_n} = -G_{w_n}\left[\left(\frac{k_{rg}}{\mu_g B_g}\right) \cdot \left(P_{g_n} - P_{wf_n}\right) + \left(\frac{k_{ro} R_s}{\mu_o B_o}\right) \cdot \left(P_{o_n} - P_{wf_n}\right)\right] \tag{2.47}$$

The flowing bottom-hole pressure at the depth of each control element situated within the completed zone may be approximated by a reference pressure and the average hydrostatic pressure gradient, meanwhile ignoring the frictional pressure drop that occurs within the well, between the multiple completion layers.

$$P_{wf_n} = P_{wf_{REF}} + \bar{\gamma}_w \cdot (Z_n - Z_0) \tag{2.48}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

where $P_{wf_{REF}}$ is the pressure at a reference depth Z_0; Z_n is the depth of the control element; and $\bar{\gamma}_w$ is the average pressure gradient in the well, estimated as

$$\bar{\gamma}_w = \frac{B_o \gamma_o q_{OSC} + B_w \gamma_w q_{WSC} + B_g \gamma_g q_{FGSC}}{B_o q_{OSC} + B_w q_{WSC} + B_g q_{FGSC}} \tag{2.49}$$

2.4
Discretization of the Flow Equations

There exist several methods available for discretizing a partial differential equation, such as finite differences, finite elements and finite volumes. Traditionally, the finite difference technique is the one most employed in the field of reservoir simulation. Albeit one of the oldest methods developed, it is still the most dominant choice of numerical treatment for partial differential equations (Grossmann et al., 2007). Each one of the multiphase equations of the reservoir model contain two types of derivatives that need to be discretized: second-order spatial derivatives and first-order time derivatives. The spatial derivatives on the left-hand side of the equations are of the form

$$\frac{\partial}{\partial x}\left(\xi_x \frac{\partial \eta}{\partial x}\right)_{i,j,k} \Delta x_{i,j,k} = \left(\frac{\partial f}{\partial x}\right)_{i,j,k} \Delta x_{i,j,k} \tag{2.50}$$

where ξ_x represents the transmissibility in the x direction, times Δx; η represents the potential gradient of the phase; and the indices i, j, k represent the numerical ordering of the control element in a given coordinate system.

This ordering of the elements, henceforth called grid blocks or cell blocks, may be done in different fashions. Some common choices are (i) Natural ordering, in which cells are numbered sequentially by rows or columns; (ii) D-4 ordering, in which cells are numbered sequentially by diagonals; and (iii) Red-Black or A-3 ordering, in which cells are numbered sequentially by rows or columns but skipping every other grid block; among others. The manner chosen to order the grid will impact directly on the matrix structure that will arise in the linear system *solver* part of the simulator, as shall be discussed subsequently (Ertekin et al., 2001; Behie et al., 1984; and Price and Coats, 1974).

PUC-Rio - Certificação Digital Nº 1612782/CA

For grids constructed using a block-centered system, as opposed to those employing a point-distributed system, the problem unknowns represent the value of the different variables (P_o, S_w and S_g) at the center of each cell. Also, in this system, the transmissibility coefficients represent the connections between cells, as illustrated in Figure 2.1 (Ertekin et al., 2001; and Peaceman, 1977).

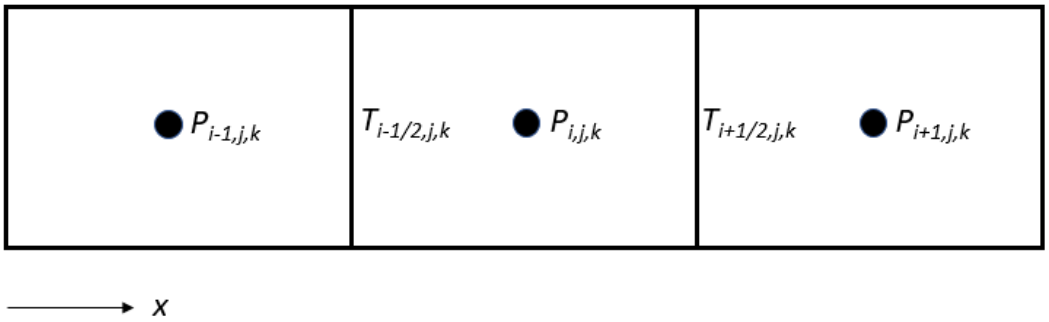

Figure 2.1 – Schematic of a block-centered grid (Adapted from Ertekin et al., 2001).

PUC-Rio - Certificação Digital Nº 1612782/CA

Returning to the spatial derivative form presented previously, the terms may be approximated by applying central-difference discretization to the derivatives (Ertekin et al., 2001; Chen et al., 2006), resulting in

$$\left(\frac{\xi_x}{\Delta x}\right)_{i-\frac{1}{2},j,k} \cdot \left(\eta_{i-1,j,k} - \eta_{i,j,k}\right) + \left(\frac{\xi_x}{\Delta x}\right)_{i+\frac{1}{2},j,k} \cdot \left(\eta_{i+1,j,k} - \eta_{i,j,k}\right) = \left(T_{cx}\right)_{i-\frac{1}{2},j,k} \cdot \left(\eta_{i+1,j,k} - \eta_{i,j,k}\right) - \left(T_{cx}\right)_{i+\frac{1}{2},j,k} \cdot \left(\eta_{i,j,k} - \eta_{i-1,j,k}\right) \tag{2.51}$$

Due to the size of the equations appearing in the formulation process, it is worthwhile to introduce here the following simplifying notation

$$\Delta_x(T_{cx}\Delta_x\eta)_{i,j,k} = \left(T_{cx}\right)_{i-\frac{1}{2},j,k} \cdot \left(\eta_{i+1,j,k} - \eta_{i,j,k}\right) - \left(T_{cx}\right)_{i+\frac{1}{2},j,k} \cdot \left(\eta_{i,j,k} - \eta_{i-1,j,k}\right) \tag{2.52}$$

where is Δ_x is a finite-difference operator on the space domain.

Analyzing the three 3D multiphase equations in the black-oil model, we observe that there will be three second derivative approximation, as exemplified in Equation (2.52), in each one of them – one for each direction. This allows the left-hand side of the equations to be further simplified by the following notation

$$\begin{aligned}\Delta(T_c\Delta\eta)_{i,j,k} \; &= \Delta_x(T_{cx}\Delta_x\eta)_{i,j,k} + \Delta_y\left(T_{cy}\Delta_y\eta\right)_{i,j,k} \\ &+ \Delta_z(T_{cz}\Delta_z\eta)_{i,j,k}\end{aligned} \tag{2.53}$$

Considering now the time derivatives on the right-hand side, if the space domain is fixed, the partial derivative becomes an ordinary derivative to be evaluated at the grid points where the unknowns are defined. Then, discretizing the time derivatives with backwards finite-difference, the terms representing the mass accumulation within a cell may be expressed as

$$\frac{\partial}{\partial t}(f)_{i,j,k} = \frac{1}{\Delta t}\cdot\left({f_{i,j,k}}^{n+1} - {f_{i,j,k}}^{n}\right) = \frac{1}{\Delta t}\cdot\Delta_t f \tag{2.54}$$

where n is the time-step at which the function f will be evaluated, representing a moment in time; Δt is the time difference between two consecutive time-steps; and Δ_t is a finite-difference operator on the time domain. The option for a simpler, first-order approximation of the time derivative stems from instability issues that arise with forward or central-difference techniques.

In addition, to preserve mass conservation and avoid instabilities throughout the simulation, the finite-difference time operator Δ_t must be expanded using a conservative scheme. For the two different functions that arise inside the partial time derivatives in the multiphase equations, this equates to

$$\begin{aligned}\Delta_t\left(\frac{\phi S_c}{B_c}\right) = &\left[\frac{\phi'}{{B_c}^n} + \phi^{n+1}\left(\frac{1}{B_c}\right)'\right]\cdot {S_c}^n\cdot\left({P_o}^{n+1} - {P_o}^n\right) \\ &+ \left(\frac{\phi}{B_c}\right)^{n+1}\cdot\left({S_c}^{n+1} - {S_c}^n\right)\end{aligned} \tag{2.55}$$

and

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\Delta_t\left(\frac{\phi R_s S_o}{B_o}\right) = \left\{\left[\frac{\phi'}{{B_o}^n} + \phi^{n+1}\left(\frac{1}{B_o}\right)'\right]\cdot {R_s}^n + \left(\frac{\phi}{B_o}\right)^{n+1}\cdot {R_s}'\right\} \cdot {S_o}^n \cdot \left({P_o}^{n+1} - {P_o}^n\right) + \left(\frac{\phi}{B_o}\right)^{n+1}\cdot {R_s}^{n+1} \cdot \left({S_o}^{n+1} - {S_o}^n\right) \tag{2.56}$$

in which the following definitions apply

$$\phi' = \frac{\phi^{n+1} - \phi^n}{{P_o}^{n+1} - {P_o}^n} \tag{2.57}$$

$${R_s}' = \frac{{R_s}^{n+1} - {R_s}^n}{{P_o}^{n+1} - {P_o}^n} \tag{2.58}$$

$$\left(\frac{1}{B_c}\right)' = \frac{\left(\frac{1}{{B_c}^{n+1}}\right) - \left(\frac{1}{{B_c}^n}\right)}{{P_o}^{n+1} - {P_o}^n} \tag{2.59}$$

Thus far, only the variable and coefficient terms on the right-hand side of the equations have had specified the time-step in which they should be evaluated. The selection of the backwards-difference operator to discretize the time-derivative terms now implies that the base time of the equations is $n+1$. This leads to the evaluation of the left-hand side terms on the same $n+1$ time-step. Hence, the variable ${P_o}^{n+1}$ will now appear on both sides of the equations, and the transmissibility coefficients, which contain terms that are dependent on pressure – such as B_c, μ_c and R_s – must also be evaluated at this same time-step. Furthermore, other terms – such as k_{rc} – depend on the values of the saturation unknowns, also to be calculated at time-step $n+1$. Consequently, the problem becomes highly coupled and nonlinear. This specific construction of the problem equations is denoted the Fully Implicit formulation (FIM).

The decision to apply a backwards-difference operator to the time derivatives was not the sole one available. The reason for discretizing via backwards-difference is related to the consequent stability of the problem. This option leads to a

PUC-Rio - Certificação Digital Nº 1612782/CA

simulation that is unconditionally stable, regardless of the size of the time-steps, which allows for large time-steps to be taken even in difficult scenarios (strong heterogeneities, gravitational effects, high flow rates, etc.) (Ertekin et al., 2001; and Cheshire et al., 1980). The price to be paid for this stability is that the simulation now requires the simultaneous solution of large, difficult-to-solve systems of equations. Conversely, a forward-difference discretization would lead to a system that is much less demanding to solve, but that is only conditionally stable, usually demanding relatively small time-steps to be capable of achieving convergence. Additionally, it may have greater difficulty representing the physics of the problem; since, for example, in the resulting explicit formulation a pressure transient front can only advance the distance of a single grid block per simulation step.

Finally, the finite difference multiphase flow equations derived in the fully implicit formulation may be written concisely as

$$\Delta[T_o \cdot (\Delta P_o - \gamma_o \Delta Z)]^{n+1} = C_{op}\Delta_t P_o + C_{ow}\Delta_t S_w + C_{og}\Delta_t S_g - {q_{OSC}}^{n+1} \tag{2.60}$$

$$\Delta[T_w \cdot (\Delta P_o - \Delta P_{cow} - \gamma_w \Delta Z)]^{n+1} = C_{wp}\Delta_t P_o + C_{ww}\Delta_t S_w + C_{wg}\Delta_t S_g - {q_{WSC}}^{n+1} \tag{2.61}$$

$$\Delta\left[T_g \cdot \left(\Delta P_o - \Delta P_{cgo} - \gamma_g \Delta Z\right)\right]^{n+1} + \Delta[T_o R_s \cdot (\Delta P_o - \gamma_o \Delta Z)]^{n+1} = C_{gp}\Delta_t P_o + C_{gw}\Delta_t S_w + C_{gg}\Delta_t S_g - {q_{GSC}}^{n+1} \tag{2.62}$$

where the following definitions were employed

$$C_{op} = \frac{\forall_B}{\Delta t} \cdot \left[\frac{\phi'}{{B_o}^n} + \phi^{n+1}\left(\frac{1}{B_o}\right)'\right] \cdot \left(1 - {S_w}^n - {S_g}^n\right) \tag{2.63}$$

$$C_{ow} = -\frac{\forall_B}{\Delta t} \cdot \left(\frac{\phi}{B_o}\right)^{n+1} \tag{2.64}$$

$$C_{og} = -\frac{\forall_B}{\Delta t} \cdot \left(\frac{\phi}{B_o}\right)^{n+1} \tag{2.65}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

$$C_{wp} = \frac{\forall_B}{\Delta t} \cdot \left[\frac{\phi'}{{B_w}^n} + \phi^{n+1} \left(\frac{1}{B_w} \right)' \right] \cdot {S_w}^n \tag{2.66}$$

$$C_{ww} = \frac{\forall_B}{\Delta t} \cdot \left(\frac{\phi}{B_w} \right)^{n+1} \tag{2.67}$$

$$C_{wg} = 0 \tag{2.68}$$

$$\begin{aligned} C_{gp} = \frac{\forall_B}{\Delta t} \cdot \Bigg(& \left\{ \left[\frac{\phi'}{{B_o}^n} + \phi^{n+1} \left(\frac{1}{B_o} \right)' \right] \cdot {R_s}^n + \left(\frac{\phi}{B_o} \right)^{n+1} \cdot {R_s}' \right\} \\ & \times \left(1 - {S_w}^n - {S_g}^n \right) + \left[\frac{\phi'}{{B_g}^n} + \phi^{n+1} \left(\frac{1}{B_g} \right)' \right] \cdot {S_g}^n \Bigg) \end{aligned} \tag{2.69}$$

$$C_{gw} = -\frac{\forall_B}{\Delta t} \cdot \left[\left(\frac{\phi}{B_o} \right)^{n+1} \cdot {R_s}^{n+1} \right] \tag{2.70}$$

$$C_{gg} = \frac{\forall_B}{\Delta t} \cdot \left[\left(\frac{\phi}{B_g} \right)^{n+1} - \left(\frac{\phi}{B_o} \right)^{n+1} \cdot {R_s}^{n+1} \right] \tag{2.71}$$

The expanded version of these equations is presented separately in Appendix A, for reference.

2.5
Linearization of the Multiphase Flow Equations

As previously mentioned, the equations describing the problem in the fully implicit formulation are nonlinear in nature. This presents a significant challenge to the solution of the system of equations that arises in each time-step of the reservoir simulation, since nonlinear systems tend to be much more difficult to solve. The linearization of the nonlinear equations is then of paramount importance.

A further issue that has not yet been handled and which will be addressed presently is the evaluation of the transmissibility terms at the boundary positions. Theoretically, the value of the pressure or saturation dependent terms that constitute

PUC-Rio - Certificação Digital Nº 1612782/CA

the transmissibility coefficients should be evaluated at the particular pressure or saturation level seen at the interface between their two respective grid blocks. However, in practice, these unknowns are determined solely at the cell centers, and are not available at the coordinates of the block boundaries. This ensues the need for a spatial weighing of these coefficients, through information obtained from the individual cell blocks.

Before proceeding with the spatial weighing of the coefficients that arise on the left-hand side of the multiphase equations, it is convenient to notice that they may be subdivided into different terms, representing weak and strong nonlinearities, as well as geometric aspects of the reservoir grid. The weak nonlinearities are related to components which are just function of pressure, and whose variations tend to be smoother, while the strong nonlinearities correspond to the terms which are function either of saturation or capillary pressure, and whose variations are generally more abrupt.

With regards to the capillary pressure terms, it is convenient to represent them in a slightly different manner, making their relation to the saturation unknowns explicit

$$\Delta P_{cow} = P_{cow}{}' \Delta S_w \tag{2.72}$$

$$\Delta P_{cgo} = P_{cgo}{}' \Delta S_g \tag{2.73}$$

in which the following definitions apply

$$P_{cow}{}' = \frac{dP_{cow}}{dS_w} \tag{2.74}$$

$$P_{cgo}{}' = \frac{dP_{cgo}}{dS_g} \tag{2.75}$$

and where these derivatives can be obtained from the capillary pressure curves available from laboratory experiments or from correlation models.

The interblock transmissibilities can then be expressed as

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\xi = G \cdot f_p \cdot f_s \tag{2.76}$$

where G is a geometric factor; f_p is a pressure dependent function; and f_s is a saturation dependent function. The functions f_p and f_s must be determined at the beginning of every time-step, and may assume one of the following forms

$$f_p \equiv \left(\frac{1}{\mu_c B_c}\right) \text{ or } \left(\frac{\gamma_c}{\mu_c B_c}\right) \text{ or } \left(\frac{R_s}{\mu_o B_o}\right) \text{ or } \left(\frac{R_s \gamma_o}{\mu_o B_o}\right) \tag{2.77}$$

$$f_s \equiv (k_{rc}) \text{ or } (k_{rc} P_{cow}{}') \text{ or } \left(k_{rc} P_{cgo}{}'\right) \tag{2.78}$$

The geometric factors, on the other hand, are constant throughout the simulation, and are conditioned only to the grid properties. They may be approximated using the harmonic average of the properties of two contiguous grid blocks. For a block-centered grid, the geometric factor between two cells aligned in the x direction assumes the form

$$G_{i\pm\frac{1}{2},j,k} = \frac{2A_{x_{i,j,k}} k_{x_{i,j,k}} A_{x_{i\pm1,j,k}} k_{x_{i\pm1,j,k}}}{A_{x_{i,j,k}} k_{x_{i,j,k}} \Delta x_{i\pm1,j,k} + A_{x_{i\pm1,j,k}} k_{x_{i\pm1,j,k}} \Delta x_{i,j,k}} \tag{2.79}$$

The weak nonlinearity terms f_p may be approximated in different ways, the most common being upstream weighing and midpoint weighing. For the simulator used in the scope of this work, the method of choice was upstream weighing. This consists of evaluating the pressure dependent function as if it were in the center of the cell which has the greatest flow potential between the two neighbors, that is, the cell which is upstream of the other in the fluid flow. This procedure is exemplified in Equation (2.80), once again for two cells aligned in the x direction

$$\left(\frac{\gamma_c}{\mu_c B_c}\right)_{i+\frac{1}{2},j,k} = \left(\frac{\gamma_c}{\mu_c B_c}\right)_{i,j,k}$$
$$\text{when } \left(P_{c_{i,j,k}} - \gamma_{c_{i+\frac{1}{2},j,k}} Z_{i,j,k}\right) > \left(P_{c_{i+1,j,k}} - \gamma_{c_{i+\frac{1}{2},j,k}} Z_{i+1,j,k}\right) \tag{2.80}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

Here, the pressure used to calculate the value of the fluid properties is usually taken to be the oil phase pressure, even if the fluid in question is water or gas. Neglecting the capillary pressures in this particular application causes little or no loss of accuracy.

The strong nonlinearity terms f_s may also be approximated in different ways, the most common being single-point upstream weighing and two-points upstream weighing. Once more, the method chosen for the simulator used in this work was single-point upstream, as illustrated in Equation (2.80).

It is now possible to return to the main challenge at hand, which is the linearization of the left-hand side in the time domain. There are several different techniques available to accomplish this. The most straightforward – and unstable – one is the explicit method, briefly mentioned beforehand, in which the interblock transmissibilities are simply evaluated at time-step n. Additional methods are: extrapolation, simple iteration, linearized implicit and semi-implicit; all of which improve stability, but that may still be unstable depending on the simulation parameters. To achieve unconditional stability the linearization must be performed with the fully implicit method. Difficult problems, such as those modeled by compositional or thermal simulators, or those containing significant heterogeneities and fracture networks, might only be solvable using this more rigorous approach.

In the fully implicit method, an iterative process is applied to the problem, in which a term to be evaluated at time $n+1$ is approximated by its value at iteration level $v+1$. Furthermore, the value at $v+1$ can be estimated by its value at previous iteration v plus a linear combination of terms arising from the partial differentiation of the term with respect to each of the problem unknowns. This aspect will be further detailed at the appropriate moment.

Next, it is necessary to deal with the nonlinearities present on the right-hand side of the multiphase flow equations. The accumulation coefficients C_{cp}, C_{cw} and C_{cg} all involve exclusively weak nonlinearities. Since they all refer to properties within the cells, no spatial weighing is required here. Moreover, the time linearization in this case cannot be done via the explicit method, by the very nature of the conservative expansion used to generate these coefficients in the first place. The implicit method is also not appropriate for this application, due to new

PUC-Rio - Certificação Digital Nº 1612782/CA

nonlinear terms that would arise in the process. Therefore, the most commonly used method for the accumulation terms is simple iteration, in which the value of the coefficient at time $n+1$ and iteration level $v+1$ is approximated directly by its value at time $n+1$ and previous iteration v.

Finally, the rate terms ${q_{CSC}}^{n+1}$ should be linearized in the time domain in a manner similar to that chosen for the transmissibility coefficients, so as to not introduce instabilities. These terms are related to the problem variables through the inflow performance relationship equations, whose weak and strong nonlinearities are evaluated using the value of the properties in the cell for which the equation is written. Thus, no spatial weighing is required once more.

2.6
Solution of the Linear Finite-Difference Flow Equations

PUC-Rio - Certificação Digital Nº 1612782/CA

The application of the discretization procedure and the spatial and time domain linearization methods described will transform the multiphase partial differential equations into a set of linear finite difference equations. These can now be more thoroughly defined for each individual grid block, using the following additional terminology: subscripts n to represent the cell for which the equations are written, positioned at coordinate $\langle i,j,k\rangle$; m to represent a cell adjacent to the one in question; ψ_n to represent the set of all neighboring cells adjacent to cell n, such that $\psi_n = \psi_x \cup \psi_y \cup \psi_z$; and where Δ_m represents a finite-difference operator in the space domain, defined as

$$\Delta_m \zeta = \zeta_m - \zeta_n \tag{2.81}$$

With this notation, the multiphase flow equations for an arbitrary grid block n may be written as

$$\begin{aligned}\sum_{m\,\epsilon\,\psi_n} {T_{o_{n,m}}}^{n+1}\left(\Delta_m {P_o}^{n+1} - {\bar{\gamma}_{o_{n,m}}}^{n}\Delta_m Z\right) \\ = C_{op_n}\Delta_t P_{o_n} + C_{ow_n}\Delta_t S_{w_n} + C_{og_n}\Delta_t S_{g_n} \\ - {q_{OSC_n}}^{n+1}\end{aligned} \tag{2.82}$$

$$\sum_{m\,\epsilon\,\psi_n} {T_{w_{n,m}}}^{n+1}\left(\Delta_m {P_o}^{n+1} - \Delta_m {P_{cow}}^{n+1} - {\bar{\gamma}_{w_{n,m}}}^{n}\Delta_m Z\right) = C_{wp_n}\Delta_t P_{o_n} + C_{ww_n}\Delta_t S_{w_n} + C_{wg_n}\Delta_t S_{g_n} - {q_{WSC_n}}^{n+1} \tag{2.83}$$

$$\sum_{m\,\epsilon\,\psi_n} \left[{T_{g_{n,m}}}^{n+1}\left(\Delta_m {P_o}^{n+1} + \Delta_m {P_{cgo}}^{n+1} - {\bar{\gamma}_{g_{n,m}}}^{n}\Delta_m Z\right) + {(T_o R_s)_{n,m}}^{n+1}\left(\Delta_m {P_o}^{n+1} - {\bar{\gamma}_{o_{n,m}}}^{n}\Delta_m Z\right)\right] = C_{gp_n}\Delta_t P_{o_n} + C_{gw_n}\Delta_t S_{w_n} + C_{gg_n}\Delta_t S_{g_n} - {q_{GSC_n}}^{n+1} \tag{2.84}$$

where the terms ${\bar{\gamma}_{c_{n,m}}}^{n}$ represent the mean specific weight of the components c, averaged between the values calculated for grid blocks n and m.

The solution of these equations when the fully implicit discretization method is applied in the time domain will require that Newton's Iteration be employed (Ertekin et al., 2001; and Chen et al., 2006). This numerical solution method attempts to find the value of the unknowns that minimize the residuals of the multiphase flow equations. Thus, it commences by rearranging the former equations to residual form

$${R_{o_n}}^{n+1} = \sum_{m\,\epsilon\,\psi_n} {T_{o_{n,m}}}^{n+1}\left(\Delta_m {P_o}^{n+1} - {\bar{\gamma}_{o_{n,m}}}^{n}\Delta_m Z\right) - C_{op_n}\left({P_{o_n}}^{n+1} - {P_{o_n}}^{n}\right) - C_{ow_n}\left({S_{w_n}}^{n+1} - {S_{w_n}}^{n}\right) - C_{og_n}\left({S_{g_n}}^{n+1} - {S_{g_n}}^{n}\right) + {q_{OSC_n}}^{n+1} \tag{2.85}$$

$${R_{w_n}}^{n+1} = \sum_{m\,\epsilon\,\psi_n} {T_{w_{n,m}}}^{n+1}\left(\Delta_m {P_o}^{n+1} - \Delta_m {P_{cow}}^{n+1} - {\bar{\gamma}_{w_{n,m}}}^{n}\Delta_m Z\right) - C_{wp_n}\left({P_{o_n}}^{n+1} - {P_{o_n}}^{n}\right) - C_{ww_n}\left({S_{w_n}}^{n+1} - {S_{w_n}}^{n}\right) - C_{wg_n}\left({S_{g_n}}^{n+1} - {S_{g_n}}^{n}\right) + {q_{WSC_n}}^{n+1} \tag{2.86}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

$$
\begin{aligned}
{R_{g_n}}^{n+1} = \sum_{m\,\epsilon\,\psi_n} \Big[& {T_{g_{n,m}}}^{n+1}\big(\Delta_m {P_o}^{n+1} + \Delta_m {P_{cgo}}^{n+1} \\
& - {\bar{\gamma}_{g_{n,m}}}^{n}\Delta_m Z\big) \\
& + {(T_o R_s)_{n,m}}^{n+1}\big(\Delta_m {P_o}^{n+1} - {\bar{\gamma}_{o_{n,m}}}^{n}\Delta_m Z\big)\Big] \\
& - C_{gp_n}\big({P_{o_n}}^{n+1} - {P_{o_n}}^{n}\big) - C_{gw_n}\big({S_{w_n}}^{n+1} - {S_{w_n}}^{n}\big) \\
& - C_{gg_n}\big({S_{g_n}}^{n+1} - {S_{g_n}}^{n}\big) + {q_{GSC_n}}^{n+1}
\end{aligned}
\tag{2.87}
$$

The next step is to approximate the residual at time-step $n+1$ by its value at iteration $\nu+1$. This can be expressed as

$$
{R_{c_n}}^{n+1} \approx {R_{c_n}}^{n+1^{(\nu+1)}} \tag{2.88}
$$

The residual values at iteration level $\nu+1$ can in turn be approximated by their values at iteration ν, which are already known, plus a linear combination of terms derived from the differentiation of the residual equations with respect to all of the problem unknowns. This amounts to the following expression

$$
\begin{aligned}
{R_{c_n}}^{n+1^{(\nu+1)}} = {R_{c_n}}^{n+1^{(\nu)}} & + \left(\frac{\partial R_{c_n}}{\partial P_{o_n}}\right)^{(\nu)} \delta P_{o_n} + \left(\frac{\partial R_{c_n}}{\partial S_{w_n}}\right)^{(\nu)} \delta S_{w_n} \\
& + \left(\frac{\partial R_{c_n}}{\partial S_{g_n}}\right)^{(\nu)} \delta S_{g_n} \\
& + \sum_{m\,\epsilon\,\psi_n} \left[\left(\frac{\partial R_{c_n}}{\partial P_{o_m}}\right)^{(\nu)} \delta P_{o_m} + \left(\frac{\partial R_{c_n}}{\partial S_{w_m}}\right)^{(\nu)} \delta S_{w_m} \right. \\
& \left. + \left(\frac{\partial R_{c_n}}{\partial S_{g_m}}\right)^{(\nu)} \delta S_{g_m}\right]
\end{aligned}
\tag{2.89}
$$

where the subsequent definitions apply

$$
\delta P_{o_m} = {P_{o_m}}^{n+1^{(\nu+1)}} - {P_{o_m}}^{n+1^{(\nu)}} \tag{2.90}
$$

$$
\delta S_{w_m} = {S_{w_m}}^{n+1^{(\nu+1)}} - {S_{w_m}}^{n+1^{(\nu)}} \tag{2.91}
$$

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\delta S_{g_m} = {S_{g_m}}^{n+1^{(\nu+1)}} - {S_{g_m}}^{n+1^{(\nu)}} \tag{2.92}$$

The definition of the various partial derivatives of the residual equations is left for Appendix A.

Since the objective of the linearization method is to find the value of the unknowns that tend to minimize the system residuals, the final step consists of setting the residual at iteration level $\nu + 1$ equal to zero, and solving the resulting equations for all the unknowns at iteration level $\nu + 1$. That is, solve

$$\begin{aligned} &\left(\frac{\partial R_{c_n}}{\partial P_{o_n}}\right)^{(\nu)} \delta P_{o_n} + \left(\frac{\partial R_{c_n}}{\partial S_{w_n}}\right)^{(\nu)} \delta S_{w_n} + \left(\frac{\partial R_{c_n}}{\partial S_{g_n}}\right)^{(\nu)} \delta S_{g_n} \\ &\quad + \sum_{m\,\epsilon\,\psi_n} \left[\left(\frac{\partial R_{c_n}}{\partial P_{o_m}}\right)^{(\nu)} \delta P_{o_m} + \left(\frac{\partial R_{c_n}}{\partial S_{w_m}}\right)^{(\nu)} \delta S_{w_m}\right. \\ &\quad \left. + \left(\frac{\partial R_{c_n}}{\partial S_{g_m}}\right)^{(\nu)} \delta S_{g_m}\right] = -{R_{c_n}}^{n+1^{(\nu)}} \end{aligned} \tag{2.93}$$

– which is the final form of the residual equation of each phase, in each grid block – simultaneously for all grid blocks. This entails the solution of a linear system of $3N$ equations, per iteration level, per simulation time-step, where N is the total number of grid blocks.

The natural manner of treating this system of equations is to transform it into matrix form and then solve for the unknowns through an adequate numerical linear algebra technique. Therefore, the foremost step is determining how to construct the coefficient matrix $\widetilde{A}$ and the right-hand side vector $\bar{f}$, so that an equation of the form

$$\widetilde{\boldsymbol{A}}\,\bar{\boldsymbol{u}} = \bar{\boldsymbol{f}} \tag{2.94}$$

can be solved for the vector $\bar{\boldsymbol{u}}$.

Observing the structure of the final form of the residual equation, it is apparent that the problem unknowns are contained within terms of the form $\delta\zeta$,

PUC-Rio - Certificação Digital Nº 1612782/CA

where ζ represents either oil pressure, water saturation or gas saturation. So, if the vector containing the actual unknowns is given by

$$\overline{\boldsymbol{x}} = (\overline{\boldsymbol{x}}_1, \overline{\boldsymbol{x}}_2, \overline{\boldsymbol{x}}_3, \cdots, \overline{\boldsymbol{x}}_N)^T \tag{2.95}$$

in which

$$\overline{\boldsymbol{x}}_n = \left(P_{o_n}, S_{w_n}, S_{g_n}\right)^T \tag{2.96}$$

then we can define the following differential solution vector

$$\overline{\boldsymbol{u}} = \delta\overline{\boldsymbol{x}} = \overline{\boldsymbol{x}}^{n+1^{(\nu+1)}} - \overline{\boldsymbol{x}}^{n+1^{(\nu)}} \tag{2.97}$$

and the following initial condition

$$\overline{\boldsymbol{x}}^{n+1^{(0)}} = \overline{\boldsymbol{x}}^{n} \tag{2.98}$$

Returning to the final form of the residual equation (2.93), it is clear that the right-hand side vector equates to the negative of the residual calculated from the solution of the previous iteration

$$\overline{\boldsymbol{f}} = -\overline{\boldsymbol{R}}^{n+1^{(\nu)}} \tag{2.99}$$

where

$$\overline{\boldsymbol{R}} = (\overline{\boldsymbol{R}}_1, \overline{\boldsymbol{R}}_2, \overline{\boldsymbol{R}}_3, \cdots, \overline{\boldsymbol{R}}_N)^T \tag{2.100}$$

in which

$$\overline{\boldsymbol{R}}_n = \left(R_{o_n}, R_{w_n}, R_{g_n}\right)^T \tag{2.101}$$

with R_{c_n} being defined in the initial form of the residual equations (2.85) – (2.87).

PUC-Rio - Certificação Digital Nº 1612782/CA

Lastly, the coefficients multiplying the problem unknowns, in differential form, are the partial derivatives of the residuals, with respect to the corresponding unknowns. A matrix with coefficients of this kind is denoted a Jacobian matrix. Therefore, the general format of the coefficient matrix is the following

$$\widetilde{\boldsymbol{A}} = \widetilde{\boldsymbol{J}} = \begin{bmatrix} \widetilde{\boldsymbol{J}}_{1,1} & \cdots & \widetilde{\boldsymbol{J}}_{n,m} \\ \vdots & \ddots & \vdots \\ \widetilde{\boldsymbol{J}}_{n,m} & \cdots & \widetilde{\boldsymbol{J}}_{N,N} \end{bmatrix} \tag{2.102}$$

in which

$$\widetilde{\boldsymbol{J}}_{n,m} = \begin{bmatrix} \frac{\partial R_{o_n}}{\partial P_{o_m}} & \frac{\partial R_{o_n}}{\partial S_{w_m}} & \frac{\partial R_{o_n}}{\partial S_{g_m}} \\ \frac{\partial R_{w_n}}{\partial P_{o_m}} & \frac{\partial R_{w_n}}{\partial S_{w_m}} & \frac{\partial R_{w_n}}{\partial S_{g_m}} \\ \frac{\partial R_{g_n}}{\partial P_{o_m}} & \frac{\partial R_{g_n}}{\partial S_{w_m}} & \frac{\partial R_{g_n}}{\partial S_{g_m}} \end{bmatrix} \tag{2.103}$$

Analogously to the procedure employed with the solution vector, at the first iteration level (when $\nu = 0$), the residual vector and Jacobian matrix are evaluated using the pressure and saturation values determined at the end of the previous time-step n.

The submatrices $\widetilde{\boldsymbol{J}}_{n,m}$ defined in Equation (2.103) represent the partial derivatives of the three residual equations related to each grid block n, taken with respect to the pressure and saturation unknowns of grid block m. However, observing the initial residual equations, it becomes clear that these derivatives will equate to zero for every grid block m that is not adjacent to n, or that is not grid block n itself. Consequently, the coefficient matrix will be a sparse matrix[3], composed mostly of zeros, with non-zero block elements occurring only on the main block diagonal and some off-diagonal blocks. The degree of sparsity may vary

[3] **Sparse Matrix** – Term referring to a matrix in which most of the elements are zero (Peaceman, 1977). Its sparsity may be defined as the percentage of zero elements in the total element count. An alternative description is presented by Wilkinson, to whom a matrix may be considered sparse if it is advantageous to exploit the presence of the zeros, to save time and memory usage (Davis and Hu, 2010).

PUC-Rio - Certificação Digital Nº 1612782/CA

significantly throughout the simulation process, usually being on the order of $O(N)$(Stüben, 2007), but eventually reaching levels as high as 95%, depending on the problem (Sheth and Younis, 2017). These off-diagonals with non-zeros correspond to the coupling that exists between neighboring cells, that is, the relation between the oil, water and gas residuals of one cell to the pressure and saturation unknowns of the adjacent cells. The actual shape of the sparse Jacobian matrix will then be intrinsically associated to the ordering scheme chosen for the reservoir grid, as has been briefly explained, and to the manner with which the residual equations are chosen to be numbered. For example, in the formulation presented in Equations (2.96), (2.100) and (2.103) the residual variables were grouped together per grid block, forming a repeating sequence of oil, water and gas equations. An alternative scheme might be to group together all the oil equations, then all the water equations and finally all the gas equations; this would significantly change the shape of $\widetilde{\boldsymbol{J}}$. These different options for constructing the coefficient matrix can have direct implications in the numerical methods to be utilized for solving the linear system of equations, as will be shown further ahead.

The reservoir simulator used in this research provides both possibilities of equation numbering, either grouped per grid block or per equation type. The grid ordering scheme available thus far is natural ordering.

With these given definitions, the matrix equation of the system may now be expressed as

$$\widetilde{\boldsymbol{J}}^{(\nu)} \cdot \delta\overline{\boldsymbol{x}} = -\overline{\boldsymbol{R}}^{(\nu)} \tag{2.104}$$

The solution procedure using Newton's Iteration starts with the evaluation of the residuals and their partial derivatives at the first iteration level, leading to the construction of the Jacobian matrix $\widetilde{\boldsymbol{J}}^{(\nu=0)}$ and the residual vector $\overline{\boldsymbol{R}}^{(\nu=0)}$. Next, the matrix equation is solved via an appropriate numerical method and the differential solution vector $\delta\overline{\boldsymbol{x}}$ is determined. From this, the next solution vector is obtained

$$\overline{\boldsymbol{x}}^{n+1^{(\nu=1)}} = \delta\overline{\boldsymbol{x}} + \overline{\boldsymbol{x}}^{n+1^{(\nu=0)}} = \delta\overline{\boldsymbol{x}} + \overline{\boldsymbol{x}}^{n} \tag{2.105}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

Continuing the process, a new Jacobian matrix $\tilde{\boldsymbol{J}}^{(1)}$ and residual vector $\bar{\boldsymbol{R}}^{(1)}$ can be constructed, evaluated at $\bar{\boldsymbol{x}}^{(1)}$, and the new matrix equation can be solved so as to obtain a new $\delta\bar{\boldsymbol{x}}$ vector, from which $\bar{\boldsymbol{x}}^{(2)}$ is calculated subsequently.

The method then proceeds in identical fashion for iterations $\nu = 2, 3, \ldots$ until convergence is achieved or, alternatively, if a pre-established maximum number of iterations are attempted. Once convergence is reached, the value of $\bar{\boldsymbol{x}}^{(\nu)}$ at the final iteration level is considered to be the solution to the current time-step $\bar{\boldsymbol{x}}^{n+1}$. Finally, the simulation advances to the next time-step and this same iterative process starts anew.

As a last remark, there are two notable alternative solution methods available for solving the simulation unknows. The first is named Implicit Pressure Explicit Saturations (IMPES), because, as the name suggests, the pressure unknowns are solved implicitly, while the saturations are determined explicitly. To accomplish this, the equations in each grid block are initially combined so as to eliminate the saturation variables, and then the pressure system is solved for simultaneously as a first step. Afterwards, the pressure values obtained are used directly in the saturation equations written for each grid block. The rationale behind this method is the fact that pressure is considered to fluctuate more intensely and that its variation travels farther into the reservoir. Saturations, on the other hand, tend to alter less overall, and these variations move more slowly throughout the reservoir. The second method is named Adaptative Implicit Method (AIM) and it combines aspects of the Fully Implicit Method and of IMPES. The concept here is that some grid blocks are treated in a fully implicit manner, while in others the pressures are treated implicitly and the saturations explicitly. This is done because of eventual convergence difficulties that might arise with the IMPES method, due to the fact that in some regions of the reservoir the saturations could actually be undergoing significant variations. This is especially true in water or gas fronts emanating from injection wells.

Nevertheless, the linear system *solvers* implemented in the scope of this work focused uniquely on the solution of systems arising from the fully implicit method, with no treatment given to any other solution method. IMPES simulations are commonly capable of being executed in shorter runtimes than AIM or FIM, but with a price to be paid in terms of accuracy. Therefore, best overall efficiency is

PUC-Rio - Certificação Digital Nº 1612782/CA

probably achieved by applying IMPES to *easy* problems, and AIM or FIM to more *difficult* ones (Aziz and Settari, 1979).

PUC-Rio - Certificação Digital Nº 1612782/CA

3
Numerical Methods

The field of numerical linear algebra strives to solve large systems of equations, written in matrix form, often originating from the mathematical modeling of physical problems. One of the principal sources of matrix equations to be solved are those resulting from the discretization of partial differential equations, such as those describing the multiphase flow of fluid in porous media. This discretization process generally leads to very large sparse linear systems, which requires an efficient numerical method to be solved within a reasonable timeframe (Trefethen and Bau III, 1997; Behie et al., 1984).

There are a variety of methods available for this purpose, with each presenting advantages and disadvantages that must be carefully weighed so as to choose the one most suited for the desired application. The selection of an appropriate method is of particular importance in reservoir simulations due to the fact that the solution of the linear system of equations being built at each iteration level is one of the most processing-intensive, time-consuming steps of the simulation (Brown et al., 2015; Sheth and Younis, 2017; and SPE – Reservoir Simulation).

PUC-Rio - Certificação Digital Nº 1612782/CA

3.1
Direct Methods

The solution methods originally developed to tackle linear systems of equations belong to the category of Direct Methods, as described by Price and Coats (1974). Among these methods, the most traditional one is known as Gaussian Elimination (GE); which involves the factorization of the coefficient matrix $\boldsymbol{A}$ into lower and upper triangular matrices, respectively $\boldsymbol{L}$ and $\boldsymbol{U}$, and the subsequent solution of two simpler systems via forward and then backward substitution (Lay, 2003). This is exemplified in Equations (3.1) – (3.5):

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{3.1}$$

where

$$\boldsymbol{A = LU} \tag{3.2}$$

is transformed into

$$\boldsymbol{LUx = b} \tag{3.3}$$

which can be easily solved in the following two steps

$$\boldsymbol{Ly = b} \tag{3.4}$$

$$\boldsymbol{Ux = y} \tag{3.5}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

In the equations hereafter, boldface capital letters shall be used to represent matrices, while boldface lower-cased letters shall be used to represent vectors. Constants and scalar values will be represented by non-bold lower-cased letters. To simplify notation, matrices and vectors will not be identified with any symbols above their names (i.e. tilde or bar), unless otherwise specified.

A setback of the process described in Equations (3.1) – (3.5) involves the fact that, in constructing $\boldsymbol{L}$ and $\boldsymbol{U}$, often it is not sufficient to apply straightforward factorization to $\boldsymbol{A}$, since it may be prone to instability due to round-off errors, and so pivoting techniques must also be introduced to the procedure. This entails additional calculations and the storage of at least one extra matrix. Moreover, when dealing with sparse matrices, Gaussian Elimination or other factorization schemes such as QR, SVD[4] or Cholesky (the latter being applicable only to Hermitian[5] positive definite[6] matrices) tend to introduce vast quantities of non-zero terms into their respective factor matrices – which greatly increases the cost of solving the

[4] **SVD** – Singular Value Decomposition.

[5] **Hermitian Matrix** – A complex square matrix that is equal to its conjugate transpose. Represents the generalization of a real symmetrical matrix in the complex domain ($A = A^T \rightarrow A = A^*$).

[6] **Positive Definite Matrix** – A matrix whose eigenvalues are all positive. Alternatively, a matrix for which $x^T Ax > 0$, given any nonzero $x \in \mathcal{R}^m$.

system, since there will be many more numbers on which to operate (Trefethen and Bau III, 1997). Furthermore, for similar reasons, these techniques also significantly augment the storage requirement of the solution process, so as to keep all the new non-zero terms that arise.

Due to these considerations, the cost of solving a linear system through direct methods can be quite elevated, reaching the order of $O(m^3)$ operations for a dense coefficient matrix, where m represents the matrix dimension (Trefethen and Bau III, 1997). Since a *solver's* runtime is strongly correlated to the number of operations that must be performed, this means that the time required for a direct *solver* to reach the solution increases very rapidly as the problem dimensions grow in size. Even though for sparse matrices it is possible to exploit the sparsity pattern to reduce the number of operations required by a reasonable factor, the resulting count would still represent a major restriction in the ability of the reservoir engineer to increase the number of grid blocks used to model the reservoir. As the reservoir is further and further discretized, the time (and memory) required to solve the resulting system of equations via direct methods simply becomes prohibitive (Mattax and Dalton, 1990). Consequently, alternative solution techniques must be adopted to overcome this limitation – leading to the application of a class of algorithms called Iterative Methods.

Even though there exist other direct methods that do not explicitly factor the coefficient matrix, such as Gauss-Jordan Reduction or Thomas' Algorithm, normally they either naturally require a greater number of operations or are restricted to specific matrix structures – for example, tridiagonal matrices (Ertekin et al., 2001). These limitations often make them unsuited to solve modern reservoir simulation problems, with more complex matrix structures.

PUC-Rio - Certificação Digital Nº 1612782/CA

3.2
Iterative Methods

The overall concept behind Iterative Methods is to start with an initial guess to the solution vector x and successively improve this solution estimation in each step of an iterative process, gradually progressing towards the actual solution. The objective at each iteration step is to find a search-direction vector that most efficiently points from the current intermediate solution to the true solution, as well

as to find the length of the path that should be pursued in that particular direction. This true solution, however, is seldom actually reached in practice, because the method is normally halted beforehand, whenever an intermediate solution comes sufficiently close to it.

This particular characteristic of iterative methods, of delivering only approximate solutions to the problems to which they are applied, might cause them to seem less powerful than direct ones at first glance. However, oftentimes this is a misconception. First, because the iterative method may well be capable of reaching a solution as precise as one obtained by a direct method, given sufficient iteration steps. More importantly though, they possess a very powerful trait not found in direct methods, which is the fact that they may be able to provide a *precise enough* answer with an appreciably smaller operation count; whereas direct methods will only provide *any* answer at all after every operation has been concluded. Consequently, the typical performance delivered by iterative methods in terms of operation count is in the order of $O(m^2)$, which represents a very significant improvement from the work needed by general direct methods (Trefethen and Bau III, 1997).

Furthermore, either kind of method may be considered inexact in the sense that, when carried out on a computer, their results will be accurate, at best, to *machine precision* ε_M; even in the absence of rounding errors. In practice, for most methods, the normal *machine error* will be on the order of 100 to 1000 times ε_M. Evidently, for iterative methods, the actual error of the final solution will also depend on the convergence criteria adopted to end the iterative process, and the rigor of this criteria will be contingent upon the accuracy of the answer desired by the user. Hence, achieving rapid convergence for a given error tolerance is the main objective of any iterative method (Trefethen and Bau III, 1997).

The customary choice of convergence criteria employed in iterative methods designed to solve linear system of equations is presented in Equation (3.6)

$$\|\boldsymbol{r}_n\| < \eta \cdot \|\boldsymbol{b}\| \tag{3.6}$$

where $\boldsymbol{r}_n$ is the residual vector found during iteration level n of the linear iterative method; and η is an arbitrarily specified parameter, denoted residual tolerance. For

PUC-Rio - Certificação Digital Nº 1612782/CA

all of the simulations performed throughout this research, the chosen residual tolerance equaled $\eta = 10^{-10}$. It should be apparent that the smallest the value chosen, the more precise the final result of the numerical method will be; however, the more expensive it will be to reach that result.

The convergence criteria detailed in Equation (3.6) should not be confused with the one applied to Newton's Iteration in the linearization process performed at each simulation time-step, which is based upon material balance considerations. Likewise, the term mentioned Equation (3.6) as being the residual vector is not the same as the one referred to in Chapter 2, associated with the mass-balance equations (which is actually the right-hand side vector $\boldsymbol{b}$ here), but instead it relates to the error between the projection of the current intermediate solution, found at iteration n, and the right-hand side vector; as such

$$\boldsymbol{r}_n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n \tag{3.7}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

Finally, independent of the precision desired, a method cannot be allowed to run freely until obtaining convergence, since it may eventually stagnate or even become unstable along the process. Therefore, it is also necessary to define an upper limit to the number of permissible iterations. The iteration limit defined for all the simulations performed throughout this research was $\mathrm{iter\_lim} = 15000$.

Stationary Methods

Historically, the first group of iterative methods to be developed were based upon the relaxation of the problem coordinates, eliminating components of the residual vector through sequential relaxation steps (Saad, 2003; Peaceman, 1977). These classical methods, known as stationary methods (Barrett et al., 1994), involve the splitting of the coefficient matrix $\boldsymbol{A}$ into two components

$$\boldsymbol{A} = \boldsymbol{A}_1 + \boldsymbol{A}_2 \tag{3.8}$$

and then converting matrix equation (3.1) into a fixed-point problem

$$\boldsymbol{x}_{n+1} = \boldsymbol{A}_1{}^{-1}(\boldsymbol{b} - \boldsymbol{A}_2\boldsymbol{x}_n) = \boldsymbol{B}\boldsymbol{x}_n + \boldsymbol{c} \tag{3.9}$$

where $\boldsymbol{B}$ and $\boldsymbol{c}$ are independent of the iteration step.

Examples of this class of algorithms include Jacobi Iteration, Gauss-Seidel (GS), as well as different kinds of Successive Overrelaxation Methods (SOR). However, none of these methods are powerful enough to efficiently solve very large systems of equations, such as the ones arising in modern reservoir simulations (Chen et al., 2006; Kelly, 1995).

Dimensional Splitting Methods

One of the oldest class of iterative methods developed was based upon the concept of splitting the dimensions of the problem and solving them independently, as one-dimensional problems, with the aid of some parameters that varied with each iteration. Of particular relevance was the Alternating-Direction Implicit Procedure (ADIP), which was popular in reservoir simulation applications in the 1960's and 1970's, before other, more powerful methods were established (Ertekin et al., 2001; Peaceman, 1977).

Approximate Factorization Methods

Another group of methods developed early on was based upon approximate-factorization techniques. This involves the factorization of the coefficient matrix into factors that do not decompose it in an exact manner, but which are easier to compute and whose storage is less expensive. The most important of these to be applied to reservoir simulation is Stone's Strongly Implicit Procedure (SIP) from 1968, later improved by Weinstein et al. in 1969 and 1970, in which the problem is transformed into

$$\boldsymbol{L}'\boldsymbol{U}' \cdot (\boldsymbol{x}_{n+1} - \boldsymbol{x}_n) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n \tag{3.10}$$

where $\boldsymbol{L}'$ and $\boldsymbol{U}'$ are approximate factors of A (Mattax and Dalton, 1990; Peaceman, 1977).

PUC-Rio - Certificação Digital Nº 1612782/CA

As shall be shown further ahead, there are other approximate factorization techniques available, but they were usually introduced to act as a preconditioner to some other iterative method and, unlike SIP, were not developed to be stand-alone procedures.

Projection Methods

More recently, a new class of non-stationary algorithms has been introduced into the reservoir simulation field, based on the projection of m-dimensional problems into a lower-dimensional Krylov subspace $\mathcal{K}_n$. This means that, for a given matrix equation, the search for the solution vector x is performed on the subspace spanned by the set of vectors composing a Krylov sequence. This subspace during iteration level n is defined as

$$\boldsymbol{\mathcal{K}}_n = Span\{\boldsymbol{b}, \boldsymbol{Ab}, \boldsymbol{A}^2\boldsymbol{b}, \boldsymbol{A}^3\boldsymbol{b}, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{b}\} \tag{3.11}$$

or, alternatively, if $\boldsymbol{x}_0 = 0$ (which is a common choice)

$$\boldsymbol{\mathcal{K}}_n = Span\{\boldsymbol{r}_0, \boldsymbol{Ar}_0, \boldsymbol{A}^2\boldsymbol{r}_0, \boldsymbol{A}^3\boldsymbol{r}_0, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{r}_0\} \tag{3.12}$$

where this sequence keeps growing larger as the iterative process advances.

Therefore, the intermediate solutions will be contained within the following subspace

$$\boldsymbol{x}_n \in \boldsymbol{x}_0 + \boldsymbol{\mathcal{K}}_n \tag{3.13}$$

Moreover, the intermediate solutions may be calculated from

$$\boldsymbol{x}_n = \boldsymbol{x}_0 + \sum_{j=0}^{n} \xi_j \boldsymbol{A}^j \boldsymbol{b} \tag{3.14}$$

where ξ represents the linear coefficients multiplying each basis vector (Saad, 2003).

PUC-Rio - Certificação Digital Nº 1612782/CA

Methods belonging to this class are quite proficient at solving large systems of equations and will be the focus of this thesis. The original Krylov projection method is named Conjugate[7] Gradient (CG) and was developed by Hestenes and Stiefel in 1952 (Trefethen and Bau III, 1997). Although it is very efficient and remains popular, its application is restricted to symmetric positive definite (SPD) matrices, a prerequisite that makes it unsuited for solving the matrix equations encountered in fully implicit multiphase flow, whose coefficient matrix is highly asymmetrical and indefinite (Stüben et al., 2007). For this reason, modern reservoir simulators must employ newer, more versatile methods, that combine speed with the ability to tackle complex problems. The focus of this research will be the study of three particular methods that tend to best encompass these characteristics, of efficiency and robustness, and which are most commonly applied to current reservoir simulators (Jackson et al., 2014; SPE – Reservoir Simulation Linear Equation Solver). Specifically, it shall compare the performance of the following algorithms: (i) ORTHOMIN, (ii) GMRES and (iii) BiCGSTAB.

The necessity of comparing different methods such as these stems from the fact that there is not a fundamental rule determining an overall best method. The performance of the methods is strongly related to the application in which they are used, with each method possibly being the one most suited for a specific class of problems, while being the worst one for a separate class (Berrett et al., 1994).

Furthermore, as computer processing power improves, the difference in performance between methods of different quality tends to be accentuated; thus, as the computer becomes faster, the more important it becomes to optimize the numerical *solver* (Trefethen and Bau III, 1997).

One straightforward manner of comparing the different methods would be through the cost required for them to perform each iteration step. Table 3.1 summarizes both the computational cost, in terms of the various operations involved in the algorithms, and the memory necessary for each method to be executed.

PUC-Rio - Certificação Digital Nº 1612782/CA

[7] **Conjugate Vectors** – Two non-zero vectors $p \in \mathcal{R}^m$ and $q \in \mathcal{R}^m$ are defined as being conjugate with respect to matrix A if the following property holds true: $p^T Aq = 0$.

Table 3.1 – Computational cost and memory requirement of the iterative methods studied.

Iterative Method	Computational Cost			Memory Requirement
	MV	AXPY	DOT	
ORTHOMIN(*k*)	2	~ 2 + k/2	~ 3 + *k*/2	1 + 2*k*
GMRES(*k*)	1	~ *k*/2	~ 1½ + *k*/2	2 + *k*
BiCGSTAB	2	6	6	7

Here, Memory Requirement represents the number of additional vectors that must be stored (excluding $\boldsymbol{x}$ and $\boldsymbol{b}$); MV represents the number of matrix-vector multiplications; AXPY represents the number of vector additions and subtractions, combined with a scalar multiplication; and DOT represents the total number of inner products and vector norm calculations performed.

Although this table may serve as a reference for the cost of using any particular method, the number of necessary iteration steps to solve a given reservoir problem will vary between algorithms. Hence, to accurately assess their performances, it is essential to compare the total runtime demanded by each one of them. This entails running them on an actual simulator, using sample reservoir models, and analyzing the times required for their convergence. These tests and their results will be detailed in Chapter 5.

With regards to the convergence capability of the methods examined, in contrast to the CG method, whose result is assured to converge after at most m iterations, these do not have a well-established general convergence theory; in fact, they may even diverge altogether. This behavior is often related to the conditioning of the coefficient matrix, which may be expressed by the matrix's condition number $\kappa(\boldsymbol{A})$, an attribute related to the perturbation behavior of the problem.

To visualize this, matrix equation (3.1) may be seen as a function mapping an independent vector $\boldsymbol{x}$ onto a resulting vector $\boldsymbol{b}$

$$\boldsymbol{x} \rightarrow f(\boldsymbol{x}) = \boldsymbol{b} \tag{3.15}$$

For a well-conditioned problem, small perturbations in $\boldsymbol{x}$ lead to only small perturbations in $f(\boldsymbol{x})$, while in ill-conditioned problems they may lead to large

PUC-Rio - Certificação Digital Nº 1612782/CA

perturbations (Trefethen and Bau III, 1997). The condition number of a matrix is defined as

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\| = \frac{\sigma_{max}}{\sigma_{min}} \tag{3.16}$$

where σ_{max} and σ_{min} represent the maximum and minimum singular values, respectively. Alternatively, if matrix $\boldsymbol{A}$ is normal (that is, $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A}$), then it may also be written as

$$\kappa(\boldsymbol{A}) = \frac{\|\lambda_{max}\|}{\|\lambda_{min}\|} \tag{3.17}$$

where λ_{max} and λ_{min} represent the maximum and minimum eigenvalues, respectively.

The smaller the condition number, the more concentrated the spectrum of the matrix (the smaller the spectral radius[8]), and thus the better conditioned it is. Unfortunately, the Jacobian matrices built during each step of Newton's Iteration are generally very ill-conditioned. This implies that if one attempts to solve the corresponding system of linear equations straightaway with one of the algorithms mentioned here, there is a high probability that the iterative method will stagnate. Indeed, in practical applications, these methods are rarely functional without the aid of a preconditioner. The aim of the preconditioner is, just as the name suggests, to better condition the problem before attempting to solve it or, in other words, to reduce its condition number. This theme will be further explored in 87 and different preconditioning strategies shall be evaluated. Preconditioning is of such importance to the solution methods that oftentimes they are tailored to the specific application at hand and, frequently, constructing the preconditioning matrix and performing calculations with it essentially consumes more time than the underlying iterative method itself (Stüben et al., 2007). For this reason, the iterative algorithms are

[8] **Spectrum of a Matrix** – Refers to its complex set of eigenvalues, with the **Spectral Radius** representing the magnitude of the largest eigenvalue.

PUC-Rio - Certificação Digital Nº 1612782/CA

sometimes called accelerators and the numerical *solvers* for a specific problem are actually named after the preconditioning strategy.

Nonetheless, for now this feature will be simply integrated directly into the original versions of the aforementioned methods, to form their respective preconditioned versions. This entails that one or more steps of the algorithm will consist of pre-multiplying some vector $\boldsymbol{r}$ by $\boldsymbol{M}^{-1}$, where $\boldsymbol{M}$ is the preconditioning matrix, to obtain a resulting vector $\boldsymbol{q}$. The cost of preconditioning, in terms of the number of times it is applied per iteration, is presented in Table 3.2 for each method.

Table 3.2 – Computational cost of applying a preconditioner to the iterative methods.

	Computational Cost
Iterative Method	Preconditioning
ORTHOMIN(*k*)	1
GMRES(*k*)	1
BiCGSTAB	2

The memory requirement associated with the preconditioners may vary widely. It may be as inexpensive as storing a single extra vector, or as costly as storing an additional matrix the size of $\boldsymbol{A}$; and eventually even larger. More will be said of this in Chapter 5.

PUC-Rio - Certificação Digital Nº 1612782/CA

3.2.1 ORTHOMIN Method

Prior to the development of the ORTHOMIN method by Vinsome (1976), SIP was the most commonly used method in reservoir simulators. This new method, based on the minimization of the residual vector computed at each iteration level, proved not only to be very competitive for solving reservoir problems, but also introduced a set of advantages to previous methods in use, such as (i) not depending on arbitrary iteration parameters, whose optimal value needed to be estimated; (ii) its convergence is insensitive to transmissibility ratios and asymmetry of the coefficient matrix; (iii) and being applicable to matrices with any number of diagonal bands (Vinsome, 1976).

Ultimately, the aim of the method is to minimize the following objective function

$$\|\boldsymbol{r}_{n+1}\| = \|\boldsymbol{r}_n - \alpha \boldsymbol{A}\boldsymbol{q}_n\| = minimum \tag{3.18}$$

Therefore, the step-length α used to advance from one intermediate solution $\boldsymbol{x}_n$ to the next $\boldsymbol{x}_{n+1}$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha \boldsymbol{q}_n \tag{3.19}$$

may be determined directly from

$$(\boldsymbol{A}\boldsymbol{q}_n)^T \cdot (\boldsymbol{r}_n) = (\boldsymbol{A}\boldsymbol{q}_n)^T \cdot (\alpha \boldsymbol{A}\boldsymbol{q}_n) \tag{3.20}$$

which, in turn, also implies that the new residual vector $\boldsymbol{r}_{n+1}$ computed at each step

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha \boldsymbol{A}\boldsymbol{q}_n \tag{3.21}$$

will be orthogonal to the previous projection vector $\boldsymbol{A}\boldsymbol{q}_n$.

Furthermore, at every iteration step the new search vector $\boldsymbol{q}_n$ is constructed in a manner so that its $\boldsymbol{A}$-projection is orthogonal to all of the previously generated projections ($(\boldsymbol{A}\boldsymbol{q}_n) \perp (\boldsymbol{A}\boldsymbol{q}_i)\,, \forall\, i < n$). This attempts to optimize the search for the true solution by covering an ample search region.

On the other hand, this manner of selecting the search vectors leads to a marked disadvantage of the method, which is the fact that, as the iterative process progresses, the additional storage requirement to keep all of the previous vectors may become significant, potentially growing to the size of the original coefficient matrix. Moreover, this feature also means that an increasing number of orthogonality coefficients β must be calculated each time, which causes the orthogonalization process to become increasingly more costly and susceptible to rounding errors (Vinsome, 1976).

However, in practice, these orthogonalized projection vectors do not need to all be stored, and the method may be constrained so as to maintain only the previous

PUC-Rio - Certificação Digital Nº 1612782/CA

k vectors, without losing its ability to converge. This leads to the ORTHOMIN(k) variation of the method. There is no theory as of yet on an optimal number for the parameter k, though traditionally it is set at some pre-established level and kept constant throughout the process. For this research a novel approach was taken, as introduced by Baker et al. (2009), in which the reset parameter is permitted to vary between cycles. Although it was originally proposed for the GMRES(k) method, this same concept is also applicable for ORTHOMIN(k), and was found to accelerate the convergence rate when compared to a fixed reset strategy.

The idea behind the procedure is that the drop in the residual norm between iterations is correlated to the angle between their residual vectors

$$\cos \angle(\boldsymbol{r}_{n+1}, \boldsymbol{r}_n) = \frac{\|\boldsymbol{r}_{n+1}\|}{\|\boldsymbol{r}_n\|} = \gamma \tag{3.22}$$

where γ is the convergence rate, and where smaller values represent faster convergence.

If the reduction in the residual norm is significative, then the angle between the residual vectors should be large. To the limit, if consecutive residual vectors are determined to be orthogonal, then an exact solution has been found.

The strategy consists of varying the reset parameter k between a maximum value k_{max} and a minimum value k_{min}, through steps of size δ, depending on the quality of the rate of convergence. It starts by adopting $k = k_{max}$ and then gradually decreasing it every time that the convergence rate between cycles is worse than γ_{min}, but better than γ_{max}, until the limit of $k = k_{min}$ is reached. Alternatively, if convergence is better than γ_{min} then k is left unaltered, while if convergence is worse than γ_{max} then k is reset back directly to its maximum value k_{max}. Similarly, once k reaches its minimum value k_{min}, if a further attempt is made to reduce it, instead of doing so the procedure resets it directly back to k_{max}.

The values of these parameters were originally chosen from the authors' recommendations, with $k_{max} = 30$, which is a typical value for the reset parameter in any case; $k_{min} = 3$; $\delta = 3$; $\gamma_{min} = \cos(80°) \approx 0.175$; and $\gamma_{max} = \cos(8°) \approx 0.99$. They were further tested for optimization in this work, but since they were

PUC-Rio - Certificação Digital Nº 1612782/CA

found to provide the best results in several cases, they were maintained as suggested.

The pseudocode of the preconditioned ORTHOMIN method implemented is presented in Algorithm 3.1 (Ertekin et al., 2001). Besides the different resetting strategy, a second alteration was made to the original version of the algorithm presented in literature. In particular, whenever starting a new reset cycle, the first search vector is taken directly as the final search vector of the previous cycle, instead of being computed anew from the final residual vector of the previous cycle. This adaptation was found to consistently aid convergence in the test cases performed.

As a note, the matrix storage format adopted for the work done in this research was the traditional Compressed Sparse Row (CSR) format. In it, matrices are not stored in their full forms, with all zero and non-zero entries arranged according to their coordinate positions. Instead, it represents the matrices in an abstract manner, through the use of three auxiliary vectors: (i) row_ptr, which stores the address of where the first non-zero in each row is located; (ii) col_ind, which stores the column index of the non-zero terms sequentially, row by row; and (iii) nz_values, which keeps the actual values of the non-zero entries, also grouped row wise. Consequently, all mathematical operations done on the matrices, as well as the construction of the different preconditioning matrices, had to be tailored to conform to this format.

PUC-Rio - Certificação Digital Nº 1612782/CA

Algorithm 3.1 – Preconditioned ORTHOMIN Iterative Method.

$\boldsymbol{x}_0 = 0$

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$

$\boldsymbol{q}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0$

$Check\ convergence\ (\|\boldsymbol{r}_0\| < \eta \cdot \|\boldsymbol{b}\|)$

$\rho = \|\boldsymbol{r}_0\|, \qquad \gamma = 1$

while $j < iter\_lim$

$Test\ quality\ of\ convergence\ (\gamma)\ and\ adjust\ cycle\ size\ (k)\ accordingly$

for $j = 0$ **to** k

$\alpha = \frac{\langle \boldsymbol{r}_j \cdot \boldsymbol{A}\boldsymbol{q}_j \rangle}{\langle \boldsymbol{A}\boldsymbol{q}_j \cdot \boldsymbol{A}\boldsymbol{q}_j \rangle}$

$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha \boldsymbol{q}_j$

$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha \boldsymbol{A}\boldsymbol{q}_j$

$Check\ convergence\ \left(\|\boldsymbol{r}_j\| < \eta \cdot \|\boldsymbol{b}\|\right)$

for $i = 0$ **to** j

$\beta_i = \frac{\langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{r}_{j+1} \cdot \boldsymbol{A}\boldsymbol{q}_i \rangle}{\langle \boldsymbol{A}\boldsymbol{q}_i \cdot \boldsymbol{A}\boldsymbol{q}_i \rangle}$

end

$\boldsymbol{q}_{j+1} = \boldsymbol{M}^{-1}\boldsymbol{r}_{j+1} + \sum_{i=j-k+1}^{j} (\beta_i \boldsymbol{q}_j)$

end

$\gamma = \frac{\|\boldsymbol{r}_k\|}{\rho}$

$\rho = \|\boldsymbol{r}_k\|$

$\boldsymbol{q}_0 = \boldsymbol{q}_k$

end

3.2.2 GMRES Method

The Generalized Minimum Residual method is an extension of the Minimum Residual (MINRES) method, which had been developed for symmetric matrices, to encompass asymmetric ones. It was first introduced by Saad and Schultz (1986) and it is also based upon the successive minimization of the residual vector over each iteration step. This means that the magnitude of the residual vector decreases

PUC-Rio - Certificação Digital Nº 1612782/CA

monotonically as the iterations progress, and that after m steps the method is assured to converge:

$$\|\boldsymbol{r}_{n+1}\| \leq \|\boldsymbol{r}_n\| \tag{3.23}$$

The problem of minimizing the residual vector is analogous to a least square problem. The algorithm commences with the following objective function

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n\| = \|\boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x}_0 + \mathcal{K}_n)\| = minimum \tag{3.24}$$

Then, the vector belonging to the Krylov subspace that takes the solution vector from $\boldsymbol{x}_0$ to $\boldsymbol{x}_n$ is substituted for the product of an orthonormal[9] matrix $\boldsymbol{V}_n$ of dimension $(m \times n)$, whose columns span the Krylov subspace $\mathcal{K}_n$, and an unknown vector $\boldsymbol{y}_n$

$$\|\boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x}_0 + \boldsymbol{V}_n\boldsymbol{y}_n)\| = \|\boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{V}_n\boldsymbol{y}_n\| = minimum \tag{3.25}$$

Note that the term inside the parenthesis is equivalent to

$$\begin{aligned} \boldsymbol{x}_n &= \boldsymbol{x}_0 + Span\{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n\} \\ &= \boldsymbol{x}_0 + \psi_1\boldsymbol{v}_1 + \psi_2\boldsymbol{v}_2 + \cdots + \psi_n\boldsymbol{v}_n \end{aligned} \tag{3.26}$$

where ψ_n represents the n-th entry of the vector $\boldsymbol{y}_n$.

The columns of the orthogonal matrix $\boldsymbol{V}_n$ can be constructed one at a time by applying a procedure known as the Arnoldi Iteration, which consists of sequentially reducing matrix $\boldsymbol{A}$ to Hessenberg[10] form $\widetilde{\boldsymbol{H}}_n$

$$\boldsymbol{A}\boldsymbol{V}_n = \boldsymbol{V}_{n+1}\widetilde{\boldsymbol{H}}_n \tag{3.27}$$

[9] **Orthonormal Matrix** – A matrix whose columns consist of orthogonal unit vectors; that is, vectors whose norm equals one.

[10] **Hessenberg Matrix** – A nearly triangular matrix that also has the first opposite sub-diagonal filled with nonzero values. A matrix may possess either upper or lower Hessenberg form.

PUC-Rio - Certificação Digital Nº 1612782/CA

where $\widetilde{\boldsymbol{H}}_n$ has the following format

$$\widetilde{\boldsymbol{H}}_n = \begin{bmatrix} h_{11} & \cdots & & h_{1n} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \\ & & h_{n,n-1} & h_{nn} \\ & & & h_{n+1,n} \end{bmatrix} \tag{3.28}$$

This recursive process generates a new column $\boldsymbol{v}_{n+1}$ at each step, which can be computed by the following relationship involving the columns calculated at the previous steps

$$\boldsymbol{A}\boldsymbol{v}_n = h_{1n}\boldsymbol{v}_1 + \cdots + h_{nn}\boldsymbol{v}_n + h_{n+1,n}\boldsymbol{v}_{n+1} \tag{3.29}$$

The Arnoldi Iteration that generates $\boldsymbol{v}_{n+1}$ is ultimately a modified Gram-Schmidt orthogonalization procedure that implements the relationship described in Equation (3.29). As an alternative, orthogonalization could be performed using Householder Triangularization, but this option was not investigated further in this research.

Because even a modified Gram-Schmidt procedure may still suffer from loss of orthogonality due to round-off errors, a test for loss of orthogonality was implemented into the Arnoldi Iteration, as suggested by Kelly (1995). If ever such a loss was detected, the vectors would then be submitted to a re-orthogonalization process. The verification criteria for this correction is presented in Equation (3.30)

$$\|\boldsymbol{A}\boldsymbol{v}_n\| + \varepsilon\|\boldsymbol{v}_{n+1}\| = \|\boldsymbol{A}\boldsymbol{v}_n\| \tag{3.30}$$

where ε represents the numerical tolerance for orthogonalization breakdown. Here this parameter was adopted to be $\varepsilon = 10^{-32}$.

Once the Arnoldi Iteration is complete, the problem is transformed into

$$\left\|\boldsymbol{r}_0 - \boldsymbol{V}_{n+1}\widetilde{\boldsymbol{H}}_n\boldsymbol{y}_n\right\| = minimum \tag{3.31}$$

The first unitary vector $\boldsymbol{v}_1$ may be defined conveniently as

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\|\boldsymbol{r}_0\|} = \frac{\boldsymbol{r}_0}{\beta} \tag{3.32}$$

and the problem thus rewritten to

$$\left\|\beta \boldsymbol{v}_1 - \boldsymbol{V}_{n+1}\widetilde{\boldsymbol{H}}_n \boldsymbol{y}_n\right\| = minimum \tag{3.33}$$

Since both vectors inside the norm are contained within the column space of $\boldsymbol{V}_{n+1}$, multiplying on the left by its transpose does not alter the magnitude of the norm (Trefethen and Bau III, 1997). Given that $\boldsymbol{V}_{n+1}$ is orthonormal, this equates to

$$\left\|\boldsymbol{V}_{n+1}{}^T \beta \boldsymbol{v}_1 - \widetilde{\boldsymbol{H}}_n \boldsymbol{y}_n\right\| = minimum \tag{3.34}$$

which, by construction of $\boldsymbol{V}_{n+1}$ (since $\boldsymbol{v}_1$ is orthogonal to all other $\boldsymbol{v}_i$), is also equivalent to

$$\left\|\beta \boldsymbol{e}_1 - \widetilde{\boldsymbol{H}}_n \boldsymbol{y}_n\right\| = minimum \tag{3.35}$$

where $\boldsymbol{e}_1$ is the canonical vector pointing towards the first direction $(1, 0, 0, \dots)^T$.

Therefore, the challenge is to now find a minimizer $\boldsymbol{y}_n$ that suits Equation (3.35). This is a much less costly task than solving the original problem, because the problem dimensions have now been reduced from $m \times m$ to $(n+1) \times n$, where $n \ll m$.

A simple manner of obtaining this minimizer is applying QR factorization to the Hessenberg matrix, by way of plane rotations (Saad and Schultz, 1986). The idea here is to pre-multiply $\widetilde{\boldsymbol{H}}_n$ by the following rotational matrix $\boldsymbol{F}_n$ at each step of the iteration

$$\boldsymbol{F}_n = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & c_n & -s_n \\ & & s_n & c_n \end{bmatrix} \tag{3.36}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

where $c_n = \cos(\theta_n)$ and $s_n = \sin(\theta_n)$; and where the angle of rotation θ_n is chosen so as to eliminate the new element created in the last row of $\widetilde{\boldsymbol{H}}_n$.

Now, if the sequence of rotations applied throughout the iterative process is defined as

$$\boldsymbol{Q}_n = \boldsymbol{F}_1 \cdot \boldsymbol{F}_2 \ldots \cdot \boldsymbol{F}_n \tag{3.37}$$

then one has that

$$\boldsymbol{Q}_n \widetilde{\boldsymbol{H}}_n = \boldsymbol{R}_n \tag{3.38}$$

where $\boldsymbol{R}_n$ has the following structure

$$\boldsymbol{R}_n = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1,n-1} & r_{1,n} \\ & r_{22} & \cdots & r_{2,n-1} & r_{2,n} \\ & & \ddots & \vdots & \vdots \\ & & & r_{n-1,n-1} & r_{n-1,n} \\ & & & & r_{n,n} \\ 0 & & \cdots & & 0 \end{bmatrix} \tag{3.39}$$

Finally, the problem may be converted into

$$\left\| \boldsymbol{Q}_n \beta \boldsymbol{e}_1 - \boldsymbol{Q}_n \widetilde{\boldsymbol{H}}_n \boldsymbol{y}_n \right\| = \left\| \boldsymbol{g}_n - \boldsymbol{R}_n \boldsymbol{y}_n \right\| = minimum \tag{3.40}$$

where

$$\boldsymbol{g}_n = \boldsymbol{Q}_n \beta \boldsymbol{e}_1 \tag{3.41}$$

represents a transformed right-hand side.

It is important to note that it is not necessary to explicitly recalculate $\boldsymbol{g}_n$ and $\boldsymbol{R}_n$ in their entireties at every iteration, since the latest rotation to be applied, corresponding to the current iteration, will solely affect the final two elements of $\boldsymbol{g}_n$ and the final column of $\boldsymbol{R}_n$. On the other hand, the final column of $\boldsymbol{R}_n$ added at

PUC-Rio - Certificação Digital Nº 1612782/CA

each iteration will also need to be updated with respect to all of the previous rotations (Saad and Schultz, 1986).

At each iteration, if the column being added to $\widetilde{\boldsymbol{H}}_{n+1}$, and consequently to $\boldsymbol{R}_{n+1}$, is represented by

$$\boldsymbol{h}_{n+1} = \begin{bmatrix} r_{1,n+1} \\ r_{2,n+1} \\ \vdots \\ r_{n,n+1} \\ r_{n+1,n+1} \\ h_{n+2,n+1} \end{bmatrix} \tag{3.42}$$

then the rotation necessary to eliminate its last element may be determined by the following definitions of the terms comprising the rotational matrix $\boldsymbol{F}_{n+1}$:

$$c_{n+1} = \frac{r_{n+1,n+1}}{\sqrt{\left(r_{n+1,n+1}\right)^2 + \left(h_{n+2,n+1}\right)^2}} \tag{3.43}$$

and

$$s_{n+1} = \frac{-h_{n+2,n+1}}{\sqrt{\left(r_{n+1,n+1}\right)^2 + \left(h_{n+2,n+1}\right)^2}} \tag{3.44}$$

Once the rotations have been performed, the least square problem takes the format of an upper triangular system, which can consequently be solved without much effort via backwards substitution, by simply disregarding the last row of $\boldsymbol{R}_n$ and the last element of $\boldsymbol{g}_n$. The resulting vector is the minimizer $\boldsymbol{y}_n$, with whom it is possible to subsequently determine the solution vector at the current step

$$\boldsymbol{x}_n = \boldsymbol{x}_0 + \boldsymbol{V}_n \boldsymbol{y}_n \tag{3.45}$$

Afterwards, the residual $\boldsymbol{r}_n$ at that same step could theoretically be computed, to verify if convergence has been reached satisfactorily. However, due to the manner with which $\boldsymbol{y}_n$ is constructed, the norm of the residual is actually available

PUC-Rio - Certificação Digital Nº 1612782/CA

at every iteration at no additional cost. It can be obtained directly as the final element of the vector $\boldsymbol{g}_n$. Therefore, the repetitive calculation of $\boldsymbol{y}_n$, $\boldsymbol{x}_n$ and $\boldsymbol{r}_n$ per iteration step are not required, except once convergence has been successfully achieved, which can be easily established by the value of the $(n+1)$ component of $\boldsymbol{g}_n$.

Furthermore, the only situation where the algorithm may breakdown, which would occur if element $h_{n+1,n}$ equals zero during the Arnoldi process (because then $\boldsymbol{v}_n$ can no longer be built recursively), can be demonstrated to mean that the algorithm has in fact converged. Thus, in this instance $\boldsymbol{x}_n$ may be considered to be the exact solution to the problem (Trefethen and Bau III, 1997).

The final issue to be tackled with the algorithm presented is the fact that, as the process advances, the number of vectors $\boldsymbol{v}_n$ that must be stored increases continuously. Since these vectors have dimension m, the storage cost may become prohibitive if too many iterations are required for convergence, similar to the ORTHOMIN(*k*) case. To remedy this, the key is to reset the method after a pre-determined *k* number of steps and use the latest solution prior to reset $\boldsymbol{x}_k$ as the new initial guess in the next cycle. This leads to the GMRES(*k*) version of the method.

The reset strategy adopted for the GMRES(*k*) method is identical as the one described for ORTHOMIN(*k*), with the exception of two parameters. Here the maximum reset value was altered to $k_{max} = 50$ and the minimum to $k_{min} = 5$. This was due to the results of some test cases performed, that found these higher values to improve the convergence rate of the problems.

The pseudocode of the preconditioned GMRES method implemented is presented in Algorithm 3.2 (Saad and Schultz, 1986; Barrett et al., 1994).

PUC-Rio - Certificação Digital Nº 1612782/CA

Algorithm 3.2 – Preconditioned GMRES Iterative Method.

$\boldsymbol{x}_0 = 0, \quad j = 0, \quad \gamma = 1$

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0, \quad \beta = \|\boldsymbol{M}^{-1}\boldsymbol{r}_0\|, \quad \alpha = \beta$

$Check\ convergence\ (\beta < \eta \cdot \|\boldsymbol{M}^{-1}\boldsymbol{b}\|)$

while $j < iter\_lim$

$\boldsymbol{v}_0 = \frac{\boldsymbol{r}_0}{\beta},\ \boldsymbol{g} = 0,\ \boldsymbol{g}[0] = \beta,\ j = j + 1$

$Test\ quality\ of\ convergence\ (\gamma)\ and\ adjust\ cycle\ size\ (k)\ accordingly$

for $i = 0$ **to** k

$\boldsymbol{v}_{i+1} = \boldsymbol{M}^{-1}(\boldsymbol{A}\boldsymbol{v}_i)$

for $w = 0$ **to** i

$\widetilde{\boldsymbol{H}}[w, i] = \langle \boldsymbol{v}_{i+1} \cdot \boldsymbol{v}_w \rangle$

$\boldsymbol{v}_{i+1} = \boldsymbol{v}_{i+1} - \widetilde{\boldsymbol{H}}[w, i]\boldsymbol{v}_w$

end

$Test\ for\ loss\ of\ orthogonality\ and\ correct\ if\ necessary$

$\widetilde{\boldsymbol{H}}[i + 1, i] = \|\boldsymbol{v}_{i+1}\|$

$\boldsymbol{v}_{i+1} = \frac{\boldsymbol{v}_{i+1}}{\|\boldsymbol{v}_{i+1}\|}$

for $w = 0$ **to** i

$Givens\_Rotation\big(\widetilde{\boldsymbol{H}}[w, i],\ \widetilde{\boldsymbol{H}}[w + 1, i],\ c[i],\ s[i]\big)$

end

$Givens\_Build\big(\widetilde{\boldsymbol{H}}[i, i],\ \widetilde{\boldsymbol{H}}[i + 1, i],\ c[i],\ s[i]\big)$

$Givens\_Rotation\big(\widetilde{\boldsymbol{H}}[i, i], \widetilde{\boldsymbol{H}}[i + 1, i],\ c[i],\ s[i]\big)$

$Givens\_Rotation(\boldsymbol{g}[i],\ \boldsymbol{g}[i + 1],\ c[i],\ s[i])$

$Check\ convergence\ (|\boldsymbol{g}[i + 1]| < \eta \cdot \|\boldsymbol{M}^{-1}b\|)$

end

$\gamma = \frac{|\boldsymbol{g}[i+1]|}{\alpha}$

$\alpha = |\boldsymbol{g}[i + 1]|$

$Least\_Square\_Update\big(\widetilde{\boldsymbol{H}},\ \boldsymbol{g},\ \boldsymbol{V},\ i - 1\big) \rightarrow \boldsymbol{y} \rightarrow \boldsymbol{x}_j$

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_j$

$\beta = \|\boldsymbol{M}^{-1}\boldsymbol{r}_0\|$

$Check\ convergence\ (\beta < \eta \cdot \|\boldsymbol{M}^{-1}\boldsymbol{b}\|)$

end

PUC-Rio - Certificação Digital Nº 1612782/CA

3.2.3
BiCGSTAB Method

The most modern of the iterative methods implemented was the Biconjugate Gradient Stabilized algorithm. It was devised by Henk Van der Vorst (1992) as an improvement over the previously known Biconjugate Gradient (BCG) method by Lanczos, dating from 1952, the Conjugate Gradient Squared (CGS) method by Sonneveld, dating from 1989, the and Quasi-Minimal Residual (QMR) method by Freund and Nachtigal, dating from 1991. It was devised to smooth the oftentimes erratic convergence rate of some of the other methods, and to avoid matrix operations on the transpose of $\boldsymbol{A}$ (Barrett et al., 1994; Sleijpen and Fokkema, 1993).

All of these methods listed are efforts to generalize the CG method for nonsymmetrical matrices and are based on three-term recurrences, which may be considered to be the most powerful nonsymmetrical approach currently available (Trefethen and Bau III, 1997). Further variations of the BiCGSTAB method that were later developed, such as BiCGSTAB2 by Gutknecht (1993) and BiCGSTAB(L) by Sleijpen and Fokkema (1993), were not incorporated to the algorithm in this research. The main reason for this was due to greater focus being devoted to the different preconditioning strategies, rather than to the underlying iterative method.

Biorthogonalization[11] processes are generally based on the Lanczos Iteration, which attempts to reduce a Hermitian matrix to unitary, tridiagonal form. However, when dealing with matrices that are not Hermitian in nature, this reduction is normally not achievable to full extent; requiring a compromise between either keeping the unitary transformations or the tridiagonal form. The Hessenberg orthogonalization of GMRES does the former (resulting in the Arnoldi Iteration), whereas biorthogonalization procedures rely on the latter, leading to a non-unitary tridiagonal biorthogonalization (Trefethen and Bau III, 1997):

$$\boldsymbol{A = VTV^{-1}} \tag{3.46}$$

[11] **Biorthogonal Matrix** – A matrix V whose columns are not orthogonal to each other, but are instead orthogonal to the columns of $(V^{-1})^T$.

PUC-Rio - Certificação Digital Nº 1612782/CA

or, when $n < m$,

$$\boldsymbol{A}\boldsymbol{V}_n = \boldsymbol{V}_{n+1}\widetilde{\boldsymbol{T}}_n \tag{3.47}$$

where the columns of $\boldsymbol{V}$ are orthogonal to the columns of $(\boldsymbol{V}^{-1})^T$ – even though they are not unitary; and $\widetilde{\boldsymbol{T}}_n$ is a tridiagonal matrix.

Similarly, we have

$$\boldsymbol{A}^T\boldsymbol{W}_n = \boldsymbol{W}_{n+1}\tilde{\boldsymbol{S}}_n \tag{3.48}$$

where $\tilde{\boldsymbol{S}}_n$ is tridiagonal; and

$$\boldsymbol{W} = (\boldsymbol{V}^{-1})^T \tag{3.49}$$

Analogously to GMRES, the new vectors $\boldsymbol{v}_{n+1}$ and $\boldsymbol{w}_{n+1}$ that comprise the matrices in Equation (3.47) and Equation (3.48) may be obtained recursively, but now limited to just three-terms (as opposed to the entire sequence of terms that came before it)

$$\boldsymbol{A}\boldsymbol{v}_n = \gamma_{n-1}\boldsymbol{v}_{n-1} + \alpha_n\boldsymbol{v}_n + \beta_n\boldsymbol{v}_{n+1} \tag{3.50}$$

and

$$\boldsymbol{A}\boldsymbol{w}_n = \beta_{n-1}\boldsymbol{w}_{n-1} + \alpha_n\boldsymbol{w}_n + \gamma_n\boldsymbol{w}_{n+1} \tag{3.51}$$

where α_n are the coefficients in the main diagonal of $\widetilde{\boldsymbol{T}}_n$ and $\tilde{\boldsymbol{S}}_n$; β_n are the coefficients in the first lower diagonal of $\widetilde{\boldsymbol{T}}_n$ and first upper diagonal of $\tilde{\boldsymbol{S}}_n$; and γ_n are the coefficients in the first upper diagonal of $\widetilde{\boldsymbol{T}}_n$ and first lower diagonal of $\tilde{\boldsymbol{S}}_n$.

Thus, the vectors formed in the process will belong to the two Krylov subspaces specified in Equations (3.52) – (3.53)

$$\boldsymbol{v}_n \in Span\{\boldsymbol{v}_1, \boldsymbol{A}\boldsymbol{v}_1, \dots, \boldsymbol{A}^{n-1}\boldsymbol{v}_1\} \tag{3.52}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

and

$$\boldsymbol{w}_n \in Span\{\boldsymbol{w}_1, \boldsymbol{A}^T\boldsymbol{w}_1, \dots, (\boldsymbol{A}^T)^{n-1}\boldsymbol{w}_1\} \tag{3.53}$$

Contrary to the procedure applied in GMRES, in which the basis vectors were forcibly made to be orthogonal via Gram-Schmidt orthogonalization, the process described by Equations (3.50) – (3.51) depends on the orthogonality of the basis vectors arising automatically. In practice, this may not occur due to the accumulation of rounding errors over time and the method may be prone to stagnation and numerical breakdown. This occurs, for example, if $\boldsymbol{v}_n = 0$ or $\boldsymbol{w}_n = 0$ at some step (Trefethen and Bau III, 1997).

The process that was just introduced describes the concept behind the BCG method. The stabilized version of this biorthogonalization adjusts the recursive procedure presented, so as not to involve calculations with the transpose of matrix $\boldsymbol{A}$.

The complete derivation of BiCGSTAB is not as straightforward as those of the previous algorithms and shall not be performed in full. Nevertheless, perhaps just as valuable is its comparison to the more easily comprehended CG algorithm. Considering the CG method, in which the solution is updated via

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n \tag{3.54}$$

and the residual vector via

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{A}\boldsymbol{p}_n \tag{3.55}$$

where $\boldsymbol{p}_n$ represents a search vector belonging to the Krylov subspace; and α_n represents the step-length (calculated in a manner so that the residual vectors $\boldsymbol{r}_{n+1}$ and $\boldsymbol{r}_n$ are orthogonal to each other), in BiCGSTAB these relations can be rewritten as

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \omega_n \boldsymbol{s}_n \tag{3.56}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

and

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{A}\boldsymbol{p}_n - \omega_n \boldsymbol{A}\boldsymbol{s}_n \tag{3.57}$$

where there are now two search vectors $\boldsymbol{p}_n$ and $\boldsymbol{s}_n$; and where α_n and ω_n are their respective step-length coefficients. However, here $\boldsymbol{s}_n$ can also be considered to represent an intermediate version of the residual vector. It is easy to see this if one compares its definition

$$\boldsymbol{s}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{A}\boldsymbol{p}_n \tag{3.58}$$

with the definition of the residual $\boldsymbol{r}_{n+1}$ given in Equation (3.55), for the CG algorithm. This definition of $\boldsymbol{s}_n$ also leads to the following relationship

$$\boldsymbol{r}_{n+1} = \boldsymbol{s}_{n+1} - \omega_n \boldsymbol{A}\boldsymbol{s}_n \tag{3.59}$$

Therefore, the coefficient ω_n can be seen, additionally, as a parameter to smooth the convergence rate by minimizing the residual norm. It is calculated so as to take the steepest descent step in the direction of the intermediate residual $\boldsymbol{s}_n$ (Saad, 2003).

Finally, the search direction vector in CG is updated via

$$\boldsymbol{p}_{n+1} = \boldsymbol{r}_{n+1} + \beta_n \boldsymbol{p}_n \tag{3.60}$$

where β_n may be calculated based on the fact (also valid for BiCGSTAB) that $\boldsymbol{p}_{n+1}$ is constructed orthogonal to the current projection vector $\boldsymbol{A}\boldsymbol{p}_n$ – meaning that they are $\boldsymbol{A}$-conjugate by definition:

$$\boldsymbol{p}_{n+1}{}^T \boldsymbol{A}\boldsymbol{p}_n = 0 \tag{3.61}$$

Thus, its definition in CG becomes

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\beta_n = \frac{\boldsymbol{r}_{n+1}{}^T \cdot \boldsymbol{r}_{n+1}}{\boldsymbol{r}_n{}^T \cdot \boldsymbol{r}_n} \tag{3.62}$$

where β_n may be seen to represent the improvement in the residual reduction obtained in the current step (since it is the ratio of the current and previous norms).

Comparatively, BiCGSTAB updates the search direction vectors via

$$\boldsymbol{p}_{n+1} = \boldsymbol{r}_{n+1} + \beta_n \boldsymbol{p}_n - \beta_n \omega_n \boldsymbol{A}\boldsymbol{p}_n \tag{3.63}$$

and, as stated in Equation (3.58), via

$$\boldsymbol{s}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{A}\boldsymbol{p}_n \tag{3.64}$$

Consequently, the definition of β_n is somewhat altered to

$$\beta_n = \frac{\tilde{\rho}_{n+1}}{\tilde{\rho}_n} \cdot \frac{\alpha_n}{\omega_n} \tag{3.65}$$

which includes the previously identified parameters α_n and ω_n, as well as the term $\frac{\tilde{\rho}_{n+1}}{\tilde{\rho}_n}$, which from the definition of $\tilde{\rho}_n$

$$\tilde{\rho}_n = \boldsymbol{r}_n{}^T \cdot \boldsymbol{r}_0 = \langle \boldsymbol{r}_n \cdot \boldsymbol{r}_0 \rangle \tag{3.66}$$

may also be seen to represent an improvement in residual reduction, similarly to CG (Saad, 2003; and Yuvashankar et al., 2016).

The pseudocode of the preconditioned BiCGSTAB method implemented is presented in Algorithm 3.3 (Barrett et al., 1994).

PUC-Rio - Certificação Digital Nº 1612782/CA

Algorithm 3.3 – Preconditioned BiCGSTAB Iterative Method.

$\boldsymbol{x}_0 = 0, \qquad \rho_0 = 1,$

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$

$\hat{\boldsymbol{r}}_0 = \boldsymbol{r}_0, \qquad \boldsymbol{p}_1 = \boldsymbol{r}_0$

for $i = 1$ **to** $iter\_max$

$\rho_i = \langle \hat{\boldsymbol{r}}_0 \cdot \boldsymbol{r}_{i-1} \rangle$

$Test\ |\rho_i|\ for\ numerical\ breakdown$

if $i > 1$

$\beta_i = \frac{\rho_i}{\rho_{i-1}} \cdot \frac{\alpha_{i-1}}{\omega_{i-1}}$

$\boldsymbol{p}_i = \boldsymbol{r}_{i-1} + \beta(\boldsymbol{p}_{i-1} - \omega_{i-1}\boldsymbol{u}_{i-1})$

end

$\boldsymbol{u}_i = \boldsymbol{A}(\boldsymbol{M}^{-1}\boldsymbol{p}_i)$

$\alpha_i = \frac{\rho_i}{\langle \hat{\boldsymbol{r}}_0 \cdot \boldsymbol{u}_i \rangle}$

$\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \alpha_i(\boldsymbol{M}^{-1}\boldsymbol{p}_i)$

$\boldsymbol{s}_i = \boldsymbol{r}_{i-1} - \alpha_i \boldsymbol{u}_i$

$Check\ convergence\ (\|\boldsymbol{s}_i\| < \eta \cdot \|\boldsymbol{b}\|)$

$\boldsymbol{t}_i = \boldsymbol{A}(\boldsymbol{M}^{-1}\boldsymbol{s}_i)$

$\omega_i = \frac{\langle \boldsymbol{t}_i \cdot \boldsymbol{s}_i \rangle}{\langle \boldsymbol{t}_i \cdot \boldsymbol{t}_i \rangle}$

$Test\ |\omega_i|\ for\ numerical\ breakdown$

$\boldsymbol{x}_i = \boldsymbol{x}_i + \omega_i(\boldsymbol{M}^{-1}\boldsymbol{s}_i)$

$\boldsymbol{r}_i = \boldsymbol{s}_i - \omega_i \boldsymbol{t}_i$

$Check\ convergence\ (\|\boldsymbol{r}_i\| < \eta \cdot \|\boldsymbol{b}\|)$

end

PUC-Rio - Certificação Digital Nº 1612782/CA

4
Preconditioners

Preconditioners form an integral part of any efficient numerical *solver* designed to handle linear systems of equations of considerable size. State-of-the-art preconditioners may be of such complexity that their processing time may even surpass that of the iterative method with which they are coupled (Trefethen and Bau III, 1997). However, if well designed, they are capable of significantly reducing the total time required to solve some of the most challenging problems encountered. In many instances, their use may be the only manner available to reach a precise enough solution to a specific problem. Hence, the robustness of the solution method may actually be much more dependent on the quality of the preconditioner than on the Krylov projection method employed (Saad, 2003).

As previously explained, the convergence rate of an iterative method depends on some intrinsic properties of the coefficient matrix, such as the spectrum of its singular values or its eigenvalues. The concept behind preconditioning is that of transforming the original linear system into an equivalent one that has a more favorable spectrum. Consequently, the solution of this equivalent system will be identical to that of the original system, but should be more easily attained (Barrett et al., 1994).

PUC-Rio - Certificação Digital Nº 1612782/CA

4.1
Incorporation of the Preconditioner into the Iterative Method

There are several forms of applying a preconditioner to a matrix equation. The most straightforward one, named left preconditioning, involves the pre-multiplication of both sides of the original equation by the inverse of the preconditioning matrix

$$\boldsymbol{M^{-1}Ax = M^{-1}b} \tag{4.1}$$

where $\boldsymbol{M}$ represents the preconditioning matrix.

Analyzing the transformed Equation (4.1), it should be clear that the convergence behavior of any iterative method used to solve it will now depend on the properties of $\boldsymbol{M^{-1}A}$, instead of just $\boldsymbol{A}$. The ideal situation, in terms of improving the matrix conditioning, would be the case where $\boldsymbol{M = A}$, because then $\boldsymbol{M^{-1}A = I}$, and the condition number would equal to $\kappa = 1$. Unfortunately, in this case calculating $\boldsymbol{M^{-1}A}$ would be as hard as solving the original problem. Alternatively, if $\boldsymbol{M = I}$, then calculating $\boldsymbol{M^{-1}A}$ would prove to be trivial, but the condition number would remain unchanged. Therefore, determining a suitable $\boldsymbol{M}$ involves finding a matrix between these extremes – that is, one that offers a favorable compromise between being a reasonable approximation of $\boldsymbol{A}$ and having an inverse that is not too costly to obtain. Overall, it is common for $\boldsymbol{M}$ to be derived in some way from the original coefficient matrix $\boldsymbol{A}$ (Trefethen and Bau III, 1997).

An alternative form of preconditioning, denoted as right preconditioning, takes the form

$$\boldsymbol{AM^{-1}Mx = b} \tag{4.2}$$

which may be rewritten as

$$\boldsymbol{AM^{-1}y = b} \tag{4.3}$$

with

$$\boldsymbol{x = M^{-1}y} \tag{4.4}$$

This option is usually the preferred one, due to the fact that it does not alter the residual vector to be minimized in the iterative method and, consequently, does not require the convergence criteria to be modified (Saad, 2003; and Al-Shaalan et al., 2009). This can be seen in Equation (4.5) for the right preconditioned case

$$\boldsymbol{r_n = b - AM^{-1}Mx = b - Ax} \tag{4.5}$$

which in the left preconditioned case would otherwise become

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\boldsymbol{r}_n = \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \tag{4.6}$$

Finally, even further preconditioning options are available, such as a combination of the previous ones into a two-stage approach. If we now consider

$$\boldsymbol{M} = \boldsymbol{M}_L \boldsymbol{M}_R \tag{4.7}$$

the equation would then be transformed into

$$\boldsymbol{M}_L{}^{-1}\boldsymbol{A}\boldsymbol{M}_R{}^{-1}\boldsymbol{M}_R\boldsymbol{x} = \boldsymbol{M}_L{}^{-1}\boldsymbol{b} \tag{4.8}$$

The direct application of the preconditioner in any of these forms – for example, by converting the matrix equation into

$$\widetilde{\boldsymbol{A}} \cdot \widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}} \tag{4.9}$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{M}_L{}^{-1}\boldsymbol{A}\boldsymbol{M}_R{}^{-1}$; $\widetilde{\boldsymbol{x}} = \boldsymbol{M}_R\boldsymbol{x}$; and $\widetilde{\boldsymbol{b}} = \boldsymbol{M}_L{}^{-1}\boldsymbol{b}$, and then solving for $\widetilde{\boldsymbol{x}}$ (and subsequently for $\boldsymbol{x}$), is usually not necessary, nor is it practicable. It is preferable instead to transform the original versions of the iterative algorithms into alternate, preconditioned versions, where the preconditioner is introduced in the fashion previously mentioned

$$\boldsymbol{M}^{-1}\boldsymbol{r} = \boldsymbol{q} \tag{4.10}$$

as recurring steps throughout the iterative process. An example demonstrating a complete derivation of this type of transformation can be found in Golub and Van Loan's Matrix Computations (1996). The algorithms exhibited in Chapter 3 already had the preconditioning step from Equation (4.10) embedded into them, as explained by Barrett et al. (1994).

This form of preconditioning is of particular importance because often neither matrix $\boldsymbol{M}$ nor its inverse $\boldsymbol{M}^{-1}$ are ever actually built. In fact, Equation (4.10) should not be interpreted as involving the actual multiplication of the inverse of $\boldsymbol{M}$ by a

PUC-Rio - Certificação Digital Nº 1612782/CA

vector $\boldsymbol{r}$, but understood as the solution of a new system of equations ($\boldsymbol{Mq} = \boldsymbol{r}$), whose result will be obtained by exploiting special characteristics of $\boldsymbol{M}$. Furthermore, this latter strategy completely avoids matrix-matrix operations, such as the ones that would occur if right or left preconditioning were performed directly, and which are very costly to compute.

Ideally, solving a system of the form of Equation (4.10) several times in the course of an iterative method should be as inexpensive as possible. Ultimately, it will be a compromise that involves both the costs of constructing the preconditioner during an initialization phase as well as applying it at every iteration, versus a reduction in the number of total iteration steps required for the convergence of the iterative method. The construction cost varies widely depending on the preconditioning method (as does the impact it might have on the number of iterations), while its application normally leads to an increase in the work count by a constant multiplicative factor per iteration (Barrett et al., 1994).

PUC-Rio - Certificação Digital Nº 1612782/CA

4.2
Survey of Preconditioners

The variety of preconditioners available is very diverse, ranging from incredibly simple to extremely complex configurations. Their strength, in terms of being able to solve difficult problems, will normally vary in a corresponding fashion. This section shall present some of the most commonly used methods (Trefethen and Bau III, 1997).

The most basic preconditioner is named Jacobi, and consists of building $\boldsymbol{M}$ from the main diagonal of the coefficient matrix

$$\boldsymbol{M} = diag(\boldsymbol{A}) \tag{4.11}$$

This has the significant convenience of requiring the storage of just m additional numbers, the equivalent of a single vector in $\mathcal{R}^m$. Moreover, the construction of $\boldsymbol{M}^{-1}$ is quite straightforward – its structure is also that of a diagonal matrix, whose entries are the reciprocals of their corresponding entries in $\boldsymbol{M}$.

Furthermore, a variant version exists for matrices that contain blocks of non-zeros (Barrett et al., 1994). This is the case of the coefficient matrix arising in the

simulation of multiphase flow, where there are several variables per grid block, which may be grouped together. In this case, matrix $\boldsymbol{M}$ becomes a block diagonal matrix, where each block may be seen as a submatrix within $\boldsymbol{M}$. The entries of $\boldsymbol{M}^{-1}$ are then equal to the inverse of their corresponding submatrices in $\boldsymbol{M}$. For two or three-phase problems these are still relatively cheap calculations.

This preconditioner was implemented during the course of this research. However, as was expected, its results were very poor, due to the strong ill-conditioned nature of the matrix equations in fully implicit multiphase simulations. Therefore, it will not feature in the comparisons displayed in the following chapter nor will it be further examined. The next techniques discussed, on the other hand, were not implemented and are listed here for historical purpose.

A second option of preconditioner involves performing one or more steps of one of the stationary methods, such as Jacobi, Gauss-Seidel or SSOR. For example, the preconditioning matrix based on SSOR may be written as

$$\boldsymbol{M} = (\boldsymbol{D} + \omega\boldsymbol{E}) \cdot \boldsymbol{D}^{-1} \cdot (\boldsymbol{D} + \omega\boldsymbol{F}) \tag{4.12}$$

where $\boldsymbol{E}$ represents the lower triangular part of $\boldsymbol{A}$, without the main diagonal; $\boldsymbol{F}$ represents the upper triangular part of $\boldsymbol{A}$, without the main diagonal; $\boldsymbol{D}$ represents the main diagonal; and ω is the relaxation parameter. The preconditioning steps of the algorithms can then be implemented in two stages

$$(\boldsymbol{I} + \omega\boldsymbol{E}\boldsymbol{D}^{-1}) \cdot \boldsymbol{z} = \boldsymbol{r} \tag{4.13}$$

$$(\boldsymbol{D} + \omega\boldsymbol{F}) \cdot \boldsymbol{q} = \boldsymbol{z} \tag{4.14}$$

where the matrix structure in Equations (4.13) – (4.14) renders them easy to solve (Saad, 2003).

Another class of preconditioners that is likewise related to one of the iterative methods mentioned in Chapter 3 is called Dimensional Splitting.

In addition, direct methods may also be applied as preconditioners, for example, via Gaussian Elimination without pivoting. The idea here would be to solve the system relatively fast, but without much concern for precision, that is, in

PUC-Rio - Certificação Digital Nº 1612782/CA

an unstable manner; then use the result as input for the iterative method (Trefethen and Bau III, 1997).

Other available options that will not be described in detail, but that are listed here for the sake of completeness, include: Lower-Order Discretization; Domain Decomposition; Coarse-Grid Approximation; Local Approximation; Symmetric Approximation; Splitting a Multi-Term Operator; Periodic Approximation; and Polynomial Preconditioners (Trefethen and Bau III, 1997).

4.3
Implemented Preconditioners

In the next sections will be presented the algorithms that were implemented in this research, namely: (i) ILU; (ii) Nested Factorization; and (iii) Constrained Pressure Residual. The first two belong to a class of methods called Approximate Factorization. One algorithm of this kind, SIP, was also discussed in the previous chapter. The third and final method is also the more complex one, and combines the techniques of Splitting the Problem Variables and Coarse-Grid Approximation, as shall be detailed further ahead. These three methods are the ones that have been most commonly implemented in recent reservoir simulators (Hammersley and Ponting, 2008; Gries et al., 2013; SPE – Reservoir Simulation Linear Equation Solver).

PUC-Rio - Certificação Digital Nº 1612782/CA

4.3.1
ILU Preconditioner

Approximate factorizations schemes consist of determining factors of a matrix that do not perfectly decompose it. These factors should be easier to build than the exact ones and should keep some of the advantageous characteristics of the original matrix, such as sparsity, while also serving as a reasonable approximation to it. To maintain sparsity, some or all of the new non-zero elements that would have been created during the factorization process, denoted *fill*, are discarded from the factor matrices. The degree of *fill-in* permitted to occur during the factorization is subject to user choice and will impact both the cost of the process, as well as how closely the factors represent the original matrix. To be worthwhile, carrying additional non-zeros must be compensated by faster convergence of the iterative

method. In general terms, constructing the approximate factors is a costly procedure and often will equal that of one or more iterations of the method (Barrett et al., 1994).

Perhaps the most famous preconditioning method, and the one that made the preconditioning technique popular, is the Incomplete LU Factorization (ILU) (Trefethen and Bau III, 1997). It is the equivalent of the Incomplete Cholesky Factorization (IC) for nonsymmetric systems of equations, and was first introduced by Meijerink and Van der Vorst (1977), who also proved that the factors are guaranteed to exist for *M-Matrices*[12]. It entails performing Gaussian Elimination on the coefficient matrix to create both an upper triangular factor and a lower triangular one, meanwhile limiting the amount of fill-in that would normally be generated by this process.

Once the factors have been built, they can be used to solve the preconditioning steps in the iterative method. This is exemplified next:

$$\boldsymbol{A \approx \tilde{L}\tilde{U} = M} \tag{4.15}$$

where $\boldsymbol{A \neq M}$. Consequently, Equation (4.10) equals

$$\boldsymbol{\tilde{L}\tilde{U}q = r} \tag{4.16}$$

which may be solved easily in the following two steps

$$\boldsymbol{\tilde{L}y = r} \tag{4.17}$$

$$\boldsymbol{\tilde{U}q = y} \tag{4.18}$$

due to the triangular structure of the $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ matrices.

An alternative form of factoring the coefficient matrix would be via the use of three matrices (Barrett et al., 1994), an upper triangular one, a diagonal one and finally a lower triangular one, such that

[12] **M-Matrix** – A Z-Matrix (one containing positive diagonal and negative off-diagonal entries) whose eigenvalues have nonnegative real parts.

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\boldsymbol{M} = (\boldsymbol{D} + \boldsymbol{L})\boldsymbol{D}^{-1}(\boldsymbol{D} + \boldsymbol{U}) = (\boldsymbol{D} + \boldsymbol{L}) \cdot (\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{U}) \tag{4.19}$$

In this case, the preconditioner could be applied as follows

$$(\boldsymbol{D} + \boldsymbol{L}) \cdot \boldsymbol{y} = \boldsymbol{r} \tag{4.20}$$

$$(\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{U}) \cdot \boldsymbol{q} = \boldsymbol{y} \tag{4.21}$$

Meijerink (1983) and Behie et al. (1984) also proposed a block procedure for the ILU algorithm using this alternative form, in which the matrix operations would be performed on entire blocks, instead of the individual entries. Nevertheless, neither this three-term factorization nor the block treatment were implemented in this research.

The simplest form of ILU, denoted ILU(0), involves not allowing for any fill-in to be introduced; which means that the non-zero pattern of $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ correspond exactly to that of $\boldsymbol{A}$, here represented as $NZ(\boldsymbol{A})$. This factorization can be performed in different manners, depending on the way upon which the matrix is swept during the execution of the algorithm. The pseudocode presented in Algorithm 4.1 for constructing the incomplete factors considers the version known as IKJ (Saad, 2003). In practice, only one matrix is actually built, with all the non-zero elements.

From this base algorithm several variants are possible. For example, during the construction of the factors, all non-zero terms arising in positions that did not previously belong to $NZ(\boldsymbol{A})$ were simply disregarded. Alternatively, it is possible to partially compensate the exclusion of these terms by quantifying the total amount discarded per row, that is, by adding all its dropped terms, and then subtracting this sum from the corresponding diagonal entry of the preconditioner matrix. This variation is denoted Modified ILU Factorization (MILU). Here the number of non-zeros to be stored remains unchanged (Saad, 2003).

PUC-Rio - Certificação Digital Nº 1612782/CA

Algorithm 4.1 – ILU Preconditioner.

```
for i = 2, 3, ..., m
    for k = 1, ..., i − 1
        if (i, k) ∈ NZ(A)
            a_ik = a_ik / a_kk
        end
        for j = k − 1, ..., m
            if (i, j) ∈ NZ(A)
                a_ij = a_ij − a_ik a_kj
            end
        end
    end
end
```

PUC-Rio - Certificação Digital Nº 1612782/CA

Another possibility is related to the concept of fill-in previously mentioned. It involves allowing for non-zeros to appear in locations not belonging to $NZ(\boldsymbol{A})$. There exist two types of strategies for accepting fill-in; the first is based on the structure of the non-zeros, while the second is based on a numerical tolerance for the elements created. The structural option introduces the concept of *level of fill*, defined for each element (i, j) in matrix $\boldsymbol{A}$ as

$$level_{ij} = \begin{cases} 0 & \text{if} \quad a_{ij} \neq 0, \text{or } i = j \\ \infty & \text{otherwise} \end{cases} \tag{4.22}$$

Each time an element a_{ij} is calculated in Algorithm 4.1, its level of fill must also be updated by the following rule:

$$level_{ij} = min\{level_{ij};\ max\{level_{ik};\ level_{ki}\} + 1\} \tag{4.23}$$

Subsequently, the elements are updated not based on whether they belong to $NZ(\boldsymbol{A})$, as was the case before, but instead on whether their level of fill does not exceed a specified parameter p. Consequently, in ILU(p) all elements with fill level

below or equal to p are kept, while the remaining ones are dropped. The particular case $p = 0$ leads to the same algorithm previously presented. This method poses the disadvantage of often demanding high computational effort, due to the additional calculations required and to the continuous work of updating the fill levels (Saad, 2003).

The second strategy mentioned determines that all elements whose magnitude is smaller than a certain threshold value shall be dropped from the incomplete factors. This concept makes greater mathematical sense, in that it tends to keep only the most relevant entries (those with most significant physical meaning). However, it has the hindrance that it does not limit beforehand the amount of memory to be occupied by the preconditioner, since the number of fill-ins to be introduced is difficult to predict in advance (Barrett et al., 1994).

An option to circumvent this drawback is to establish a maximum number of fill-ins per row. This approach leads to the ILUT(p, τ) algorithm. It involves keeping only the p largest elements in each row of $\tilde{\boldsymbol{L}}$ and the p largest elements in each row of $\widetilde{\boldsymbol{U}}$ that are above the tolerance threshold τ_i, defined as

$$\tau_i = \tau \cdot \|row(i)\| = \tau \cdot \|a_{i*}\| \tag{4.24}$$

where τ represents a relative threshold tolerance.

Analogous to the ILU(p) algorithm, ILUT(p, τ) will also be equivalent to ILU(0) whenever both $p = 0$ and $\tau = 0$. The pseudocode for building ILUT is presented in Algorithm 4.2, in which w is a full-length working vector used to perform computations on a given row and w_k is its k-th entry (Saad, 2003).

As with any preconditioner, the more closely the $\boldsymbol{L}$ and $\boldsymbol{U}$ factors approximate $\boldsymbol{A}$ the better they should function in aiding convergence. It is to be expected that, as more non-zeros are maintained, fewer iterations would be needed to solve the linear system. On the other hand, the cost of setting-up the preconditioner and of executing the preconditioning step during each iteration will also increase correspondingly. Hence, the possible gains to be had with more accurate factorizations may be offset by the cost of determining and operating with the factors themselves (Saad, 2003).

PUC-Rio - Certificação Digital Nº 1612782/CA

Algorithm 4.2 – ILUT Preconditioner.

for $i = 1, \dots, m$
    $w = a_{i*}$
    for $k = 1, \dots, i-1$ **and** $w_k \neq 0$
        $w_k = w_k / a_{kk}$
        $Drop\ w_k\ if\ below\ threshold\ tolerance\ \tau_i$
        if $w_k \neq 0$
            $w = w - w_k * u_{k*}$
        end
    end
    $Drop\ elements\ of\ w\ below\ threshold\ tolerance\ \tau_i$
    $Keep\ only\ p\ largest\ elements\ on\ L\ and\ U\ parts$
    $l_{ij} = w_j$ **for** $j = 1, \dots, i-1$
    $u_{ij} = w_j$ **for** $j = i, \dots, m$
end

For this thesis, the ILU(0), MILU and ILUT variants were implemented in a first moment. To limit the total combinations of preconditioners and iterative methods to be eventually tested for performance, a preliminary test was completed to compare ILU(0), MILU and ILUT(5, 10^{-4}). The choice of parameters for ILUT were based upon values presented by Saad (2003), since no theory exists establishing optimal criteria for their selection.

The preliminary tests performed used a set of five matrix equations available from the Matrix Market (*https://math.nist.gov/MatrixMarket/*). This Market is a repository of sparse matrix data originating from a variety of applications and presented in triplet format (non-zero entries listed alongside their addresses, by column and then row order). Information on the matrices contained in the multiple sets and directions on how to obtain them are described in the guide presented by Duff et al. (1992).

Part of this matrix database is named the Harwell-Boeing Collection, which contains the SHERMAN set of equations, which are related to oil reservoir simulation. The characteristics of the problems are presented in Table 4.1:

PUC-Rio - Certificação Digital Nº 1612782/CA

Table 4.1 – Summary of reservoir simulation problems from the SHERMAN set.

Problem #	Grid Dimensions	Number of Components	Matrix Dimension	Number of Non-Zeros	Simulator Formulation
1	10 x 10 x 10	1	1000 x 1000	3750	Black Oil
2	6 x 6 x 5	5	1080 x 1080	23094	Thermal
3	35 x 11 x 13	1	5005 x 5005	20033	IMPES
4	16 x 23 x 3	1	1104 x 1104	3786	IMPES
5	16 x 23 x 3	3	3312 x 3312	20793	FIM Black Oil

Results from solving these matrix equations indicated that the MILU method was not effective for this class of problems, with two of the five problems not reaching convergence. This same conclusion had been observed before by Behie (1985).

Both ILU(0) and ILUT(5, 10^{-4}) were successful in solving all five problems. Overall, ILUT(5, 10^{-4}) was capable of delivering solutions in fewer iterations, sometimes with shorter computational times. However, the time required for its construction was also consistently higher, and did not scale well with problem size. Therefore, from the class of ILU preconditioners, the ILU(0) version was the one chosen for the subsequent tests to be performed, and henceforth it shall be referred to simply as ILU.

Furthermore, these problems were also used to validate the correctness of the implemented versions of the BiCGSTAB, GMRES and ILU algorithms. This was done by solving the five reservoir scenarios described in Table 4.1 using the codes projected for the reservoir simulator, and comparing the results with the ones obtained using the software MATLAB (the remaining algorithms implemented do not have built-in functions in MATLAB for comparison) (MATLAB, 2019). The analysis also involved examining the solution vectors obtained by the reservoir code and by MATLAB for each *solver* configuration (BiCGSTAB with ILU and GMRES with ILU) to establish their equivalence, as well as verifying whether the number of iterations required for convergence were similar. The results of these analyses are presented in Table 4.2. They show that the algorithms implemented for the reservoir simulator appear to be compatible with the versions available in

PUC-Rio - Certificação Digital Nº 1612782/CA

MATLAB. The improved performance seen in the GMRES method implemented for the simulator can most likely be attributed to the variable reset strategy adopted.

Table 4.2 – Iteration count comparison between the reservoir code and MATLAB.

	Reservoir Code: Iteration Count		MATLAB: Iteration Count	
Problem #	BiCGSTAB	GMRES(*k*)	BiCGSTAB	GMRES(*k*)
1	39	57	39	70
2	8	15	8	15
3	99	166	122	226
4	28	38	28	54
5	27	38	27	54

4.3.2 Nested Factorization Preconditioner

The second preconditioner implemented is called Nested Factorization, and it also belongs to the class of Approximate Factorization methods. It was first proposed by Appleyard and Cheschire (1983) and soon became extremely popular in the field of reservoir simulation due to its superior performance in several reservoir scenarios. It is considered one of the most robust techniques available (Mattax and Dalton, 1990). Contrary to other factorization schemes, such as IC or ILU, Nested Factorization does not explicitly build the factor matrices, nor are the factors structured strictly as triangular matrices. Instead, it only assembles a single diagonal matrix during initialization and afterwards, during the preconditioning steps, sequentially builds lower and upper factors by adding one dimension at a time to the preconditioner (Appleyard and Cheschire, 1983).

Because this procedure was created specifically for the purpose of solving reservoir problems, the factorization process was developed so as to assure material balance for each phase present. This is based on the fact that the sum of the elements of vector $\boldsymbol{b}$ formed during each nonlinear iteration equals the net rate of mass accumulation for that time-step. Consequently, the error in material balance may be obtained from the sum of the elements of the residual vector $\boldsymbol{r}$ during the linear iterations:

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0 \tag{4.25}$$

If the initial solution is established from

$$\boldsymbol{M}\boldsymbol{x}_0 = \boldsymbol{b} \tag{4.26}$$

then,

$$\boldsymbol{r}_0 = (\boldsymbol{M} - \boldsymbol{A})\boldsymbol{x}_0 \tag{4.27}$$

From Equation (4.27) it is possible to deduce that if the sum of each column of $\boldsymbol{M}$ and $\boldsymbol{A}$ are identical, then the sum of the elements of $\boldsymbol{r}_0$ will equate to zero, assuring material balance (Cheshire et al., 1980). Thus, $\boldsymbol{M}$ is built so that

$$colsum(\boldsymbol{M}) = colsum(\boldsymbol{A}) \tag{4.28}$$

where $colsum(\boldsymbol{A})$ represents a block diagonal matrix formed by the sum of the block column elements of matrix $\boldsymbol{A}$.

It can be further shown that if $\boldsymbol{r}_0$ is forced to have zero sum, then the subsequent residuals $\boldsymbol{r}_n$ – resulting from the remaining iteration steps – will likewise sum to zero.

In three-dimensional problems the coefficient matrix consists of seven diagonal bands, and may be decomposed in the manner shown in Equation (4.29)

$$\boldsymbol{A} = (\boldsymbol{D} + \boldsymbol{L}_1 + \boldsymbol{U}_1) + (\boldsymbol{L}_2 + \boldsymbol{U}_2) + (\boldsymbol{L}_3 + \boldsymbol{U}_3) \tag{4.29}$$

Here the outer bands $\boldsymbol{L}_3$ and $\boldsymbol{U}_3$ represent the interactions between different planes, while the central diagonals contain the interactions within planes. On the next level, inside the central diagonals, $\boldsymbol{L}_2$ and $\boldsymbol{U}_2$ represent the interactions between different lines, while the next central diagonals contain the interactions within lines. Finally, inside these central diagonals, $\boldsymbol{L}_1$ and $\boldsymbol{U}_1$ represent the interactions between cells, while the final central diagonal $\boldsymbol{D}$ contains information

PUC-Rio - Certificação Digital Nº 1612782/CA

pertaining to the individual cells. This nested tridiagonal structure is exploited by constructing the preconditioner in the following sequence:

$$\boldsymbol{M} = (\boldsymbol{P} + \boldsymbol{L}_3) \cdot (\boldsymbol{I} + \boldsymbol{P}^{-1}\boldsymbol{U}_3) \tag{4.30}$$

$$\boldsymbol{P} = (\boldsymbol{T} + \boldsymbol{L}_2) \cdot (\boldsymbol{I} + \boldsymbol{T}^{-1}\boldsymbol{U}_2) \tag{4.31}$$

$$\boldsymbol{T} = (\boldsymbol{B} + \boldsymbol{L}_1) \cdot (\boldsymbol{I} + \boldsymbol{B}^{-1}\boldsymbol{U}_1) \tag{4.32}$$

where $\boldsymbol{B}$ is a block diagonal matrix defined as

$$\boldsymbol{B} = \boldsymbol{D} - (\boldsymbol{L}_1\boldsymbol{B}^{-1}\boldsymbol{U}_1) - colsum(\boldsymbol{L}_2\boldsymbol{T}^{-1}\boldsymbol{U}_2) - colsum(\boldsymbol{L}_3\boldsymbol{P}^{-1}\boldsymbol{U}_3) \tag{4.33}$$

It is interesting to note the similarity between the format of the three tridiagonal matrices in Equations (4.30) – (4.32) and the $\boldsymbol{LDU}$ format presented in Equation (4.19). Also similar is the manner with which the equations are solved, involving two steps, one with each part of the decomposition, as per Equations (4.20) – (4.21).

For the matrices defined in Equations (4.30) – (4.32) for Nested Factorization, this solution procedure is done in a hierarchical fashion (Appleyard and Cheshire, 1983). In the outermost level one solves

$$(\boldsymbol{P} + \boldsymbol{L}_3) \cdot (\boldsymbol{I} + \boldsymbol{P}^{-1}\boldsymbol{U}_3) \cdot \boldsymbol{q} = \boldsymbol{r} \tag{4.34}$$

using

$$\boldsymbol{q}' = \boldsymbol{P}^{-1}(\boldsymbol{r} - \boldsymbol{L}_3\boldsymbol{q}') \tag{4.35}$$

which can be solved explicitly one plane at a time, in a forward sweep through the reservoir, and then use

$$\boldsymbol{q} = \boldsymbol{q}' - \boldsymbol{P}^{-1}\boldsymbol{U}_3\boldsymbol{q} \tag{4.36}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

which can also be solved explicitly one plane at a time, but now in a backward sweep.

However, in both Equation (4.35) and Equation (4.36) one must be able to solve equations of the form

$$\boldsymbol{z} = \boldsymbol{P}^{-1}\boldsymbol{y} \tag{4.37}$$

where $\boldsymbol{P}^{-1}$ is not available directly. Therefore, it becomes necessary to first solve

$$(\boldsymbol{T} + \boldsymbol{L}_2) \cdot (\boldsymbol{I} + \boldsymbol{T}^{-1}\boldsymbol{U}_2) \cdot \boldsymbol{z} = \boldsymbol{y} \tag{4.38}$$

for each plane, by using

$$\boldsymbol{z}' = \boldsymbol{T}^{-1}(\boldsymbol{y} - \boldsymbol{L}_2\boldsymbol{z}') \tag{4.39}$$

which can be solved explicitly one line at a time, in a forward sweep through the plane, and then use

$$\boldsymbol{z} = \boldsymbol{z}' - \boldsymbol{T}^{-1}\boldsymbol{U}_2\boldsymbol{z} \tag{4.40}$$

which can also be solved explicitly one line at a time, but in a backward sweep.

Again, the steps involving Equation (4.39) and Equation (4.40) require solving equations of the form

$$\boldsymbol{w} = \boldsymbol{T}^{-1}\boldsymbol{v} \tag{4.41}$$

where $\boldsymbol{T}^{-1}$ is also not available directly. Therefore, it becomes necessary to first solve

$$(\boldsymbol{B} + \boldsymbol{L}_1) \cdot (\boldsymbol{I} + \boldsymbol{B}^{-1}\boldsymbol{U}_1) \cdot \boldsymbol{w} = \boldsymbol{v} \tag{4.42}$$

for each line, by using

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\boldsymbol{w'} = \boldsymbol{B}^{-1}(\boldsymbol{v} - \boldsymbol{L}_1\boldsymbol{w'}) \tag{4.43}$$

which can be solved explicitly one cell at a time, in a forward sweep through the line, and then use

$$\boldsymbol{w} = \boldsymbol{w'} - \boldsymbol{B}^{-1}\boldsymbol{U}_1\boldsymbol{w} \tag{4.44}$$

which can also be solved explicitly one cell at a time, but in a backward sweep.

Finally, the steps involving Equation (4.43) and Equation (4.44) require knowledge of $\boldsymbol{B}^{-1}$ to be solved. Since $\boldsymbol{B}$ was constructed to be a block diagonal matrix, determining its inverse is a straightforward matter, whose operation count depends on whether the problem is one, two or three-dimensional. It is worth noting from the procedure just described that the only storage that is actually required is that of $\boldsymbol{B}^{-1}$, and that $\boldsymbol{B}$ itself is not needed ($\boldsymbol{B}^{-1}\boldsymbol{U}_1$ could also be stored to avoid some additional calculations, if so desired and if memory is not an impediment).

Although the previous equations may seem to be implicit at first glance, as stated, they are actually explicit when calculated sequentially. That is because, due to the edges of the reservoir, the diagonal bands $\boldsymbol{L}_n$ and $\boldsymbol{U}_n$ do not span the coefficient matrix completely. This can be observed in Figure 4.1, which represents the block matrix structure for a simple $N_x = 2$, $N_y = 2$, $N_z = 2$ reservoir grid numbered using natural ordering, and containing 3 components, whose equations are clustered per cell (instead of per component, for example). In terms of the efficiency of the method, it is advantageous to number the smallest dimensions innermost, which normally implies numbering in the z-direction first (Appleyard and Cheschire, 1983).

PUC-Rio - Certificação Digital Nº 1612782/CA

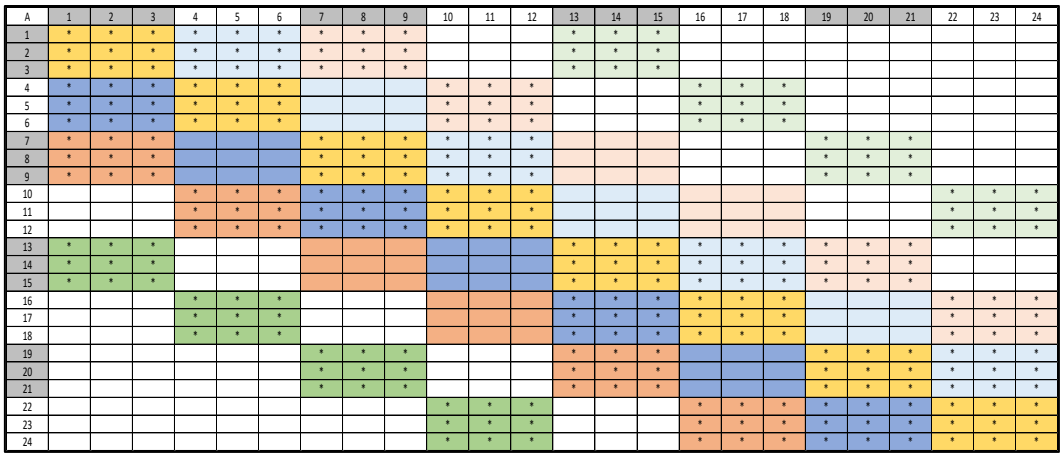

Figure 4.1 – Matrix structure for a (2×2×2) grid with natural ordering and with 3 phase components grouped together per cell. Darker colors represent the lower diagonal bands while lighter colors represent the upper ones.

As an example, when performing the calculations in the forward or backward directions for the plane equations, the first steps are reduced to

$$\boldsymbol{q}' = \boldsymbol{P}^{-1}\boldsymbol{r} \tag{4.45}$$

and

$$\boldsymbol{q} = \boldsymbol{q}' \tag{4.46}$$

respectively, because $\boldsymbol{L}_3$ and $\boldsymbol{U}_3$ will initially be zero. Furthermore, in the subsequent planes, the would-be implicit parts of the equations, $\boldsymbol{P}^{-1}\boldsymbol{L}_3\boldsymbol{q}'$ and $\boldsymbol{P}^{-1}\boldsymbol{U}_3\boldsymbol{q}$, respectively, depend only on the results obtained from the factorization of the plane that came just prior to it. The exact same occurs concerning the line and cell factorizations.

4.3.3 Constrained Pressure Residual Preconditioner

The final preconditioner implemented is called Constrained Pressure Residual. It consists of a two-stage approach based on the assumption that pressure is the principal variable governing fluid flow (Stüben et al., 2007). The foundation of the method resides in first approximately decoupling the pressure and saturation-

PUC-Rio - Certificação Digital Nº 1612782/CA

related variables through the use of a decoupler. Next, the pseudo-decoupled pressure system is solved, using techniques especially suited to it. Subsequently, its solution is used to update the residual vector, and then the full system can be solved for all variables simultaneously.

The idea behind multi-stage methods such as this is longstanding, having first been proposed in the field of reservoir simulation by Behie and Vinsome (1982), under the name Combinative Method. In their method, the subsystems were separated using partial, but exact, Gaussian Elimination onto the pressure equations, and the pressure variables were then computed through complete Gaussian Elimination. Next the residual was updated and the full system solved via incomplete factorization. Afterwards, Wallis (1983) coined the term CPR as he proposed an adaptation to the ORTHOMIN procedure that consisted of constraining the residual vector so as to establish zero residual sum on one or more subsets of coordinates, such as individual planes or, alternatively, on the pressure coefficients. This was also performed using a two-stage approach to the preconditioning steps, with incomplete factorization as the *outer* preconditioner (second stage). Furthermore, his work proposed a different preconditioning approach to the *inner* stage (first stage) that accounted for the distinct characteristics of individual submatrices in the coefficient matrix, performing local incomplete factorizations with varying levels of infill. Wallis et al. (1985) further expanded on this theme by applying a different tactic to the inner stage, through the use of a constraint matrix that sought to dampen the dominant eigenvectors of the problem, forcing the residual vectors to be orthogonal to their resulting eigenspace.

The contemporary approach to CPR is motivated by all of these concepts, and usually involves solving the pressure system with a numerical method specifically suited to the nearly elliptic[13] equations characteristic of that particular subsystem (Stüben et al., 2007; Cao et al., 2005). One of the most fitting method for this inner stage is Algebraic Multigrid (AMG), which is derived from the powerful concept of Geometric Multigrid (GM). Both AMG and GM use the strategy of *divide-and-conquer* to search for problems' solutions with the aid of less discretized grids.

PUC-Rio - Certificação Digital Nº 1612782/CA

[13] **Elliptic Equations** – Partial differential equations of the form

$$Au = a_{11}u_{xx} + a_{12}u_{xy} + a_{22}u_{yy} + a_1u_x + a_2u_y + a_0u = f$$

that possess the following property: $4a_{11}a_{22} > {a_{12}}^2$.

Multigrid methods are generally considered to be the fastest numerical methods currently available to solve elliptic PDEs (Trottenberg et al., 2001). On the other hand, the second stage of the preconditioner that solves for all of the variables simultaneously involves a hyperbolic[14] system of equations, which can be handled proficiently by ILU techniques (Stüben et al., 2007; Cao et al., 2005). In a way, the first stage may be seen as responsible for handling the long-range effects, while the second stage deals with the short-range ones (Hammersley and Ponting, 2008).

To visualize this entire process, following the approach of Stüben et al. (2007), we first consider the matrix equation

$$\boldsymbol{A}\boldsymbol{q} = \begin{bmatrix} \boldsymbol{A}_{pp} & \boldsymbol{A}_{ps} \\ \boldsymbol{A}_{sp} & \boldsymbol{A}_{ss} \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_p \\ \boldsymbol{q}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_p \\ \boldsymbol{r}_s \end{bmatrix} = \boldsymbol{r} \tag{4.47}$$

in which the equations are numbered with respect to the variable types (instead of with respect to the grid points, for example), and where $\boldsymbol{A}_{pp}$ represents the pressure block coefficients; $\boldsymbol{A}_{ss}$ represents the saturation block coefficients; $\boldsymbol{A}_{ps}$ and $\boldsymbol{A}_{sp}$ represent coupling coefficients between the pressure and saturation variables; $\boldsymbol{q}_p$ and $\boldsymbol{q}_s$ represent the pressure and saturation-related variables, respectively; and $\boldsymbol{r}_p$ and $\boldsymbol{r}_s$ represent the pressure and saturation-related residuals.

Applying a decoupler $\boldsymbol{D}$ to the coefficient matrix, it is transformed into

$$\boldsymbol{D}\boldsymbol{A} = \widetilde{\boldsymbol{A}} = \begin{bmatrix} \widetilde{\boldsymbol{A}}_{pp} & \widetilde{\boldsymbol{A}}_{ps} \\ \widetilde{\boldsymbol{A}}_{sp} & \widetilde{\boldsymbol{A}}_{ss} \end{bmatrix} \tag{4.48}$$

Now, the solution of the pressure equations $\boldsymbol{q}_p$ may be estimated via

$$\widetilde{\boldsymbol{A}}_{pp} \cdot \boldsymbol{q}_p = \boldsymbol{r}_p \tag{4.49}$$

using an AMG solver to be detailed further forward.

The residual vector $\boldsymbol{r}$ is then updated

[14] **Hyperbolic Equations** – Partial differential equations of the same form as elliptic ones, but that possess instead the following property: $4a_{11}a_{22} < {a_{12}}^2$.

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\hat{\boldsymbol{r}} = \boldsymbol{r} - \widetilde{\boldsymbol{A}} \begin{bmatrix} \boldsymbol{q}_p \\ \boldsymbol{0} \end{bmatrix} \tag{4.50}$$

so as to partially remove the effects of the pressure variables.

Next, the full system is solved approximately through an ILU factorization (represented here by $\boldsymbol{M}$)

$$\hat{\boldsymbol{q}} = \boldsymbol{M}^{-1}\hat{\boldsymbol{r}} \tag{4.51}$$

in a manner similar to the single-stage global preconditioner that was previously described.

In this research, the technique chosen for the second stage was standard ILU(0) factorization, also described previously, which was recommended by Cao et al. (2005). This option sought to limit the already increased setup time required by CPR.

Finally, the complete solution is obtained by combining the two partial solutions of each stage

$$\boldsymbol{q} = \hat{\boldsymbol{q}} + \begin{bmatrix} \boldsymbol{q}_p \\ \boldsymbol{0} \end{bmatrix} \tag{4.52}$$

Moreover, this entire process can be seen as equivalent to applying a single-stage preconditioner of the form

$$\boldsymbol{M}_{2S}{}^{-1} = \boldsymbol{M}^{-1}\left[\boldsymbol{I} - \left(\widetilde{\boldsymbol{A}} - \boldsymbol{M}\right)\begin{bmatrix} \widetilde{\boldsymbol{A}}_{pp}{}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right] \tag{4.53}$$

to the preconditioning step of the iterative method being used:

$$\boldsymbol{M}_{2S}{}^{-1}\boldsymbol{q} = \boldsymbol{r} \tag{4.54}$$

This means that the same preconditioner is being employed in each preconditioning step of the iterative method. This observation is important because

PUC-Rio - Certificação Digital Nº 1612782/CA

the preconditioned versions of the methods presented in Chapter 3 assume this particular condition. Otherwise, flexible variants such as FGMRES would have to be used (Saad, 2003; and Hammersley and Ponting, 2008).

Returning to the first part of the algorithm, the decoupling of the problem variables, its objective is to comply to four main criteria: (i) weaken the coupling between the variables – that is, minimize $\|\widetilde{\boldsymbol{A}}_{ps}\|$ as much as possible; (ii) reduce the condition number of the full system $\left(\kappa(\widetilde{\boldsymbol{A}}) < \kappa(\boldsymbol{A})\right)$ and of the pressure system $\left(\kappa(\widetilde{\boldsymbol{A}}_{pp}) < \kappa(\boldsymbol{A}_{pp})\right)$; (iii) be relatively inexpensive to compute; and (iv) maintain the nearly elliptic nature of the pressure variables (Stüben et al., 2007; and Gries et al., 2013).

Although the coefficient matrix is usually highly asymmetrical and indefinite, frequently the pressure subsystem $\boldsymbol{A}_{pp}$ is nearly symmetrical and presents diagonal dominance, with a positive diagonal and negative off-diagonal values (Z-Matrix properties), and may even possess eigenvalues with positive real parts (resembling M-Matrix properties). These are all characteristics which are amenable to AMG methods and that may eventually be lost through the decoupling process ($\widetilde{\boldsymbol{A}}_{pp}$ could become strongly asymmetrical, and even indefinite), degrading the performance of the *solver* considerably (Gries et al., 2013; and Stüben et al., 2007).

Several techniques have been proposed to act as decouplers. The most common ones include Alternate-Block Factorization (ABF) and quasi-IMPES (QI), which were used, for example, in the works of Cao et al. (2005), Stüben et al. (2007) and Hammersley and Ponting (2008). Lacroix et al. (2001), Scheichl et al. (2003) and Al-Shaalan et al. (2009) offered some additional decoupling options, however, some of their procedures tend to lead to significant transformations to the pressure system, often eliminating the elliptic nature in unpredictable ways.

In this research, an alternative form of decoupling named Dynamic Row Sum (DRS) was selected, according to the work of Gries et al. (2013), who developed it based on the work of Scheichl et al. (2003). To be more precise, in this method $\boldsymbol{D}$ is intended to act more as a global preconditioner to the matrix equation than as an actual decoupler. Considering a coefficient matrix ordered by grid points, or point-wise ordered, the decoupler may be defined as the following diagonal matrix

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\boldsymbol{D} = \begin{bmatrix} [\boldsymbol{D}]_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & [\boldsymbol{D}]_i \end{bmatrix} \tag{4.55}$$

whose diagonal blocks assume the following structure for a three-phase problem:

$$[\boldsymbol{D}]_i = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.56}$$

in which the entries of the first row are specified as

$$\delta_i = \begin{cases} 0 & \text{if} \quad \dfrac{\left|\left[a_{x,1}\right]_{i,i}\right|}{\sum_{j=1, j\neq i}^{n_{points}} \left|\left[a_{x,1}\right]_{i,j}\right|} < \varepsilon_{dd} \\ 1 & \text{otherwise} \end{cases} \tag{4.57}$$

where n_{points} represents the number of neighboring connections for a particular grid block.

This decoupler configuration will result in a pressure submatrix $\widetilde{\boldsymbol{A}}_{pp}$ that reflects the sum of all the relevant pressure-aligned parts of the different phases of the coefficient matrix. Hence, only the parts of $\boldsymbol{A}$ relevant to the pressure system and that are expected not to degrade the performance of AMG are included in the preconditioned matrix $\widetilde{\boldsymbol{A}}$, which is guaranteed to have only positive diagonal entries and which should possess diagonal dominance, unless the value of ε_{dd} is chosen too small. Here, the value of this parameter was set to $\varepsilon_{dd} = 0.2$, as per Gries et al. (2013).

Furthermore, a second check is performed to avoid the inclusion of saturation couplings that have only negligible effect on pressure. The value of δ_i is thus set to zero whenever the following condition is met:

$$\sum_{j=1}^{n_{points}} \left|\left[a_{1,x}\right]_{i,j}\right| < \varepsilon_{ps} \left|\left[a_{1,1}\right]_{i,i}\right| \tag{4.58}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

The magnitude of ε_{ps} will determine which saturation rows with weak coupling shall be excluded. As suggested by Gries et al. (2013), this was taken to be $\varepsilon_{ps} = 0.02$ for the simulations performed.

Moreover, there is a further adjustment that could be performed on the δ_i factors so as to scale all of them into the same volume base, in a procedure named weighted DRS, or wDRS (Brown et al., 2015). The method as first proposed by Scheichl et al. (2003) operated on terms that were all in reservoir volume dimension (reservoir pressure and temperature conditions), whereas both the simulators encountered commercially and the one used for this research handle all quantities in surface volume conditions (standard pressure and temperature). Thus, a proper scaling of the decoupler may be accomplished by transforming the diagonal blocks of $\boldsymbol{D}$ into

$$[\boldsymbol{D}]_i = \begin{bmatrix} \delta_o\left(B_o - R_s B_g\right) & \delta_w(B_w) & \delta_g\left(B_g\right) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.59}$$

It should be noted that the procedure presented does not truly decouple the problem variables (the submatrix $\boldsymbol{A}_{sp}$, for example, is left unaltered by wDRS), which could negatively impact the performance of the second stage of CPR. However, by facilitating the convergence rate of AMG, it seeks a compromise between the efficiency of the first and second stages of the method.

After preconditioning with wDRS, there are additional options of decouplers that could be applied to the matrix equation in an attempt to achieve a stronger degree of decoupling. These would act as right preconditioners. Gries et al. (2013) tested two such alternatives, namely quasi-ABF and Saturation Column Elimination (SCE), but neither method improved at all the performance obtained from pure wDRS. Consequently, no additional decouplers were implemented for the CPR algorithm.

The last feature of the algorithm that still requires detailing is the actual AMG *solver* used for the first stage. Some of the key features of AMG that render it so attractive as a *solver* are its robustness, its scalability (particularly when used in conjunction with a Krylov method) and the fact that no geometric grid information is required for its application. These factors provide the flexibility desired for an

PUC-Rio - Certificação Digital Nº 1612782/CA

efficient plug-in *solver* and allow for detailed reservoir models, with the added option of employing non-structured grids (Stüben et al., 2001; and De Sterck et al., 2007). In the CPR algorithm AMG constitutes part of the solution procedure, but the method is so efficient and versatile that in many scenarios it could also be applied as a stand-alone *solver* (Gries, 2017).

The overall idea behind multigrid methods involves two basic principles: (i) error smoothing; and (ii) coarse grid correction. Error smoothing implies that, after the application of some iterative smoothing procedure, the error of the approximate solution becomes smooth. This means that the high frequency components of the error, if we were to apply a Fourier expansion to it, become small, while the low frequency components remain nearly constant. In fact, the error may actually still be large, but the purpose of this principle is that it can be made smooth over the domain. Subsequently, coarse grid correction states that a quantity that is smooth on a fine grid Ω_h (say, a well discretized grid) may be approximated accurately on a coarser grid Ω_H, where fewer variables are represented (Trottenberg et al., 2001). Hence, in a way, the smoothing process is responsible for reducing the high frequency error components, while the coarser grids are responsible for reducing the low frequency ones (Stüben, 2007). This is due to the fact that those low frequency error components on the fine grid become high frequency ones on a coarser grid; thus, each grid is ultimately responsible for reducing the high frequency components corresponding to its own scale.

To perform the conversion of the variables between fine and coarse grids, a coarsening strategy must first be defined, to select which fine variables will be carried to the coarse grid and which shall remain as fine. Additionally, two operators must be introduced: (i) a restriction operator $\boldsymbol{I}_h^H$, to perform the transfer from fine to coarse grid; and (ii) an interpolation operator $\boldsymbol{I}_H^h$, to perform the transfer back from coarse to fine grid (Trottenberg et al., 2001). This is illustrated in Equations (4.60) – (4.61):

$$\boldsymbol{I}_h^H = \mathcal{F}(\Omega_h) \rightarrow \mathcal{F}(\Omega_H) \tag{4.60}$$

$$\boldsymbol{I}_H^h = \mathcal{F}(\Omega_H) \rightarrow \mathcal{F}(\Omega_h) \tag{4.61}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

Thus, given a problem originally on a fine grid

$$\boldsymbol{A}_h \boldsymbol{u}_h = \boldsymbol{f}_h \qquad (\Omega_h) \tag{4.62}$$

AMG begins by finding an approximate solution $\bar{u}_h$ via an initial guess $\boldsymbol{u}_h{}^{(0)}$ and a smoothing procedure, called pre-smoothing. This may normally be accomplished with one or more steps of a Gauss-Seidel iteration. Here a single step was chosen, as per Stüben et al. (2001). The classical Gauss-Seidel procedure is shown in Equation (4.63) (Ertekin et al., 2001):

$$u_{h_i}{}^{(k+1)} = \frac{1}{a_{h_{ii}}}\left[f_{h_i} - \sum_{j=1}^{i-1} a_{h_{ij}}\, u_{h_j}{}^{(k+1)} - \sum_{j=i+1}^{n} a_{h_{ij}}\, u_{h_j}{}^{(k)} \right] \tag{4.63}$$

where k is the Gauss-Seidel step, or iteration level; and n is the dimension of square matrix $\boldsymbol{A}_h$.

Next, the residual (also known as the defect) is computed

$$\boldsymbol{d}_h = \boldsymbol{f}_h - \boldsymbol{A}_h \bar{\boldsymbol{u}}_h \tag{4.64}$$

and then restricted

$$\boldsymbol{d}_H = \boldsymbol{I}_h^H \boldsymbol{d}_h \tag{4.65}$$

to transfer it to the coarse domain Ω_H.

AMG now exploits the fact that solving the defect equation

$$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{d} \tag{4.66}$$

is equivalent to solving the original equation, with e representing the error vector, such that

$$\boldsymbol{u} = \bar{\boldsymbol{u}} + \boldsymbol{e} \tag{4.67}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

as demonstrated next

$$\boldsymbol{Au} = \boldsymbol{A\bar{u}} + \boldsymbol{Ae} \;\rightarrow \boldsymbol{f} = (\boldsymbol{f} - \boldsymbol{d}) + \boldsymbol{d} = \boldsymbol{f} \tag{4.68}$$

Unfortunately, the solution of Equation (4.66), in the form just presented, is as difficult as the original problem. However, if in this case $\boldsymbol{A}$ is approximated by a matrix $\widetilde{\boldsymbol{A}}$ whose solution is simpler to obtain, then the following iterative process may be established

$$\begin{aligned} \bar{\boldsymbol{u}}_h{}^{(\nu)} \;\rightarrow\; \boldsymbol{d}_h{}^{(\nu)} &= \boldsymbol{f}_h - \boldsymbol{A}_h \bar{\boldsymbol{u}}_h{}^{(\nu)} \;\rightarrow \widetilde{\boldsymbol{A}}_h \boldsymbol{e}_h{}^{(\nu)} = \boldsymbol{d}_h{}^{(\nu)} \;\rightarrow\; \bar{\boldsymbol{u}}_h{}^{(\nu+1)} \\ &= \bar{\boldsymbol{u}}_h{}^{(\nu)} + \boldsymbol{e}_h{}^{(\nu)} \end{aligned} \tag{4.69}$$

In the context of AMG, $\widetilde{\boldsymbol{A}}$ is simply the coefficient matrix that operates on the coarse grid Ω_H. It can be constructed as follows

$$\boldsymbol{A}_H = \boldsymbol{I}_h^H \boldsymbol{A}_h \boldsymbol{I}_H^h \tag{4.70}$$

and also receives the name of *Galerkin operator*.

Returning to the algorithm and omitting the iteration levels (ν) for simplicity, the defect equation is now solved on the coarse grid

$$\boldsymbol{A}_H \boldsymbol{e}_H = \boldsymbol{d}_H \tag{4.71}$$

and its result is interpolated back to the fine grid

$$\boldsymbol{e}_h = \boldsymbol{I}_H^h \boldsymbol{e}_H \tag{4.72}$$

before a new approximate solution can be found

$$\boldsymbol{u}_h = \bar{\boldsymbol{u}}_h + \boldsymbol{e}_h \tag{4.73}$$

Likewise, this new solution $\boldsymbol{u}_h$ should also be smoothed out, in a process called post-smoothing. This is done, for example, by using one or more steps of

PUC-Rio - Certificação Digital Nº 1612782/CA

Gauss-Seidel iteration. Once again, here a single step was chosen, following Stüben et al. (2001). The order of relaxation in both cases is C-F, which means that it is first applied to the coarse variables and then to the fine ones.

Finally, since this is an iterative process, this solution may serve as the initial guess $\boldsymbol{u}_h{}^{(1)}$ for a subsequent cycle of the procedure described by Equation (4.69). This can be further repeated until a satisfactory solution $\boldsymbol{u}_h{}^{(\nu+1)}$ is reached, given a set of stopping criteria. The criteria chosen for this research were of either attaining a residual reduction of $\varepsilon = 10^{-6}$, as suggested by Gries et al. (2013), or reaching a maximum number of $\nu_{max} = 2$ AMG cycles. This last parameter was defined based on the results of test cases performed with the simulator. Although, these are relatively relaxed criteria, they seemed to lead to best overall convergence.

The process described thus far has been a simplified version of a full AMG cycle, having made use of only two grids, one fine and one coarse. In practice, additional coarse grids are required for the method to be functional. The reason for this is that there is a limit to the degree of how coarse a grid can be made from a given fine grid; otherwise, the interpolation of the coarse grid results will not lead to an accurate representation of the fine variables. Conversely, if the defect equation remains too challenging to be solved on the coarse grid, with too many variables present, then not enough coarsening has been accomplished and the solution process cannot advance. A compromise between these two issues is reached by performing multiple coarsening steps recursively, that is – the coarse grid of one level becomes the fine grid of the next, such that no restriction or interpolation is too abrupt, and that the dimensions of the coarsest level are sufficiently reduced so that it may be solved with minor effort (normally via a direct method). This is the reasoning behind the name multigrid (Trottenberg et al., 2001).

There are several ways to implement a multigrid cycle. Two possibilities are depicted in Figure 4.2, representing a four-grid scenario with two options of cycle index value: $\gamma = 1$ (on the left) and $\gamma = 2$ (on the right). The filled dots in the figure represent grids where only smoothing is performed, while the empty dots represent the coarsest grids of the cycles, where an exact solution is calculated. The lines symbolize either restriction or interpolation operations. The V-Cycle was the only one implemented in this work.

PUC-Rio - Certificação Digital Nº 1612782/CA

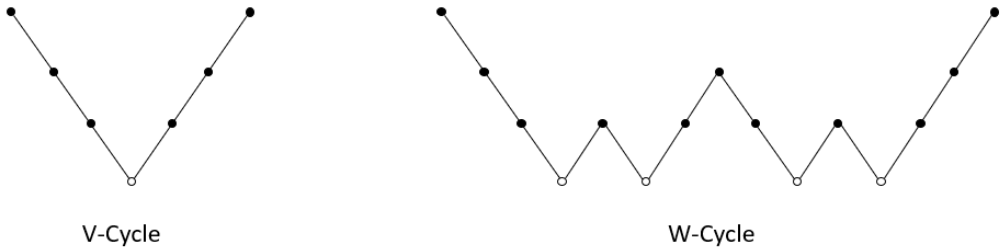

Figure 4.2 – Examples of AMG Cycles (Adapted from Trottenberg et al., 2001).

The coarsening strategy that will be used to generate these coarse grids needs to assure that the smooth errors can be determined via interpolation, while maintaining the size of the coarse grid operator at a reasonable level. These constraints are intimately related to the number of coarse variables (C-variables) and to the strength of their connectivity to the fine variables (F-variables). They implicate limiting the number of C-variables, so as to restrain the amount of work per cycle, while still keeping the remaining F-variables sufficiently connected to them, which involves surrounding the F-variables with C-variables from which to interpolate. At the end of the process every variable will belong to exactly one of two subsets

$$\Omega_h = \mathrm{C}_h \cup \mathrm{F}_h \tag{4.74}$$

where C_h is the set of coarse variables; and where F_h is the set of fine variables (Stüben et al., 2001 and Trottenberg et al., 2001).

The strategy chosen in this research is named Standard Coarsening, as presented by Stüben et al. (2001). It requires that the F-variables have a minimum number of its neighbors be represented as C-variables, and particularly those most closely related to it, guaranteeing strong C-F connectivity. For a given variable i whose coupled variables are represented by $j \in N_i$ (where N_i represents the set containing all variables connected to i), a connection is defined as strong whenever

$$-a_{ij} \geq \varepsilon_{\mathrm{str}} \cdot \max_{a_{ik}<0} |a_{ik}| \tag{4.75}$$

and its complete set of strong couplings is denoted by

PUC-Rio - Certificação Digital Nº 1612782/CA

$$S_i = \{\, j \in N_i \mid i \text{ strongly coupled to } j \,\} \tag{4.76}$$

It is assumed here that no strong positive connections exist, even if positive off-diagonals occasionally arise in the coefficient matrix. A value of $\varepsilon_{\text{str}} = 0.75$ was adopted for this parameter, after adjustments through some test cases.

Since the coupling relations might not be symmetric, a second connectivity set is also defined which contains all variables j which are strongly coupled to i

$$S_i^T = \{\, j \in \Omega \mid i \in S_j \,\} \tag{4.77}$$

The C/F splitting then consists of selecting a first variable i to become a C-variable and consequently defining all j variables which are strongly coupled to it as F-variables $\left(j \in S_i^T\right)$. The method proceeds by selecting another variable among the undecided ones to become the next C-variable, then all the variables strongly coupled to it are made to be F-variables, with these steps continuing until every variable has been accounted for as either coarse or fine. This procedure assures that coarsening is performed towards directions in which the smooth error changes more slowly.

To further ensure that a uniform distribution of C and F-variables occurs, with the F-variables surrounded by C-variables, a strategy must also be implemented for selecting the next C-variable from the undecided ones (U-variables). This is done by introducing a new parameter λ_i which measures the importance of any U-variable remaining and is defined as

$$\lambda_i = \left|S_i^T \in \mathrm{U}\right| + 2\left|S_i^T \in F\right| \tag{4.78}$$

where $|S|$ denotes the number of variables present in a given set S.

This strategy initially forces variables with many strong connections to themselves to be selected as C-variables; however, as it progresses, it begins tending to select as C-variables those U-variables with most connections to F-variables. This ensures that each F-variable is strongly coupled to at least one C-

PUC-Rio - Certificação Digital Nº 1612782/CA

variable, as well as ensuring that none of the C-variables is strongly coupled to any previously chosen C-variable (Stüben et al., 2001).

Once every variable has been selected on a given level, the multigrid approach continues by taking the variables defined as coarse (those that will compose the coarse grid) and performing a new selection onto them, to create an even coarser grid. The systematic construction of ever coarser grids persists until some pre-established criterium is met, such as a maximum number of permissible grids having been created or a minimum number of variables having been selected for a new grid. The CPR version implemented in this work employed both of these criteria, adopting a limit of at most 10 coarse grids and at most 1000 variables in the coarsest grid, and halting the grid construction process once either of them was reached. The values of these parameters were also subject to some fine tuning through experimentation.

Focusing now on the restriction and interpolation operators, it was decided to construct them transposed to one another on each grid level. As stated by Stüben et al. (2001), it has been shown in practice that even if the coefficient matrix is asymmetrical, this choice does not seem to cause increased difficulty for AMG and simplifies the setup phase of the algorithm.

The strategy chosen to interpolate (and, consequently, to restrict) is named Standard Interpolation. It attempts to enforce that each F-variable have a fixed percentage of its total connectivity accounted for in the C-variables. This is represented by the parameter $\tau \geq 1$ in Equation (4.79), such that the following inequality is satisfied

$$\sum_{k \in P_i} |a_{ik}| \geq \frac{1}{\tau} \sum_{j \in N_i} |a_{ij}| \tag{4.79}$$

where P_i represents the set of interpolation variables pertaining to $i \in F$.

In Standard Interpolation this set is defined so as to include the strong coarse connections, as well as to partially account for the strong fine connections. The strong coarse and strong fine connection sets are represented as follows, respectively

PUC-Rio - Certificação Digital Nº 1612782/CA

$$C_i^s = C \cap S_i \tag{4.80}$$

$$F_i^s = F \cap S_i \tag{4.81}$$

The interpolation set can then be defined as

$$\hat{P}_i = C_i^s \cup \bigcup_{j \in F_i^s} C_j^s \tag{4.82}$$

which includes all of the strong coarse variables directly coupled to $i \in F$, as well as those coupled to its strong fine neighbors. In this sense, the neighborhood of $i \in F$ may be extended to include the additional connections arising from its fine neighbors (De Sterck et al., 2007)

$$\hat{N}_i = N_i \cup \bigcup_{j \in F_i^s} N_j \tag{4.83}$$

This kind of interpolation strategy, that considers connections beyond the direct neighborhood of the fine variable subject to interpolation, is called a Long-Range strategy. In this case distance-two points were also included into the interpolation set. As demonstrated by De Sterck et al. (2007), such interpolation methods may significantly improve the convergence capabilities of AMG and reduce overall processing time.

Moreover, as proposed by Stüben et al. (2001), the interpolation set may also be divided into two distinct subsets, one containing the negative connections and another with the positive ones

$$\hat{P}_i = \hat{P}_i^{\,-} \cup \hat{P}_i^{\,+} \tag{4.84}$$

Assuming that smooth error has small residuals after relaxation

$$\boldsymbol{Ae} \approx 0 \tag{4.85}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

then, for each $i \in F$, the error equation can be written in the form

$$a_{ii}e_i + \sum_{j \in N_i} a_{ij}e_j = 0 \tag{4.86}$$

Substituting every $e_j \in F_i^S$ by

$$-\frac{1}{a_{jj}} \sum_{k \in N_i} a_{ik}e_k \tag{4.87}$$

leads to the following interpolation formula for the error vector:

$$e_i = \sum_{k \in \hat{P}_i} w_{ik}e_k \qquad \text{with} \qquad w_{ik} = \begin{cases} -\alpha_i \, \hat{a}_{ik}/\hat{a}_{ii} & \left(k \in \hat{P}_i^{\,-}\right) \\ -\beta_i \, \hat{a}_{ik}/\hat{a}_{ii} & \left(k \in \hat{P}_i^{\,+}\right) \end{cases} \tag{4.88}$$

where w_{ik} represents the *interpolation weights*, used to compute the F-variables from the C-variables; and where the weight coefficients, which function as scaling factors, are defined as

$$\alpha_i = \frac{\sum_{j \in \hat{N}_i} \hat{a}_{ij}^{\,-}}{\sum_{k \in \hat{P}_i} \hat{a}_{ik}^{\,-}} \tag{4.89}$$

$$\beta_i = \frac{\sum_{j \in \hat{N}_i} \hat{a}_{ij}^{\,+}}{\sum_{k \in \hat{P}_i} \hat{a}_{ik}^{\,+}} \tag{4.90}$$

in which

$$\hat{a}_{ij}^{\,-} = \begin{cases} \hat{a}_{ij} & \text{if } \hat{a}_{ij} < 0 \\ 0 & \text{if } \hat{a}_{ij} \geq 0 \end{cases} \qquad \text{and} \qquad \hat{a}_{ij}^{\,+} = \begin{cases} 0 & \text{if } \hat{a}_{ij} \leq 0 \\ \hat{a}_{ij} & \text{if } \hat{a}_{ij} > 0 \end{cases} \tag{4.91}$$

These parameters can be further detailed as follows

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\hat{a}_{ii} = a_{ii} + \sum_{j \in F_i^s} a_{ij} \frac{(-a_{ji})}{a_{jj}} \tag{4.92}$$

$$\hat{a}_{ik} = \sum_{j \in F_i^s} a_{ij} \frac{(-a_{jk})}{a_{jj}} \qquad \text{where} \qquad k \in C_j^s \bigcup N_j^w \tag{4.93}$$

$$\hat{a}_{ij} = a_{ij} \qquad \text{where} \qquad j \in C_i^s \bigcup N_i^w \tag{4.94}$$

and where a connected variable may contribute as $\hat{a}_{ik}$, $\hat{a}_{ij}$ or even both, depending on which subsets it belongs to. Notice that for these formulas *j* represents variables with a direct connection to i, while k represents those with an indirect connection.

In the scenario where there is not any strong positive connection $\left(\hat{P}_i^{\,+} = \emptyset\right)$ the equations can be modified such that $\beta_i = 0$ and all positive connections are added directly to the diagonal.

Furthermore, to avoid a substantial increase in work and storage requirement that would occur due to the Standard Interpolation strategy's broader neighboring sets, the interpolation operators are normally truncated (De Sterck et al., 2007). This is done by suppressing, in each row, all interpolation weights which are smaller than the largest one by a certain factor. A value of $\varepsilon_{tr} = 0.7$ was adopted for this parameter. Although this may seem an aggressive limit, it proved essential in avoiding that the interpolation matrix retain an excessive number of nonzero terms, as the size of the problems being solved increased. Lastly, after truncation is completed, the remaining weights are rescaled so as to preserve the total row sum.

With respect to the remaining variables $i \in C$, the transfer between grids is performed directly and interpolation is simply given by

$$(e_i)_h = (e_i)_H \tag{4.95}$$

It is worth noting that all of the coarsening process, involving the sequential selection of coarse sets of variables, and then the construction of all the restriction, interpolation and coarse-grid operators, may be performed in their entireties during the setup phase of CPR (Brown et al., 2015).

PUC-Rio - Certificação Digital Nº 1612782/CA

The solution stage of AMG (located within the solution stage of CPR), called upon to solve the pressure system, may be executed recursively, as per the pseudocode presented in Algorithm 4.3 (Trottenberg et al. 2001). In this code, the finest grid level is represented by $k = \xi$, while the coarsest level is $k = 0$, and only a single V-Cycle is performed at each iteration. This version of multigrid is denoted Correction Scheme (CS), due to the fact that, on the coarse grids, the equations solved are related to corrections to the fine grid approximation. It is those corrections which are transferred between grids. The Full Multigrid (FMG) approach, which was not implemented, starts the process with an approximate solution on the coarsest level (as opposed to the finest level with CS) and also involves the direct transfer of these approximate solutions between grids (in addition to the correction transfers).

Furthermore, in the implemented version of the algorithm the solution on the coarsest grid is obtained via Gaussian Elimination, utilizing the software UMFPACK. This direct *solver* is comprised of a series of routines for solving sparse linear systems through the Unsymmetric Multifrontal method, as described by Davis and Duff (1997).

Finally, a general overview of the CPR algorithm is depicted in Figure 4.3.

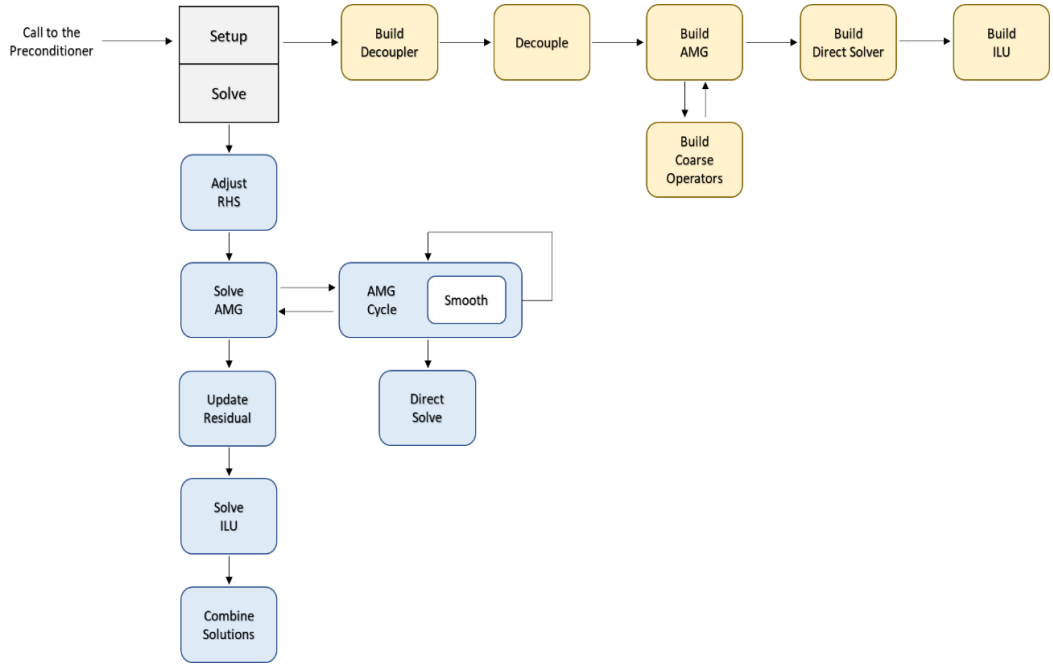

Figure 4.3 – Flow diagram representing the CPR preconditioning algorithm.

PUC-Rio - Certificação Digital Nº 1612782/CA

Algorithm 4.3 – AMG Solver.

$\boldsymbol{u}_{\xi}^{(0)} = 0$

$\boldsymbol{r}_{\xi}^{(0)} = \boldsymbol{f}_{\xi} - \boldsymbol{A}_{\xi}\,\boldsymbol{u}_{\xi}^{(0)}$

$\nu = 0$

while $\left\|\boldsymbol{r}_{\xi}^{(\nu)}\right\| > \eta \cdot \left\|\boldsymbol{f}_{\xi}\right\|$ and $\nu < \nu\_lim$

$k = \xi$

$\boldsymbol{u}_{k}^{(\nu+1)} = AMG\_Cycle\left(k, \boldsymbol{u}_{k}^{(\nu)}, \boldsymbol{A}_{k}, \boldsymbol{f}_{k}\right)$

{

$\bar{\boldsymbol{u}}_{k}^{(\nu)} = Smooth\left(\boldsymbol{u}_{k}^{(\nu)}, \boldsymbol{A}_{k}, \boldsymbol{f}_{k}\right)$

$\bar{\boldsymbol{d}}_{k}^{(\nu)} = \boldsymbol{f}_{k} - \boldsymbol{A}_{k}\,\bar{\boldsymbol{u}}_{k}^{(\nu)}$

$\bar{\boldsymbol{d}}_{k-1}^{(\nu)} = \boldsymbol{I}_{k}^{k-1}\,\bar{\boldsymbol{d}}_{k}^{(\nu)}$

if $k = 1$

$Gaussian\_Elimination\_Solve\left(\boldsymbol{A}_{k-1}\,\boldsymbol{e}_{k-1}^{(\nu)} = \bar{\boldsymbol{d}}_{k-1}^{(\nu)}\right)$

else $k > 1$

$\boldsymbol{e}_{k-1}^{(\nu)} = AMG\_Cycle\left(k-1, 0, \boldsymbol{A}_{k-1}, \bar{\boldsymbol{d}}_{k-1}^{(\nu)}\right)$

end

$\boldsymbol{e}_{k}^{(\nu)} = \boldsymbol{I}_{k-1}^{k}\,\boldsymbol{e}_{k-1}^{(\nu)}$

$\boldsymbol{u}_{k}^{(\nu)} = \bar{\boldsymbol{u}}_{k}^{(\nu)} + \boldsymbol{e}_{k}^{(\nu)}$

$\boldsymbol{u}_{k}^{(\nu+1)} = Smooth\left(\boldsymbol{u}_{k}^{(\nu)}, \boldsymbol{A}_{k}, \boldsymbol{f}_{k}\right)$

}

$\boldsymbol{r}_{\xi}^{(\nu+1)} = \boldsymbol{f}_{\xi} - \boldsymbol{A}_{\xi}\,\boldsymbol{u}_{\xi}^{(\nu+1)}$

$\nu = \nu + 1$

end

PUC-Rio - Certificação Digital Nº 1612782/CA

5
Performance of Reservoir Simulator Solvers

To assess the efficiency of the various *solvers* implemented, several reservoir simulations were carried out to compare different aspects of the *solvers'* performance, such as total simulation duration, time required to construct the preconditioners, number of linear iterations executed, among others. These simulations were performed using a simulator named Geresim Simulator (GSim), which is being currently developed at Pontifical Catholic University of Rio de Janeiro, as a joint project with Petrobras. The objective of these tests was to evaluate the robustness and efficiency of the *solvers*, in terms of their capacity to handle problems of considerable size and the computational effort they required to achieve convergence. The aim was thus to find the *solver* most suited to integrate a simulator developed for multiphase reservoir flow problems with fully implicit formulation.

This chapter will first introduce the reservoir model and the computer that were used, as well as describe some aspects of the simulations performed. Subsequently, the results obtained in these simulations shall be detailed and analyzed, with the main conclusions presented. To facilitate comprehension, a diagram is presented in Figure 5.1 depicting the main parts of a reservoir simulator that are relevant to the ensuing discussions.

PUC-Rio - Certificação Digital Nº 1612782/CA

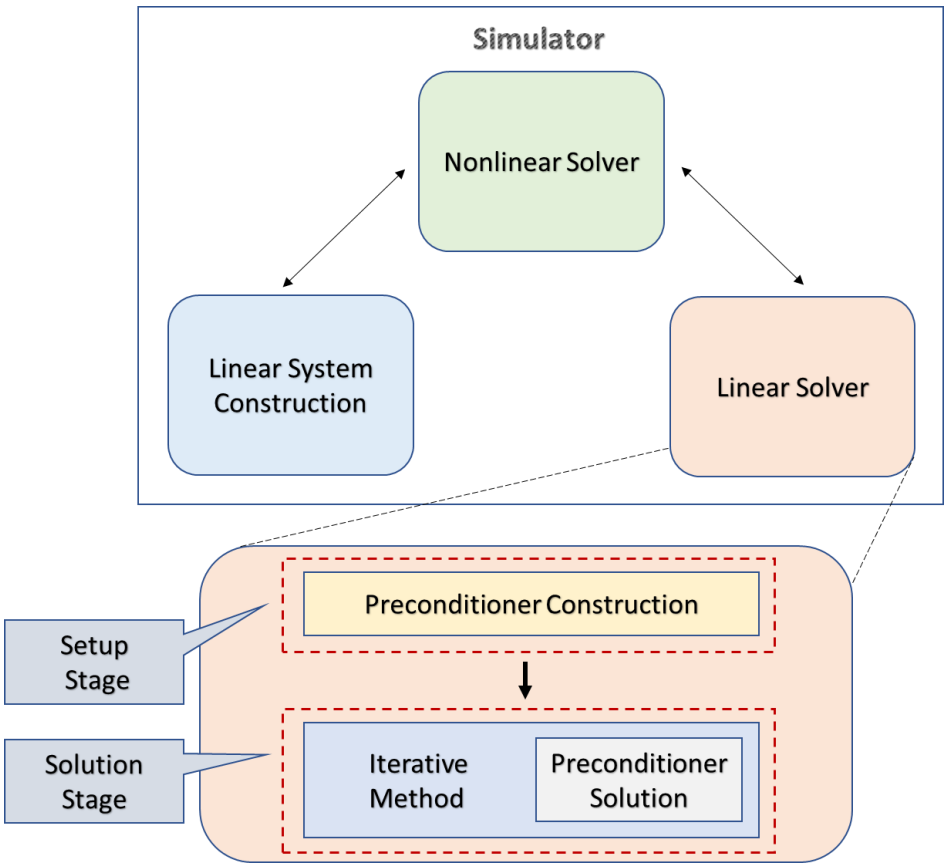

Figure 5.1 – Diagram depicting the most relevant parts of a reservoir simulator.

It can be seen here that the linear *solver*, the object of interest of this thesis, is an entity which is called upon by a nonlinear *solver* to find the solution to a linear system of equations that it deemed necessary to construct (during a linearization process). Whenever the linear *solver* is requested to return a solution to a set of equations, it will proceed by first constructing a preconditioner, in what constitutes the bulk of a Setup Stage, and then attempt to solve the system via an iterative method, in what represents a Solution Stage. During this solution stage, the iterative method will further call (several times) for the solution of some new linear systems of equations, through a process denoted here as preconditioner solution (such as described in Chapter 4). Once the iterative method identifies that it has obtained a suitable enough solution, the linear *solver* shall return this solution to the nonlinear one, which in turn will verify if convergence for a given time-step has been satisfactorily reached, or if additional linearization steps are necessary.

Finally, although the actual memory demanded by the various solution processes used in each problem will not be specifically analyzed here (the focus being directed to other aspects of their performances), this factor may eventually

PUC-Rio - Certificação Digital Nº 1612782/CA

restrict which techniques can be employed in certain cases, if the memory available is not sufficient. In terms of the total memory necessary for the *solver*, this will include the memory consumption of the iterative method, seen previously in Table 3.1, in addition to that of the preconditioner, whose values are estimated in Table 5.1. The matrix column in this table represents the number of additional matrices that must be built, and that will occupy an amount of memory equivalent to that of the original coefficient matrix, while the vector column represents the number of vectors to be built, not counting the solution and right-hand side ones. It must be stressed, however, that the figures presented here should be considered to be general approximations of the actual memory consumptions seen in practice. This is especially true in the case of CPR, whose memory requirement depends profoundly on the size of the coarse grids built during the AMG process, which is a particularly difficult trait to predict beforehand.

Table 5.1 – Memory requirement for the various implemented preconditioning methods.

Preconditioner	Memory Requirement	
	Matrices	Vectors
ILU	1	1
NF	0	~ 11
CPR	~ 3	~ 40

PUC-Rio - Certificação Digital Nº 1612782/CA

5.1 Problem Description

Reservoir Model

The reservoir model used for the simulations employed a cartesian grid with rectangular dimensions of size 2,500 meters x 2,500 meters x 100 meters in the x, y and z directions, respectively. Its spatial orientation is horizontal. The reservoir rock is homogeneous and isotropic, with an absolute permeability of 1000 mD, and it possesses a uniform initial porosity of 30%.

The reservoir is originally saturated with oil and water, with the contact between the free fluids occurring 60 meters from the top of the formation. The connate water saturation inside the free oil zone is equal to 8.8%. The original reservoir pressure at the top depth is 200 kgf/cm^2, while the initial bubble point pressure of the oil phase is 100 kgf/cm^2. The hydrocarbon phases are treated as black-oil fluids and their FVF, viscosity and solubility ratios are characterized through PVT tables. For the oil phase, at the initial reservoir pressure, these values are defined as 1.31 m^3/m^3, 1.03 cP and 105.96 m^3/m^3, respectively.

The well configuration consists of a five-spot pattern, with a single production well on each corner of the reservoir and an injection well at its very center; as seen in the topside view depicted in Figure 5.2. Every well is completed and perforated exclusively on the top layer of the reservoir. Additionally, all of the wells are controlled by their respective bottom-hole pressures (BHP), with the producers set to operate at 195 kgf/cm^2 and the injector limited to 205 kgf/cm^2. No further restrictions or boundary conditions were imposed for the simulations.

PUC-Rio - Certificação Digital Nº 1612782/CA

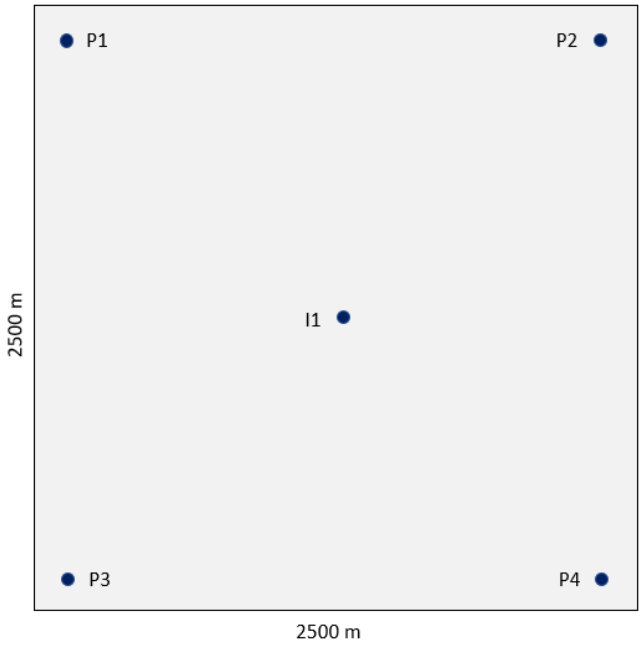

Figure 5.2 – Reservoir and well configuration viewed from the top.

Computer Configuration

The hardware used for the simulations was an Intel Core i7-6700 CPU comprised of 4 cores, each with 8 logical processors running at 3.40 GHz, and using

32 GB of RAM. In addition, the simulator software was run in a Windows 10 operating system with 64-bit architecture (x64 processor), and was compiled using Intel's C++ 19.0 Compiler.

It is important to note, however, that since neither the simulator code nor the *solvers* have yet been programmed to operate in parallel (employing multiple processors), the simulations have basically made use of a single CPU per run. Exceptions are some of the mathematical routines, such as vector norms and additions, which have been designed to operate using multiple threads.

Description of the Simulations

The simulations performed using the model previously described sought to investigate the algorithms' capacity to solve the linear equations assembled by the simulator as the dimensions of the problem grew in size, and to compare the speed of the different algorithms with respect to one another. This was accomplished by gradually increasing the number of grid blocks in which the reservoir was partitioned, thus increasing the total cell count, and consequently the number of equations to be solved simultaneously. The problem sizes selected are presented in Table 5.2, together with the number of divisions that were adopted in each direction. For these simulations every cell was considered to be active. Since the simulator is being developed using a fully implicit black-oil formulation, there are three degrees of freedom per cell; therefore, the total number of equations and, consequently, of degrees of freedom to be solved simultaneously shall be three times the total number of elements.

PUC-Rio - Certificação Digital Nº 1612782/CA

Table 5.2 – List of the problem sizes selected and the corresponding grid dimensions adopted.

Number of Elements	10,000	20,000	50,000	100,000	200,000
Grid Dimensions	35x30x10	45x45x10	70x70x10	100x100x10	100x100x20
Number of Degrees of Freedom	30,000	60,000	150,000	300,000	600,000

Furthermore, the simulations were set to run for a fixed number of 10 time-steps in each case. The results presented have been normalized per time-step, so as

to form a basis from which to estimate the duration of longer simulation scenarios. To verify that this strategy would not produce inconsistent or biased results, one simulation of a coarser problem (with approximately 6000 grid blocks) was run for a full year using the different *solvers*. The results indicated that the performance of the *solvers* relative to one another did not vary significantly as the simulation progressed from the early time-steps to the latter ones. This is depicted in Table 5.3, which shows the relative runtimes of the various *solvers* in the initial and final time-steps (using the runtime of the fastest configuration as the base time), as well as the variation observed in their relative performances when comparing these two periods.

Part of the results of this full year simulation with GSim are presented in Figure 5.3, alongside the results of an identical simulation performed with the commercial software IMEX from CMG (IMEX, 2018). The attributes shown are the production curve of the P1 producer well and injection curve of the I1 injector well as a function of time. The purpose of this comparison between simulators is simply to validate that the one used for this research has results that are compatible to those of a field-proven simulator.

PUC-Rio - Certificação Digital Nº 1612782/CA

Table 5.3 – Comparison of the *solvers'* relative performance in the early and late time-steps.

Iterative Method	First 5 Time-Steps		Last 5 Time-Steps		Relative Runtime Variation
	Relative Runtime	Relative Position	Relative Runtime	Relative Position	
BiCGSTAB_ILU	1.8703	3	1.7179	3	-8.15%
BiCGSTAB_CPR	2.1293	4	2.2939	4	7.73%
GMRES_ILU	2.8764	6	2.5287	6	-12.09%
GMRES_CPR	2.1362	5	2.4386	5	14.16%
ORTHOMIN_ILU	1.0000	1	1.0000	1	0.00%
ORTHOMIN_CPR	1.0751	2	1.0746	2	-0.05%

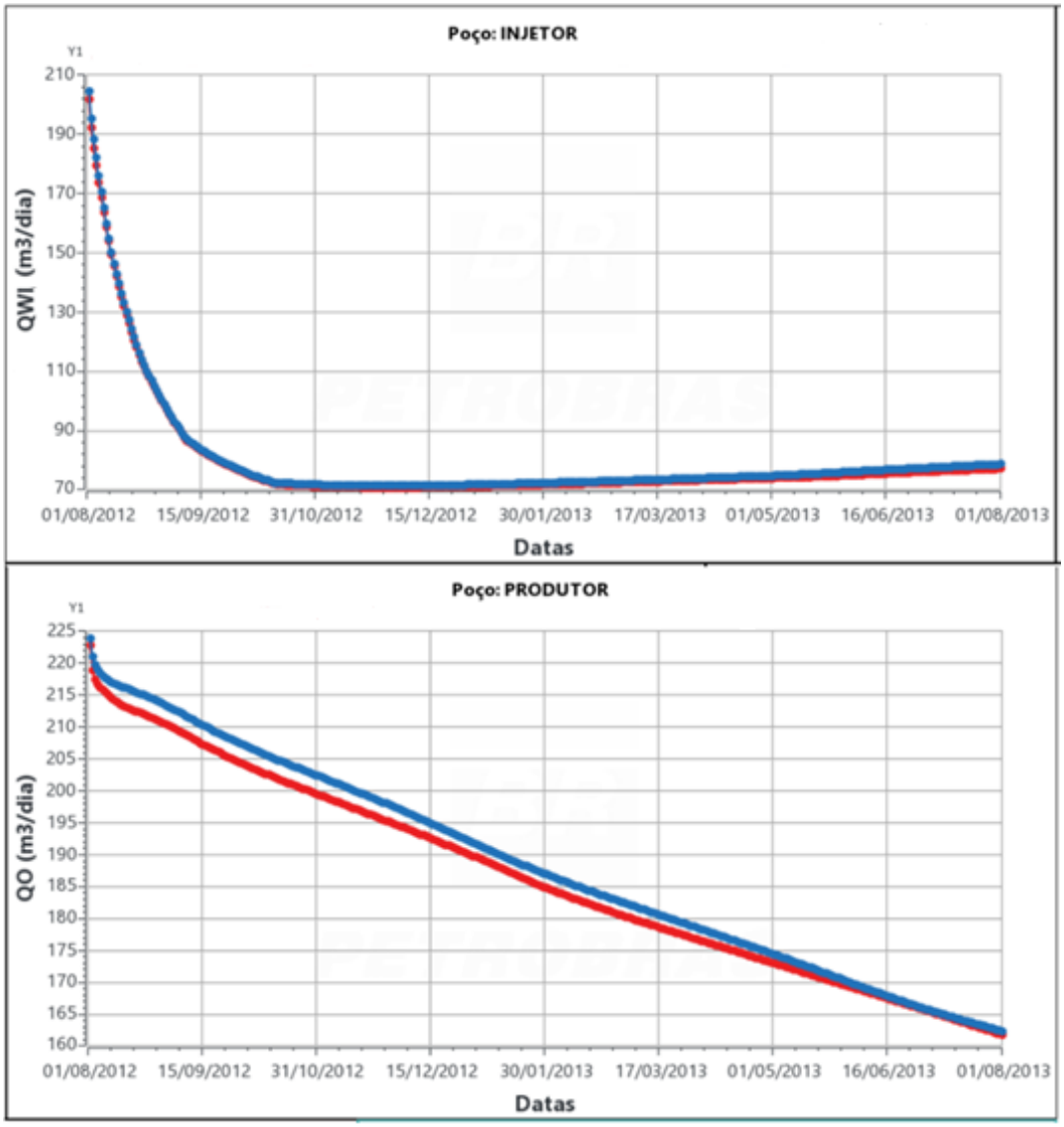

Figure 5.3 – Results for a one-year simulation using both IMEX (red curves) and GSim simulators (blue curves). The top graph represents water injection rate in well I1, while the bottom graph represents oil production rate in well P1.

Finally, because the performance of the *solvers* was not uniform, with some having relatively better or worse results than others, not all of the *solvers* were employed for every problem size. Whenever a *solver* configuration was deemed as being consistently inferior to its counterparts it ceased to be evaluated.

5.2
Numerical Results

For each of the problems considered, several attributes of the simulations were assessed. These will now be presented in this section.

PUC-Rio - Certificação Digital Nº 1612782/CA

The first aspect to be analyzed is the total simulation runtime (clock-time) measured for each *solver* configuration, in each grid size. This is depicted in Figure 5.4, normalized per time-step. The time durations reported here encompass all of the tasks the simulator must perform throughout the course of a simulation, with the main ones being the construction of a linear system of equations at every nonlinear (Newton) step of each time-step, and then the solution of these equations by means of the iterative *solvers*. As a benchmark, Intel's commercial direct *solver* Pardiso is also included in the performance assessment (Schenk and Gärtner, 2018). Pardiso is a very efficient *solver* that is designed to operate proficiently using parallel cores. This characteristic makes it distinct from conventional direct factorization methods designed for general sparse matrices, which tend to have their performances degraded when applied in a parallel manner (Davis and Duff, 1997).

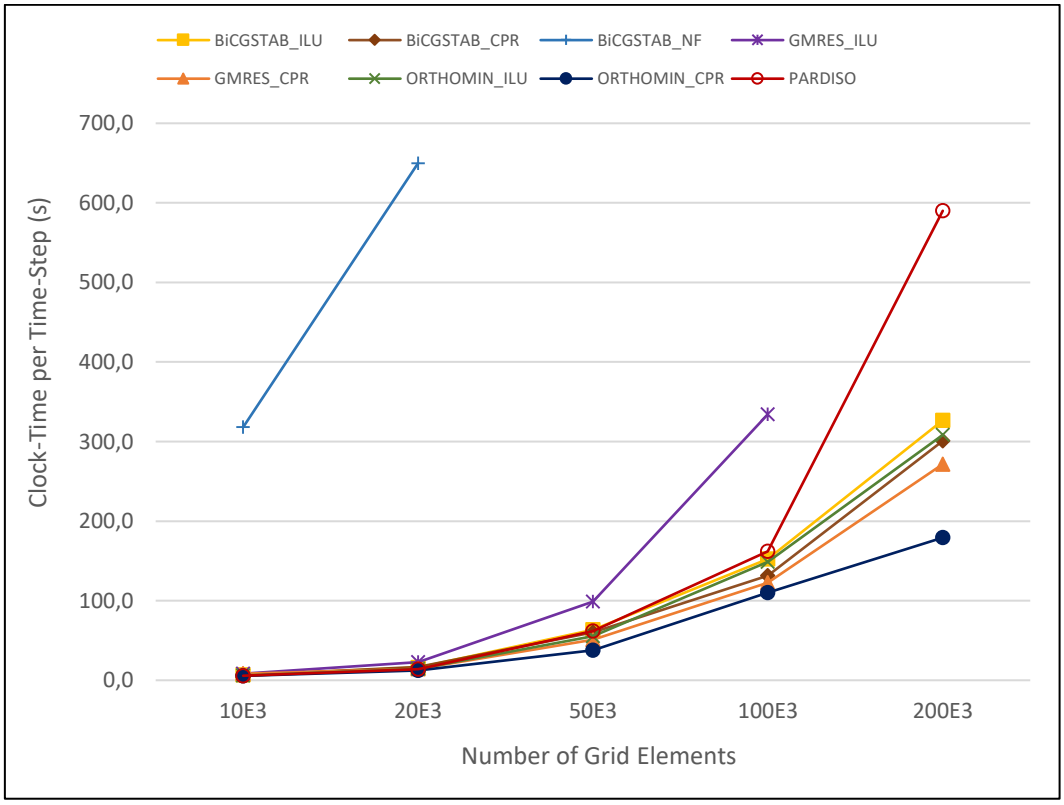

Figure 5.4 – Total simulation runtime per time-step with each iterative *solver* as function of grid size.

It can be seen here that from the very start the Nested Factorization preconditioner presented results that were an order of magnitude slower than the ILU and CPR methods. Because of this, only a few results are shown with this

PUC-Rio - Certificação Digital Nº 1612782/CA

preconditioner, since the *solver* configurations that included it were not competitive with relation to the remaining ones.

The reason for the markedly inferior results obtained with NF is not evident. It could perhaps be attributed to the implementation of the algorithm, and not to the technique itself, which has been proven very effective in commercial simulators in the past. However, it might also be related to the actual model being tested. Due to its isotropic properties and to the fact that the cells' horizontal dimensions are much larger than their vertical ones, the vertical transmissibilities are much greater than the horizontal ones. In addition, the ordering scheme adopted by the simulator in constructing the linear equations was natural order, with the numbering starting in the x-direction and then advancing in the y-direction and, finally, z-direction. Because of this, the coefficients representing interactions between planes – that is, between layers in the z-direction – have higher values than those representing interactions between lines or within lines. The issue with this is the fact that the interplane coefficients are positioned in the outer diagonals (L_3 and U_3), while NF presents much better convergence whenever the smallest coefficient values are outermost and the largest ones innermost. Nested Factorization is actually quite sensitive to the ordering of the axes, and tests performed by Appleyard and Cheschire (1983) and by Behie (1985) show cases where variations in ordering represented an increase of as much as 5 times in the number of iterations required, and in some scenarios even led to divergence. In any case, this would have to be investigated further so as to establish the true cause; unfortunately, though, time restrains prevented this from being accomplished in the course of this research.

Furthermore, the performance of the *solver* composed of GMRES preconditioned with ILU also proved to be consistently worse than the others, and was already demanding 4 times as much clock-time as the second worse method when the grid size reached 50,000 cells. Therefore, it was also discarded as a candidate for optimal configuration, and shall not be included in some of the subsequent analyses. In addition, this serves as an indicator that ILU may not be as robust as CPR, due to a stronger dependency on the iterative method to which it is coupled.

Figure 5.4 also reveals that, with the exception of the aforementioned methods, all the remaining ones were capable of delivering faster results than the

PUC-Rio - Certificação Digital Nº 1612782/CA

Pardiso *solver*. This result is compatible with the performances observed by Chen et al. (2006) with respect to banded Gaussian Elimination and indicates that, for reservoir problems of considerable size, iterative *solvers* are extremely competitive when compared to direct ones. The ratio of the clock-time required by the iterative *solvers* with respect to Pardiso is presented in Figure 5.5, corresponding to the results seen on the most refined grid, with 200,000 cells. Considering all cases, the improvement obtained with the iterative *solvers* ranged from 2.5 up to 20 times the speed of the direct *solver*.

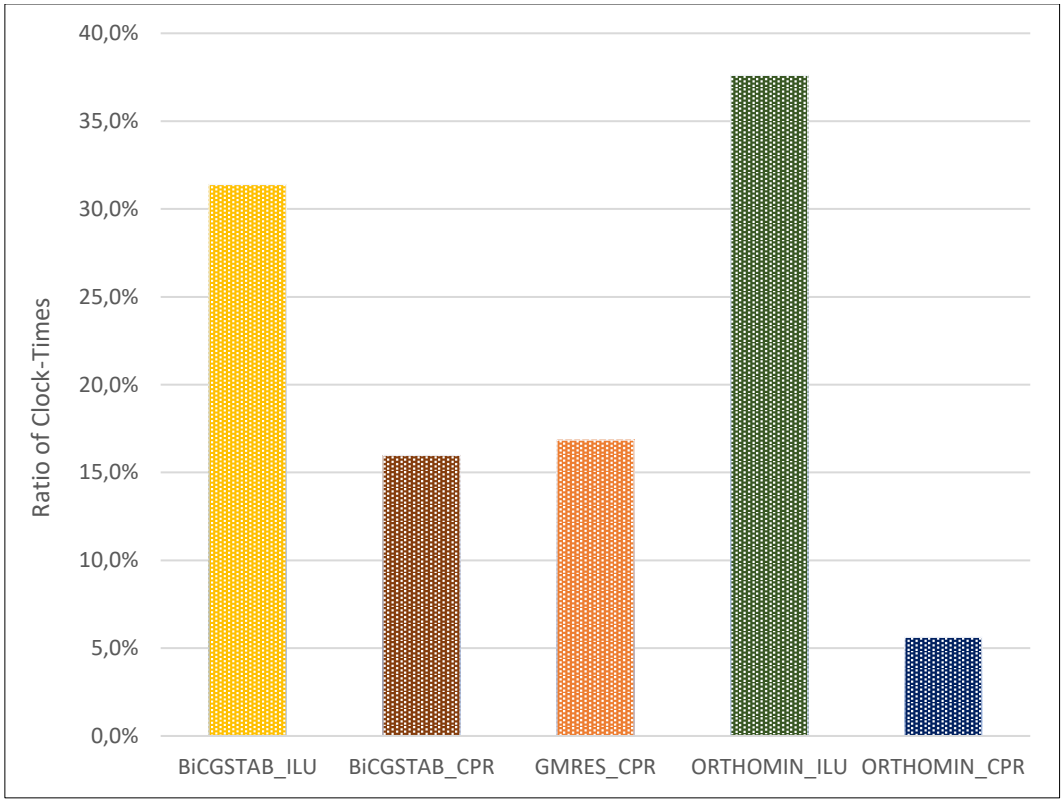

Figure 5.5 – Ratio of clock-times between the iterative *solvers* analyzed in relation to the direct *solver* Pardiso.

Focusing now on the comparison between the iterative *solvers* themselves, Figure 5.6 presents the total clock-time expended per time-step by each of them in completing all of their inner functions, comprised mainly of constructing a preconditioner at every Newton step and then of finding the solution to the system of equations. As described previously, the solution stage of the *solvers* includes both the operations performed by the iterative methods, as well as one or more calls

PUC-Rio - Certificação Digital Nº 1612782/CA

to the solution routines of the preconditioning methods, for every linear iteration performed.

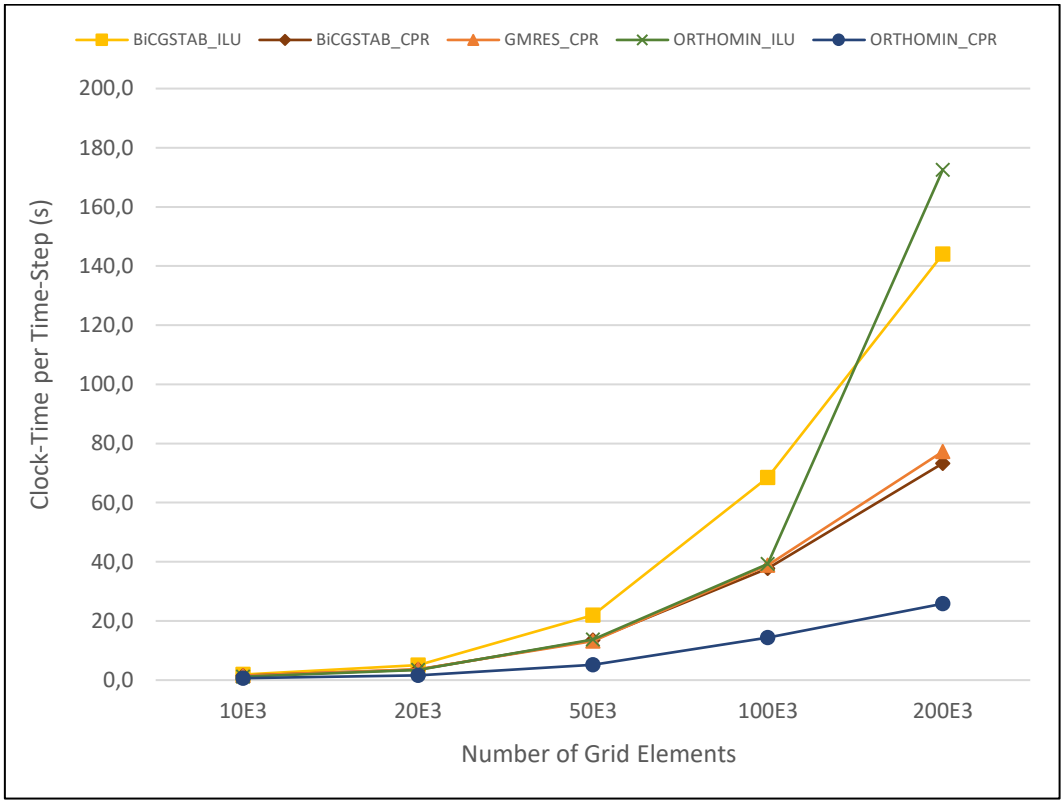

Figure 5.6 – Total clock-time required per time-step by each iterative *solver* as function of grid size.

The behavior of the curves in Figure 5.6 demonstrates that the *solver* combining the ORTHOMIN iterative method with the CPR preconditioner consistently delivers superior results, when compared to the remaining configurations. On the finest grid, the time consumed by the other *solvers* to reach convergence was up to 7 times higher than this combination, as illustrated in Figure 5.7. Moreover, the coupling of CPR with the two other iterative methods (BiCGSTAB and GMRES) produces results similar to one another, and both tend to be faster than methods preconditioned with simple ILU. This speedup seen in iterative methods preconditioned with CPR over those preconditioned with ILU is very similar to the results observed by Stüben et al. (2007) and Gries et al. (2013), in tests performed with various models using FGMRES (Saad, 2003).

Furthermore, for this problem, given a fixed preconditioner (CPR or ILU), the ORTHOMIN iterative method tends to be consistently faster than BiCGSTAB

PUC-Rio - Certificação Digital Nº 1612782/CA

(except with ILU on the 200,000 elements grid), which in turn tends to be faster than GMRES.

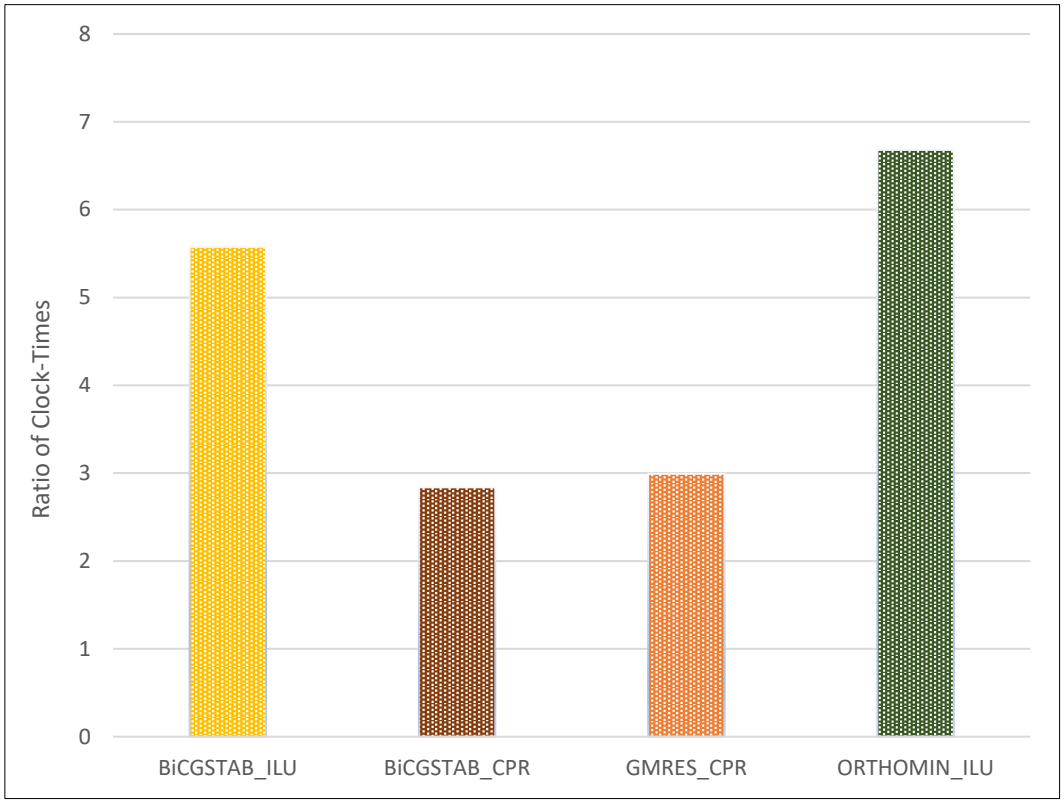

Figure 5.7 – Ratio of the different *solver* runtimes with respect to the runtime of ORTHOMIN_CPR.

The runtime required by each *solver* also has a very strong correlation to the number of linear iterations necessary for the iterative methods to converge. Figure 5.8 shows the average iteration count performed per Newton step before convergence was attained. The explanation for this behavior may be visualized in Figure 5.9, which presents the residual reduction profile of the various methods throughout an iterative process. It shows how the slope of residual reduction advances much more aggressively towards convergence in the case of CPR than with ILU.

It becomes evident from these images that the power of CPR arises from its capacity to significantly reduce the number of linear iterations required to solve the systems of equation, an order of magnitude or more below the number needed by ILU. This is further exemplified in Figure 5.10, which depicts the ratio between the number of iterations performed by the various *solvers* preconditioned by ILU, with

PUC-Rio - Certificação Digital Nº 1612782/CA

respect to their versions preconditioned by CPR. The data was taken from the 100,000 elements grid and shows that whenever the iterative methods were preconditioned with ILU, they demanded anywhere from 5 to 20 times the number of iterations when compared to CPR. These results are compatible to those observed by Brown et al. (2015), where factors greater than 10 were also detected.

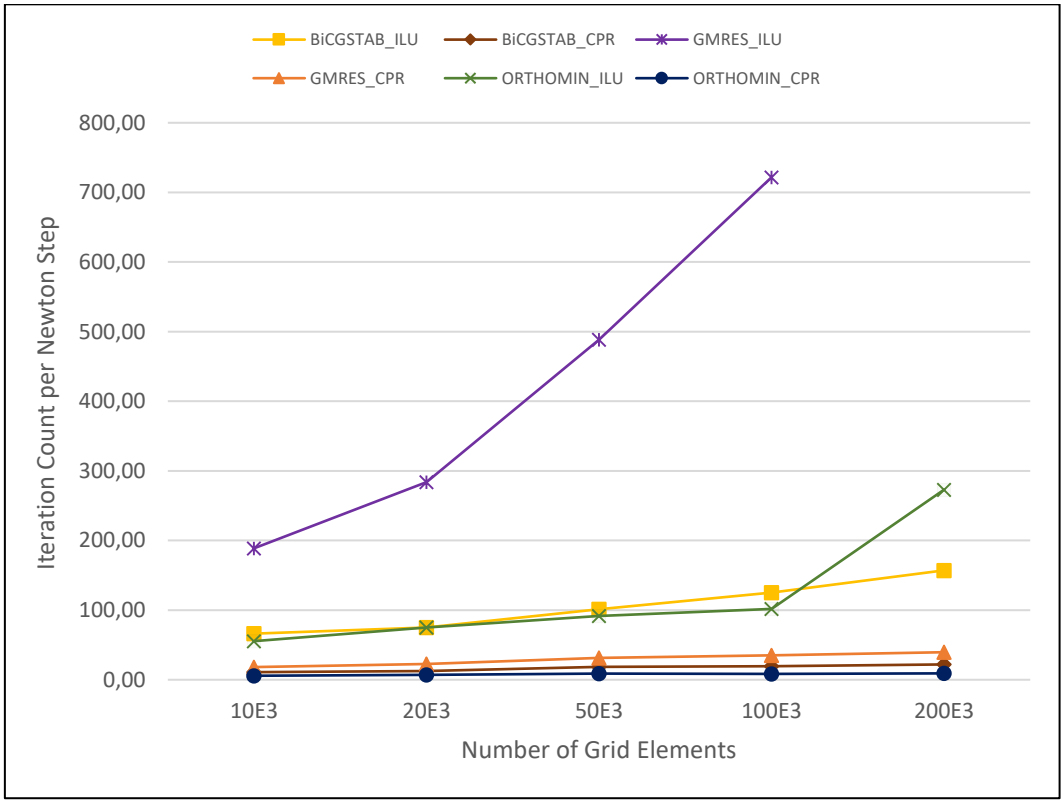

Figure 5.8 – Iteration count per Newton step for the different *solvers* as a function of grid size.

PUC-Rio - Certificação Digital Nº 1612782/CA

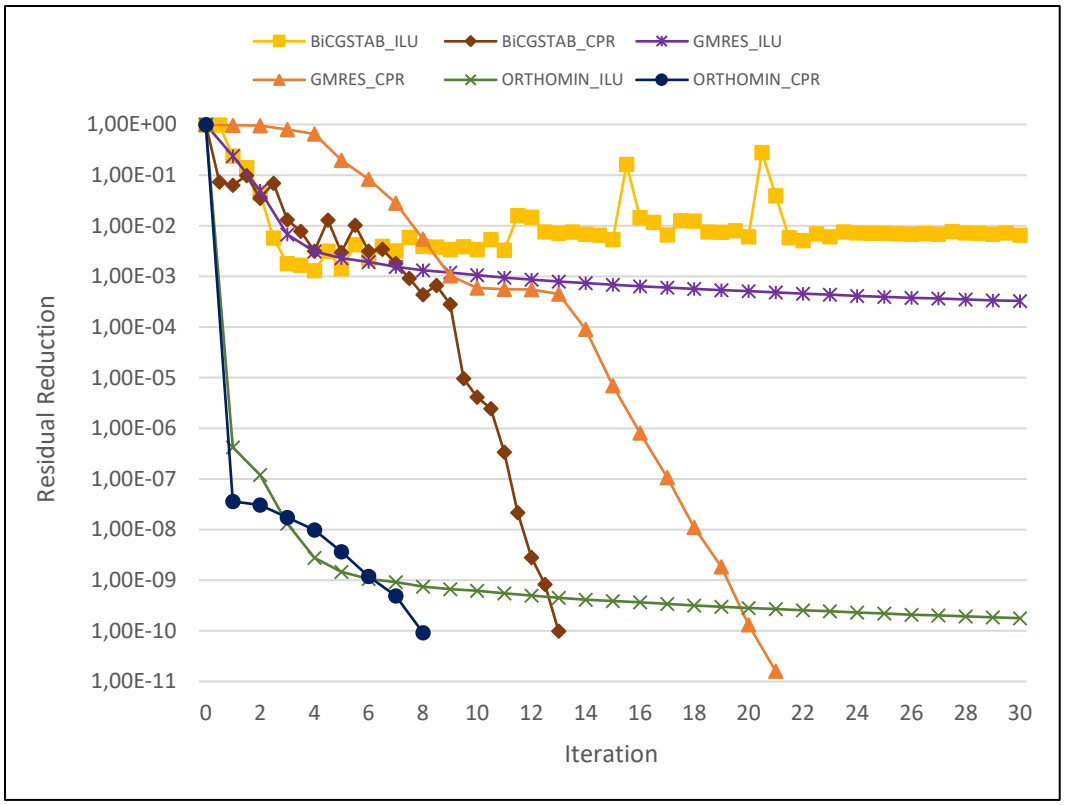

Figure 5.9 – Residual reduction behavior of the various preconditioned iterative methods.

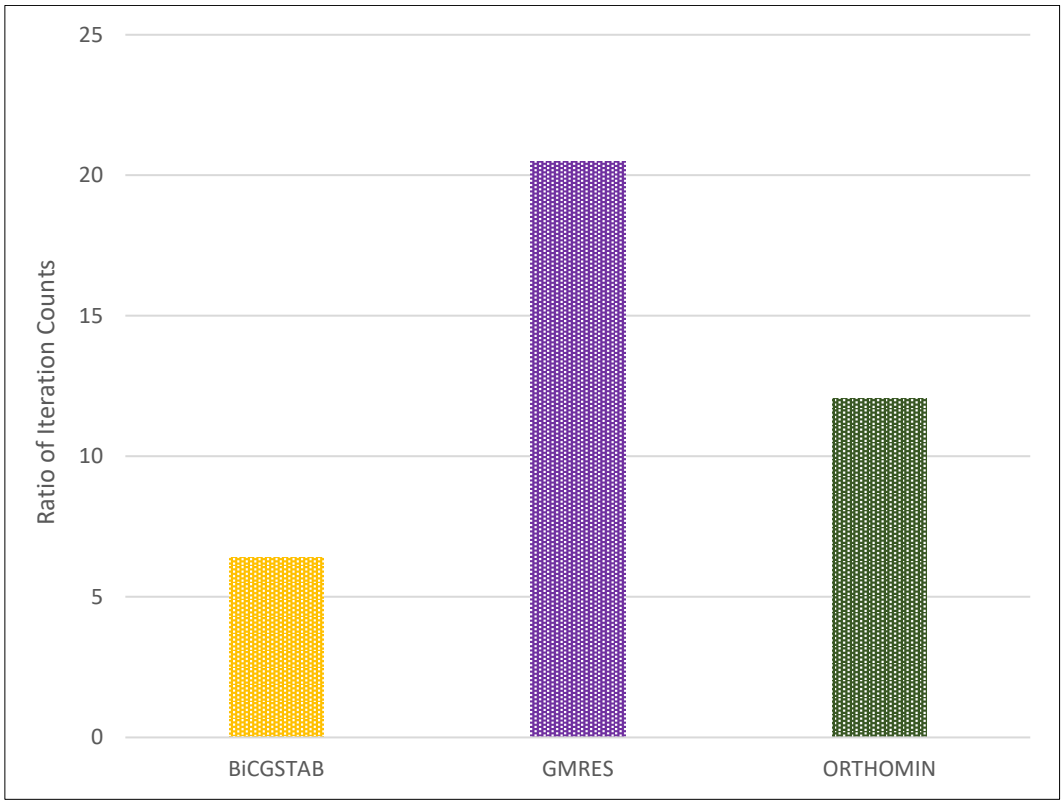

Figure 5.10 – Iteration count ratio of methods preconditioned with ILU over CPR.

PUC-Rio - Certificação Digital Nº 1612782/CA

Another analysis pertaining to Figure 5.6 and Figure 5.8 concerns the fact that, for a given preconditioner, the time expended per step by the various iterative methods may differ substantially. This observation is summarized in Table 5.4 and Table 5.5, which present the ratio of the runtimes consumed per iteration by the solution stages of the *solvers*, when preconditioned via CPR and ILU, respectively.

Table 5.4 – Ratio of the runtimes to perform each iteration of the different methods, when preconditioned with CPR.

Ratio	Grid Size					Average
	10E3	20E3	50E3	100E3	200E3	
ORTHOMIN/GMRES	1.095	1.078	1.078	1.084	1.018	1.071
BiCGSTAB/GMRES	1.698	1.753	1.770	1.752	1.693	1.733
BiCGSTAB/ORTHOMIN	1.551	1.626	1.642	1.616	1.663	1.620

Table 5.5 – Ratio of the runtimes to perform each iteration of the different methods, when preconditioned with ILU.

Ratio	Grid Size					Average
	10E3	20E3	50E3	100E3	200E3	
ORTHOMIN/GMRES	1.121	1.126	1.119	1.074	-	1.110
BiCGSTAB/GMRES	1.547	1.612	1.617	1.558	-	1.583
BiCGSTAB/ORTHOMIN	1.380	1.431	1.445	1.450	1.421	1.426

It can be observed from these that GMRES appears to have the fastest iterations, with its results being very close to those of ORTHOMIN, and both being notably faster (per iteration) than BiCGSTAB.

The reason for this can be traced to the computational costs introduced in Table 3.1 and Table 3.2. If we consider that the coefficient matrices ($R^{m\times m}$) constructed for this reservoir are very sizable and block heptadiagonal in form, it is simple to deduce that it will have, on average, approximately 20 non-zeros per row. A matrix-vector product (MV) in this case will closely equate to 20 vector-vector products, such as dot products or norm calculations (DOT), in terms of floating-point multiplications and additions. If we now bear in mind that vector-vector products consist of m multiplications and $(m-1)\approx m$ additions, and that operations of the type AXPY consist of one vector product by a scalar (m

PUC-Rio - Certificação Digital Nº 1612782/CA

multiplications) and one vector sum (m additions), the operation count for each iterative method can be estimated according to Table 5.6. The actual number of operations will be equal to the values indicated there, times the number of problem unknowns m. These values were projected assuming the reset parameters of GMRES(k) and ORTHOMIN(k) to be constant and equal to their adopted upper limits (which is the worst-case scenario) of 50 and 30 iterations, respectively.

Table 5.6 – Operation count per degree of freedom for each iterative method.

	Operation Count		
Iterative Method	Addition	Multiplication	Preconditioner
GMRES	72	72	1
ORTHOMIN	75	75	1
BiCGSTAB	52	52	2

Referring to this table and to the runtime ratios seen previously, it is apparent that the preconditioning operation of solving the system

$$\boldsymbol{q} = \boldsymbol{M}^{-1}\boldsymbol{r} \tag{5.1}$$

comprises a sizable portion of the time required for every step of the iterative method.

Even though the BiCGSTAB algorithm has only 70% of the number of floating-point multiplications and additions when compared to ORTHOMIN or GMRES (which in turn differ between themselves by only 5%), the fact that it must execute one extra time the preconditioning solution routine causes its time consumption to rise appreciably. Therefore, for this method to be competitive, either the preconditioning solution must not be too costly or, otherwise, the preconditioner must result in many less iterations being necessary. This was the case when we compare BiCGSTAB to GMRES, but not so when measured against ORTHOMIN, not even when considering the less expensive ILU preconditioner. This conclusion can be visualized through the schematic in Figure 5.11. It is also demonstrated in Table 5.7, which presents the relative weight of the preconditioner solution routine with respect to the total iterative method runtime, for each of the preconditioners considered.

PUC-Rio - Certificação Digital Nº 1612782/CA

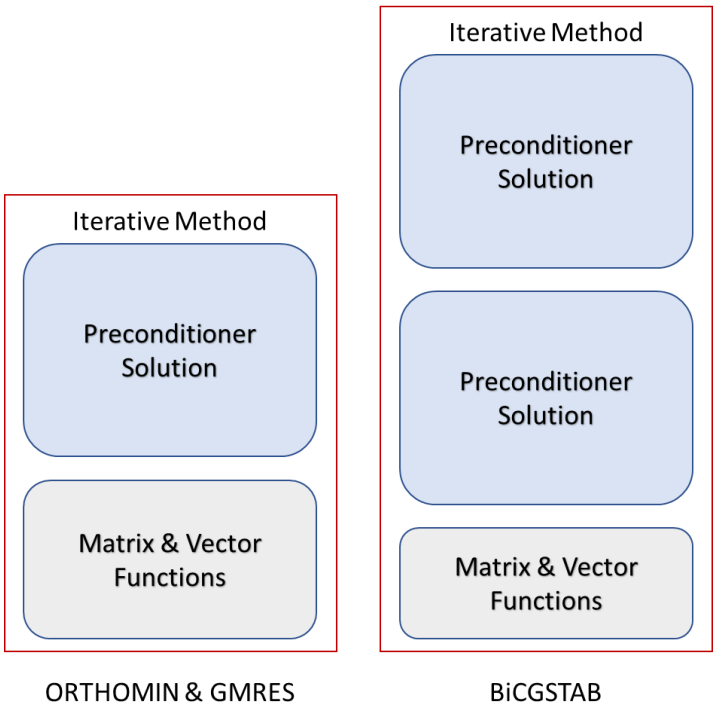

Figure 5.11 – Comparison of the potential impact of the preconditioner operations in the total runtime of the iterative method (solution stage of the *solver*).

Table 5.7 – Average time consumption of the preconditioning solution routines for each iterative method, as a percentage of the total *solver* solution stage.

Iterative Method	Preconditioner	
	CPR	ILU
GMRES	90.90%	55.10%
ORTHOMIN	87.70%	55.10%
BiCGSTAB	91.50%	66.11%

Another aspect of any preconditioner that may also cause a significant impact in the total *solver* runtime is its construction. Ideally, preconditioners should be inexpensive to build and its construction time should scale more or less linearly with problem size; otherwise, the time consumed assembling them might outweigh their benefit to the underlying iterative method.

The behavior of the construction time of CPR and ILU with respect to problem dimension can be seen in Figure 5.12. Although there is a nonlinear

PUC-Rio - Certificação Digital Nº 1612782/CA

increase in runtime early on, with the coarser grids, the tendency of the curves seemed to stabilize as the problem grew in size.

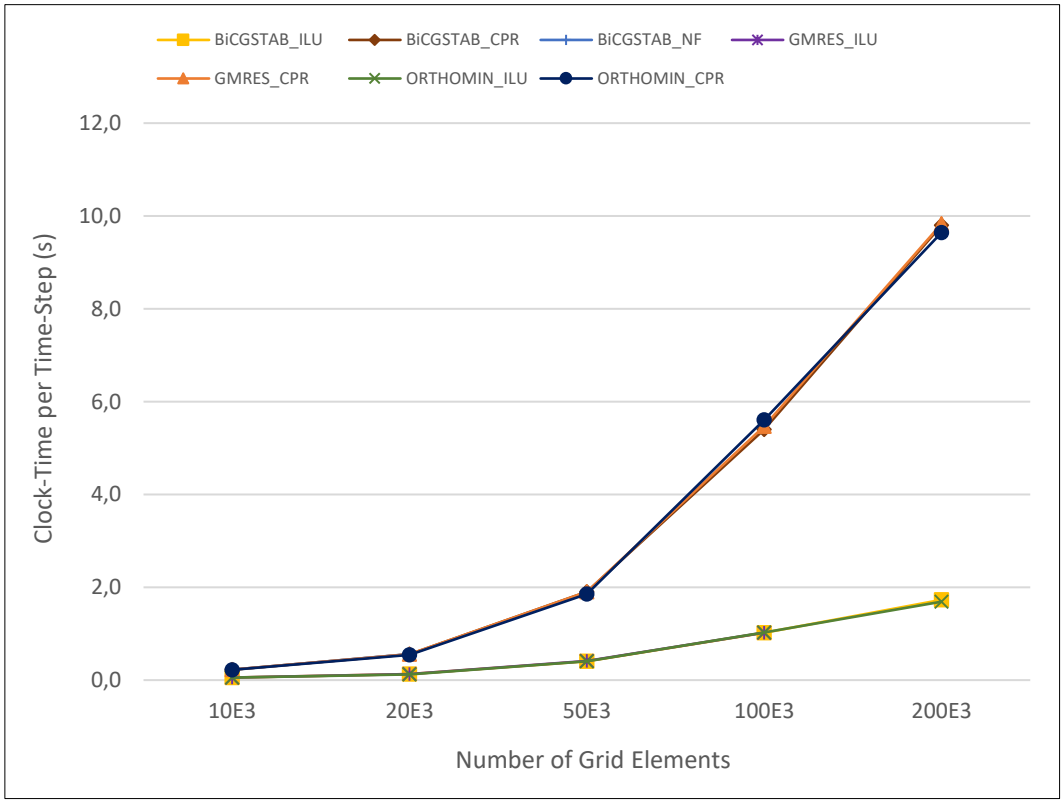

Figure 5.12 – Construction time of CPR and ILU preconditioners as a function of grid size.

In any case, considering the proportional time demanded by the construction stage of the preconditioner with respect to the total *solver* runtime, as demonstrated in Figure 5.13, the values observed can be considered to be very reasonable.

PUC-Rio - Certificação Digital Nº 1612782/CA

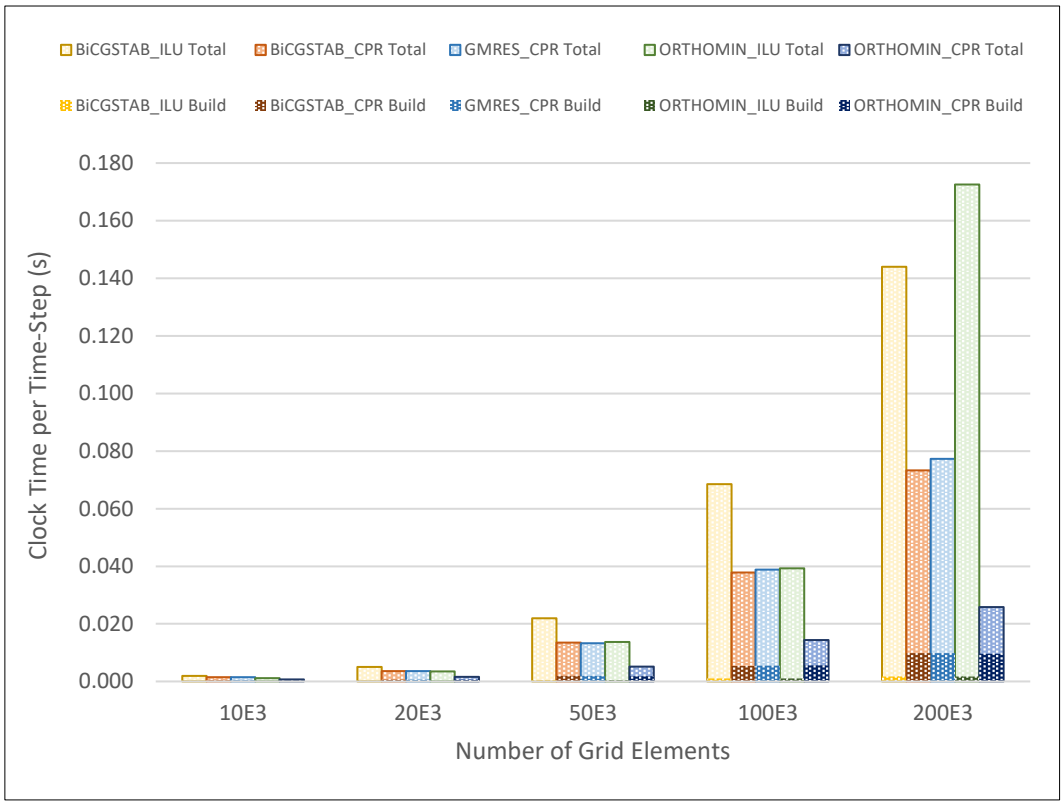

Figure 5.13 – Decomposition of the *solvers*' total runtime, in terms of the setup (i.e. preconditioner construction – dark colors) and solution (i.e. iterative method – light colors) stages.

The proportion of construction time to total time observed in Figure 5.13, and detailed in Table 5.8, remains relatively constant with respect to problem dimension, which supports the conclusion that they are scaling well.

Table 5.8 – Ratio of the time required by the preconditioner construction relative to the total *solver* runtime.

	Grid Size					
Solver	10E3	20E3	50E3	100E3	200E3	Average
BiCGSTAB_ILU	2.88%	2.50%	1.86%	1.49%	1.20%	1.99%
BiCGSTAB_CPR	14.89%	15.62%	14.06%	14.28%	13.37%	14.45%
GMRES_ILU	1.60%	1.08%	0.63%	0.41%	-	0.93%
GMRES_CPR	15.15%	15.36%	14.37%	14.08%	12.72%	14.34%
ORTHOMIN_ILU	4.63%	3.55%	2.93%	2.62%	0.98%	2.94%
ORTHOMIN_CPR	33.75%	34.49%	36.11%	39.05%	37.34%	36.15%

PUC-Rio - Certificação Digital Nº 1612782/CA

The final study undertaken involved stressing the ORTHOMIN_CPR *solver* to higher limits. Since this combination of iterative method and preconditioner had been observed to be a very promising *solver* configuration, it was decided to test its robustness even more. This was done by further discretizing the reservoir grid. The simulator was thus set to run using grid sizes of 500,000 and 1,000,000 elements (corresponding to 1,500,00 and 3,000,000 degrees of freedom, respectively), with 160x160x20 and 225x225x20 divisions in each direction, respectively.

The result of these simulations is presented in Figure 5.14, which shows that the *solver* continued to be capable of reaching convergence. Although the required runtime naturally increased in conjunction with problem dimension, it is interesting to notice that this increase seems to be of reasonable magnitude. This conclusion stems from the observation that the performance of iterative methods can normally be expected to scale in the order of $O(m^2)$ (Trefethen and Bau III, 1997); therefore, if we calculate the proportion between runtimes of different problems and also calculate the proportion of their dimensions squared, the ratio of these numbers would be expected to approach unity (or at least to approach a relatively constant scalar value):

$$\frac{\left({}^{t_2}/_{t_1}\right)}{\left({}^{m_2}/_{m_1}\right)^2} \approx 1 \qquad (5.2)$$

Moreover, ratios below this value would be an indication that the *solver* is scaling well with respect to problem dimension. When this parameter is computed for the runtimes seen with ORTHOMIN_CPR, the numbers remain below this threshold for the entire range of simulations, as seen in Figure 5.14.

PUC-Rio - Certificação Digital Nº 1612782/CA

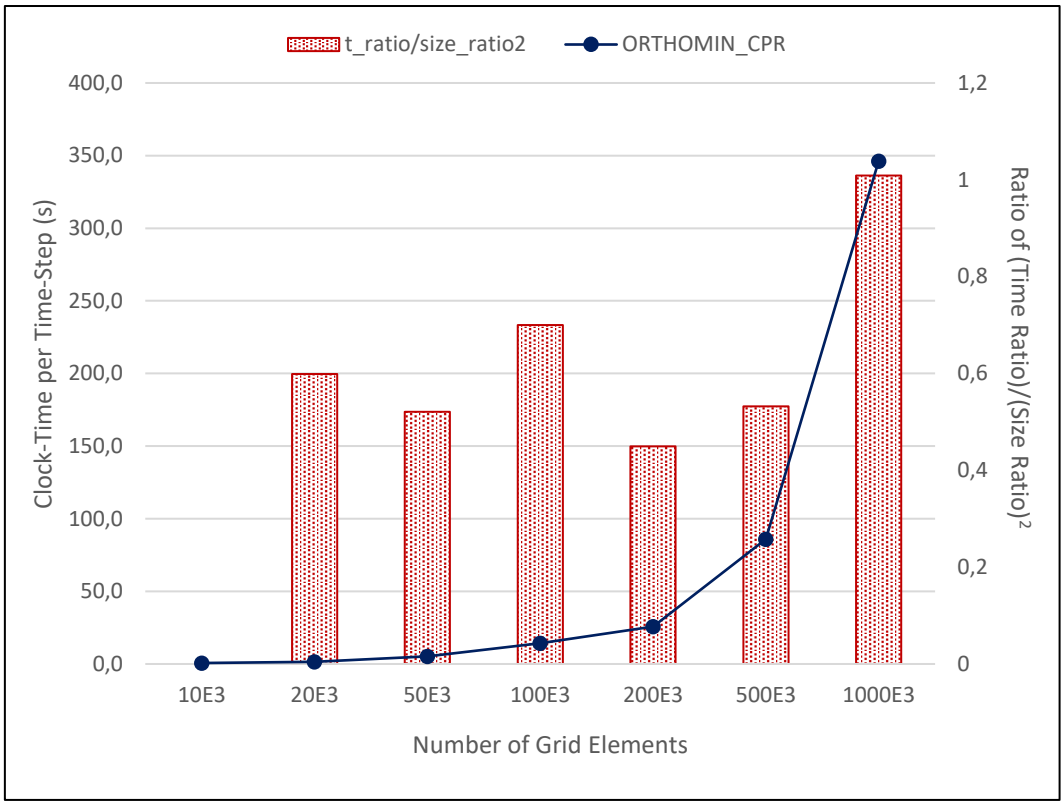

Figure 5.14 – Total *solver* runtime required by ORTHOMIN_CPR as a function of grid size, and relative rate of increase in required time in relation to the increment in problem size.

PUC-Rio - Certificação Digital Nº 1612782/CA

6
Final Considerations

Reservoir simulation is an essential activity in the development of hydrocarbon fields, because it offers insight into future reservoir performance under distinct operational conditions and well placements. However, to be of practical use, a simulator must be capable of delivering accurate results in a timely manner which, consequently, requires an appropriate numerical *solver*.

This thesis investigated various iterative *solvers* designed for reservoir simulation applications. These *solvers* combine an iterative method with a preconditioning technique to gradually approach the solution to linear systems of equations constructed by a reservoir simulator. The iterative method is responsible for defining a strategy on how the search space for the solution will be created, while the preconditioner helps optimize the search directions undertaken in that space.

Identifying robust and efficient *solvers* suited for a reservoir simulator was the primary objective of this work. Robust because it must be able to converge on a solution for a wide range of problems, since a developer does not know beforehand which models will be studied in his simulator. Efficient because it must deliver the results within a reasonable timeframe to be of use in decision-making processes during the development and management a field.

PUC-Rio - Certificação Digital Nº 1612782/CA

6.1
Conclusions

The tests completed for this thesis assessed the performance of *solvers* comprised of the BiCGSTAB, GMRES and ORTHOMIN iterative methods and ILU, NF and CPR preconditioners, considering the multiple combinations possible. This involved employing them to solve a model reservoir problem, using several different grid dimensions, and analyzing diverse aspects of the results.

These experiments indicated that the *solver* consisting of ORTHOMIN preconditioned with CPR provided the best overall results amongst all

configurations, for the reservoir model investigated. It proved to be both robust, capable of solving problems with over one million grid blocks, and efficient, delivering solutions 3 to 7 times faster than the next most competitive iterative *solvers*, and 20 times faster than Intel's commercial *solver* Pardiso.

Furthermore, the tests revealed that the CPR preconditioner serves as an excellent option for simulator *solvers*. It consistently outperformed the remaining preconditioners, in terms of reducing the number of iterations necessary for achieving convergence, and proved to scale well with problem size, even when considering its relatively costly construction stage.

Moreover, ORTHOMIN seemed to be a very competitive iterative method, especially when coupled with CPR. BiCGSTAB also displayed promising results due to the fact that it most consistently kept the number of required iterations low. However, the runtime cost per iteration of this method can be significantly superior to those of the other methods, primarily because of an extra preconditioning solution operation that it must carry out. Therefore, it must either be able to converge on even fewer iterations than the remaining methods, or be coupled with a preconditioner that consumes less clock-time for its solution stage.

PUC-Rio - Certificação Digital Nº 1612782/CA

6.2
Suggestions for Future Research

The field of numerical *solvers* is so rich and challenging that there are uncountable additional aspects that could be further explored by future research. Nonetheless, some of the most promising and interesting ones will be mentioned here: (i) study of the effect of different ordering schemes in the *solvers'* performance; (ii) implementation of new versions of the algorithms studied, for them to operate in parallel using multiple machines (clusters) and multiple cores (Collins et al., 2013); (iii) implementation of new versions of the algorithms to run on GPU instead of CPU (Appleyard et al., 2014; Zhou and Tchelepi, 2013).

Furthermore, any future research initiated based on the algorithm codes implemented for this thesis should first further validate the results observed herein by testing additional reservoir models with more complex geometries and property distributions, as well as test the appearance and disappearance of the gas phase, once the reservoir simulator is fully operational.

7
Bibliographical References

AL-SHAALAN, T. M. et al. **Studies of Robust Two Stage Preconditioners for the Solution of Fully Implicit Multiphase Flow Problems**. Proceedings from the SPE Reservoir Simulation Symposium held in The Woodlands, Texas. Society of Petroleum Engineers (SPE), 2009. SPE 118722.

APPLEYARD, J. R. et al. **Accelerating Reservoir Simulation Using GPU Technology**. Proceedings from the SPE Reservoir Simulation Symposium held in The Woodlands, Texas. Society of Petroleum Engineers (SPE), 2014. SPE 141402.

APPLEYARD, J. R.; CHESHIRE, I. M. **Nested Factorization**. Proceedings from the Reservoir Simulation Symposium held in San Francisco, California. Society of Petroleum Engineers of AIME, 1983. SPE 12264.

AZIZ, K.; SETTARI, A. **Petroleum Reservoir Simulation**. Applied Science Publishers Ltd., 1979. ISBN: 0-85334-787-5.

BAKER, A. H.; JESSUP, E. R.; KOLEV, T. V. **A Simple Strategy for Varying the Restart Parameter in GMRES(m)**. Document prepared as an account of work sponsored by an agency of the United States Government. Journal of Computational and Applied Mathematics. Lawrence Livermore National Library, 2007.

BARRETT, R. **Templates for the Solution of Linear Systems**: Building Blocks for Iterative Methods. 2nd ed. In: Other Titles in Applied Mathematics Series. Society for Industrial and Applied Mathematics (SIAM), 1994. ISBN: 978-0-89871-328-2.

BEHIE, A**. Comparison of Nested Factorization, Constrained Pressure Residual and Incomplete Factorization Preconditionings**. Proceedings from the

PUC-Rio - Certificação Digital Nº 1612782/CA

SPE Middle East Oil Technical Conference and Exhibition in Bahrain. Society of Petroleum Engineers (SPE), 1985. SPE 13531.

BEHIE, A.; COLLINS, D.; FORSYTH, P. **Incomplete Factorization Methods for Three-Dimensional Nonsymmetric Problems**. Computer Methods in Applied Mechanics and Engineering. n. 42. pp. 287-299. Elsevier, 1984.

BEHIE, A.; VINSOME, P. K. W. **Block Iterative Methods for Fully Implicit Reservoir Simulation**. In: Society of Petroleum Engineers Journal. Society of Petroleum Engineers of AIME, 1982.

BROWN, G. L.; COLLINS, D. A.; CHEN, Z. **Efficient Preconditioning for Algebraic Multigrid and Red-Black Ordering in Adaptative-Implicit Black-Oil Simulators**. Proceedings from the SPE Reservoir Simulation Symposium held in Houston, Texas. Society of Petroleum Engineers (SPE), 2015. SPE 173231.

CAO, H. **Parallel Scalable Unstructured CPR-Type Linear Solver for Reservoir Simulation**. Proceedings from the SPE Annual Technical Conference and Exhibition held in Dallas, Texas. Society of Petroleum Engineers (SPE), 2005. SPE 96809.

CELES, W.; CERQUEIRA, R.; RANGEL, J. L. **Introdução a Estruturas de Dados**: com Técnicas de Programação em C. Rio de Janeiro: Elsevier, 2004. 2ª Reimpressão. ISBN: 85-352-1228-0.

CHEN, Z.; HUAN, G.; MA, Y. **Computational Methods for Multiphase Flow in Porous Media**. Society for Industrial and Applied Mathematics (SIAM), 2006. ISBN: 0-89871-606-3.

CHESHIRE, I. M. et al. **An Efficient Fully Implicit Simulator**. Proceedings from the European Offshore Petroleum Conference and Exhibition, 1980.

PUC-Rio - Certificação Digital Nº 1612782/CA

COLLINS, D. A.; GRABENSTETTER, J. E.; SAMMON, P. H. **A Shared-Memory Parallel Black-Oil Simulator with a Parallel ILU Linear Solver**. Proceedings from the SPE Reservoir Simulation Symposium held in Houston, Texas. Society of Petroleum Engineers (SPE), 2003. SPE 79713.

DAKE, L. P. **Fundamentals of Reservoir Engineering**. In: Developments in Petroleum Science. v. 8. Elsevier, 1978. ISBN: 978-0-444-41830-2.

DAVIS, T. A.; DUFF, I. S. **An Unsymmetric-Pattern Multifrontal Method for Sparse LU Factorization**. In: SIAM J. Matrix Anal. Appl. v. 18. n. 1. pp. 140-158. Society for Industrial and Applied Mathematics (SIAM). 1997.

DAVIS, T. A.; HU, Y. **The University of Florida Sparse Matrix Collection**. ACM Transactions on Mathematical Software. v. V. n. N. pp. 1-28. 2010.

DE STERCK, H. et al. **Distance-Two Interpolation for Parallel Algebraic Multigrid**. Numerical Linear Algebra with Applications. Lawrence Livermore National Library, 2007. Document prepared as an account of work sponsored by an agency of the United States Government.

DUARTE, L. S. et al. **PolyTop++: an Efficient Alternative for Serial and Parallel Topology Optimization on CPU & GPUs**. Springer, 2015.

DUFF, I. S.; GRIMES, R. G.; LEWIS, J. G. **User's Guide for the Harwell-Boeing Sparse Matrix Collection**. 1992. Available on <*https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/*>. Accessed on 29 May, 2018.

ERTEKIN, T.; ABOU-KASSEM, J. H.; KING, G. R. **Basic Applied Reservoir Simulation**. In: SPE Textbook Series. v. 7. Society of Petroleum Engineers (SPE), 2001. ISBN: 1-55563-089-8.

PUC-Rio - Certificação Digital Nº 1612782/CA

GOLDBERG, D. **What Every Computer Scientist Should Know About Floating-Point Arithmetic**. ACM Computing Surveys. v. 23. n. 1., 1991.

GOLUB, G. H.; LOAN, C. F. V. **Matrix Computations**. 3rd ed. The John Hopkins University Press, 1996. ISBN: 0-8018-5413-X.

GRIES, S. et al. **Preconditioning for Efficiently Applying Algebraic Multigrid in Fully Implicit Reservoir Simulations**. Proceedings from the SPE Reservoir Simulation Symposium held in The Woodlands, Texas. Society of Petroleum Engineers (SPE), 2013. SPE 163608.

GRIES, S. **On the Convergence of System-AMG in Reservoir Simulation**. Proceedings from the SPE Reservoir Simulation Symposium held in Montgomery, Texas. In: SPE Journal. Society of Petroleum Engineers (SPE), 2017. SPE 182630.

GROSSMANN, C.; ROOS, H. G.; STYNES, M. **Numerical Treatment of Partial Differential Equations**. Springer Science & Business Media, 2007. ISBN: 978-3-540-71584-9.

GUTKNECHT, M. H. **Variants of BICGSTAB for Matrices with Complex Spectrum**. Journal of Scientific Computing. v. 14. n. 5. pp. 1020-1033. Society for Industrial and Applied Mathematics (SIAM), 1993.

HAMMERSLEY, R. P.; PONTING, D. K. **Solving Linear Equations in Reservoir Simulation Using Multigrid Methods**. Proceedings from the SPE Russian Oil & Gas Technical Conference and Exhibition held in Moscow, Russia. Society of Petroleum Engineers (SPE), 2008. SPE 115017.

IMEX. **IMEX Technical Manual**. Computer Modeling Group (CMG). 2018.

JACKSON, H.; TARONI, M.; PONTING, D. K. **A Two-Level Variant of Additive Schwarz Preconditioning for Use in Reservoir Simulation**. 2014.

PUC-Rio - Certificação Digital Nº 1612782/CA

KELLY, C. T. **Iterative Methods for Linear and Nonlinear Equations**. Society for Industrial and Applied Mathematics (SIAM), 1995.

LACROIX, S.; VASSILEVISKI, Y. V.; WHEELER, M. F. **Decoupling Preconditioners in the Implicit Parallel Accurate Reservoir Simulator (IPARS)**. In: Numerical Linear Algebra Applications. n. 8. pp. 537-549. John Wiley and Sons, 2001.

LAY, D. C. **Linear Algebra**: and its Applications. 3rd ed. Addison Wesley, 2003. ISBN: 0-201-70970-8.

MATLAB. **MATLAB Release Notes**. MathWorks, 2019.

MATTAX, C.; DALTON, R. L. **Reservoir Simulation**. In: SPE Monograph Series. v. 13. Society of Petroleum Engineers (SPE), 1990. ISBN: 1-55563-028-6.

MEIJERINK, J. A.; VAN DER VORST, H. A. **An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix**. Mathematics of Computation. v. 31. n. 37. pp. 148-162. American Mathematical Society (AMS), 1977.

MEYERINK, J. A. **Iterative Methods for the Solution of Linear Equations Based on Block Factorization of the Matrix**. Proceedings from the Reservoir Simulation Symposium held in San Francisco, California. Society of Petroleum Engineers of AIME, 1983. SPE 12262.

PEACEMAN, D. W. **Fundamentals of Numerical Reservoir Simulation**. In: Developments in Petroleum Science. v. 6. Elsevier, 1977. ISBN: 0-444-41578-5.

PONTING, D. K. et al. **An Efficient Fully Implicit Simulator**. Proceedings from the European Offshore Petroleum Conference and Exhibition. In: Society of Petroleum Engineers Journal, 1983. SPE 11817.

PUC-Rio - Certificação Digital Nº 1612782/CA

PRICE, H. S.; COATS, K. H. **Direct Methods in Reservoir Simulation**. Proceedings from the SPE-AIME 3rd Symposium on Numerical Simulation of Reservoir Performance held in Houston, Texas. American Institute of Mining, Metallurgical and Petroleum Engineers, 1974. SPE 4278.

SAAD, Y. **Iterative Methods for Sparse Linear Systems**. 2nd ed. Society for Industrial and Applied Mathematics (SIAM), 2003.

SAAD, Y.; SCHULTZ, M. H. **GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems**. Journal of Scientific Computing. v. 7. n. 7. Society for Industrial and Applied Mathematics (SIAM), 1986.

SCHEICHL, R.; MASSON, R.; WENDEBOURG, J. **Decoupling and Block Preconditionings for Sedimentary Basin Simulations**. Kluwer Academic Publishers, 2003.

SCHENK, O.; GÄRTER, K. **Pardiso User Guide Version 6.0.0**. Intel, 2018.

SCHILDT, H. **C++: The Complete Reference**. 3rd ed. McGraw-Hill, 1998. ISBN: 0-07-213293-0.

SHETH, S. M.; YOUNIS, R. M**. Localized Solvers for General Full-Resolution Implicit Reservoir Simulation**. Proceedings from the SPE Reservoir Simulation Symposium held in Montgomery, Texas. Society of Petroleum Engineers (SPE), 2017. SPE 182691.

SLEIJPEN, G. L. G.; FOKKEMA, D. R. **BICGSTAB(L) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum**. In: Electronic Transactions on Numerical Analysis. v. 1. pp. 11-32. Kent State University, 1993.

SOCIETY OF PETROLEUM ENGINEERS (SPE). **Reservoir Simulation Linear Equation Solver**. Available on

PUC-Rio - Certificação Digital Nº 1612782/CA

<*https://petrowiki.org/Reservoir_simulation_linear_equation_solver*>. Accessed on 07 June, 2018. [?].

SOCIETY OF PETROLEUM ENGINEERS (SPE). **Reservoir Simulation**. Available on <*https://petrowiki.org/Reservoir_simulation*>. Accessed on 07 June, 2018. [?].

STÜBEN, K. **An Introduction to Algebraic Multigrid**. In: TROTTENBERG, U.; OOSTERLEE, C.; SCHÜLLER, A. **Multigrid**. St. Augustin, Germany: German National Center for Information Technology (GMD). Institute for Algorithms and Scientific Computing (SCAI). Elsevier, 2001. Appendix A. pp. 413-532.

STÜBEN, K. **Solving Reservoir Simulation Equations**. Proceeding from the 9th International Forum on Reservoir Simulation held in Abu Dhabi, UAE. St. Augustin, Germany: German National Center for Information Technology (GMD). Institute for Algorithms and Scientific Computing (SCAI), 2007.

STÜBEN, K. et al. **Algebraic Multigrid Methods (AMG) for the Efficient Solution of Fully-Implicit Formulations in Reservoir Simulation**. Proceedings from the SPE Reservoir Simulation Symposium held in Houston, Texas. Society of Petroleum Engineers (SPE), 2007. SPE 105832.

TREFETHEN, L. N.; BAU III, D. **Numerical Linear Algebra**. Society for Industrial and Applied Mathematics (SIAM), 1997. ISBN: 978-0-898713-61-9.

TROTTENBERG, U.; OOSTERLEE, C.; SCHÜLLER, A. **Multigrid**. St. Augustin, Germany: German National Center for Information Technology (GMD). Institute for Algorithms and Scientific Computing (SCAI). Elsevier, 2001. ISBN: 0-12-701070-X.

VAN DER VORST, H. A. **BiCGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Non-Symmetric Linear Systems**. In:

PUC-Rio - Certificação Digital Nº 1612782/CA

Journal on Scientific and Statistical Computing. v. 13. pp. 631-644. Society for Industrial and Applied Mathematics (SIAM), 1992.

VINSOME, P. K. W. **Orthomin, An Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations**. Proceedings from the SPE-AIME 4th Symposium on Numerical Simulation of Reservoir Performance held in Los Angeles, California. American Institute of Mining, Metallurgical and Petroleum Engineers, 1976. SPE 5729.

WALLIS, J. R. **Incomplete Gaussian Elimination as a Preconditioning for Generalized Conjugate Gradient Acceleration**. Proceedings from the Reservoir Simulation Symposium held in San Francisco, California. Society of Petroleum Engineers of AIME, 1983. SPE 12265.

WALLIS, J. R.; KENDALL, R. P.; LITTLE, T. E. **Constrained Pressure Acceleration of Conjugate Residual Methods**. Proceedings from the SPE Reservoir Simulation Symposium held in Dallas, Texas. Society of Petroleum Engineers (SPE), 1985. SPE 13536.

YUVASHANKAR, V..; NEJAD, M. S..; LIU, A. **Understanding Bi-Conjugate Gradient Stabilized Method (Bi-CGSTAB)**. [2016?]. Available on <*http://www.yuvashankar.com/ blog/2016/6/25/understanding-the-bi-conjucate-gradient-stabilized-method*>. Accessed on 25 Dec., 2017.

ZHOU, Y.; TCHELEPI, H. A. **Multi-GPU Parallelization of Nested Factorization for Solving Large Linear Systems**. Proceedings from the SPE Reservoir Simulation Symposium held in The Woodlands, Texas. Society of Petroleum Engineers (SPE), 2013. SPE 163588.

PUC-Rio - Certificação Digital Nº 1612782/CA

A
Complete Multiphase Flow Equations and Jacobian Terms

The complete multiphase flow equations of the black-oil model and the entries to the Jacobian matrix are depicted in this appendix as derived for the fully implicit formulation, with P_o, S_w and S_g as the simulation variables. The definitions of all terms are identical to the ones presented in Chapter 2. Analogously, the equations in this appendix were also derived or extracted from Ertekin et al.'s Basic Applied Reservoir Simulation (2001).

PUC-Rio - Certificação Digital Nº 1612782/CA

A.1
Multiphase Flow Equations

(i) Oil

$$
\begin{aligned}
&T_{ox_{i+\frac{1}{2},j,k}}{}^{n+1}\left(P_{o_{i+1,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i+1,j,k}-P_{o_{i,j,k}}{}^{n+1}\right.\\
&\qquad\left.+\gamma_o{}^{n+1}Z_{i,j,k}\right)\\
&\qquad-T_{ox_{i-\frac{1}{2},j,k}}{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k}-P_{o_{i-1,j,k}}{}^{n+1}\right.\\
&\qquad\left.+\gamma_o{}^{n+1}Z_{i-1,j,k}\right)\\
&\qquad+T_{oy_{i,j+\frac{1}{2},k}}{}^{n+1}\left(P_{o_{i,j+1,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j+1,k}-P_{o_{i,j,k}}{}^{n+1}\right.\\
&\qquad\left.+\gamma_o{}^{n+1}Z_{i,j,k}\right)\\
&\qquad-T_{oy_{i,j-\frac{1}{2},k}}{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k}-P_{o_{i,j-1,k}}{}^{n+1}\right.\\
&\qquad\left.+\gamma_o{}^{n+1}Z_{i,j-1,k}\right)\\
&\qquad+T_{oz_{i,j,k+\frac{1}{2}}}{}^{n+1}\left(P_{o_{i,j,k+1}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k+1}-P_{o_{i,j,k}}{}^{n+1}\right.\\
&\qquad\left.+\gamma_o{}^{n+1}Z_{i,j,k}\right)\\
&\qquad-T_{oz_{i,j,k-\frac{1}{2}}}{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k}-P_{o_{i,j,k-1}}{}^{n+1}\right.\\
&\qquad\left.+\gamma_o{}^{n+1}Z_{i,j,k-1}\right)\\
&\qquad=\frac{\forall_B}{\Delta t}\left[\frac{\phi'}{B_o{}^{n}}+\phi^{n+1}\left(\frac{1}{B_o}\right)'\right]_{i,j,k}\cdot\left(1-S_{w_{i,j,k}}{}^{n}-S_{g_{i,j,k}}{}^{n}\right)\\
&\qquad\cdot\left(P_{o_{i,j,k}}{}^{n+1}-P_{o_{i,j,k}}{}^{n}\right)-\frac{\forall_B}{\Delta t}\left(\frac{\phi}{B_o}\right)_{i,j,k}^{n+1}\\
&\qquad\cdot\left(S_{w_{i,j,k}}{}^{n+1}-S_{w_{i,j,k}}{}^{n}+S_{g_{i,j,k}}{}^{n+1}-S_{g_{i,j,k}}{}^{n}\right)\\
&\qquad-q_{osc_{i,j,k}}{}^{n+1}
\end{aligned}
\tag{A.1}
$$

PUC-Rio - Certificação Digital Nº 1612782/CA

(ii) Water

$$
\begin{aligned}
T_{wx_{i+\frac{1}{2},j,k}}{}^{n+1}\Big(&P_{o_{i+1,j,k}}{}^{n+1} - P_{cow_{i+1,j,k}}{}^{n+1} - \gamma_w{}^{n+1} Z_{i+1,j,k} - P_{o_{i,j,k}}{}^{n+1} \\
&+ P_{cow_{i,j,k}}{}^{n+1} + \gamma_w{}^{n+1} Z_{i,j,k}\Big) \\
&- T_{wx_{i-\frac{1}{2},j,k}}{}^{n+1}\Big(P_{o_{i,j,k}}{}^{n+1} - P_{cow_{i,j,k}} - \gamma_w{}^{n+1} Z_{i,j,k} \\
&- P_{o_{i-1,j,k}}{}^{n+1} + P_{cow_{i-1,j,k}}{}^{n+1} + \gamma_w{}^{n+1} Z_{i-1,j,k}\Big) \\
&+ T_{wy_{i,j+\frac{1}{2},k}}{}^{n+1}\Big(P_{o_{i,j+1,k}}{}^{n+1} - P_{cow_{i,j+1,k}}{}^{n+1} \\
&- \gamma_w{}^{n+1} Z_{i,j+1,k} - P_{o_{i,j,k}}{}^{n+1} + P_{cow_{i,j,k}}{}^{n+1} \\
&+ \gamma_w{}^{n+1} Z_{i,j,k}\Big) \\
&- T_{wy_{i,j-\frac{1}{2},k}}{}^{n+1}\Big(P_{o_{i,j,k}}{}^{n+1} - P_{cow_{i,j,k}}{}^{n+1} - \gamma_w{}^{n+1} Z_{i,j,k} \\
&- P_{o_{i,j-1,k}}{}^{n+1} + P_{cow_{i,j-1,k}}{}^{n+1} + \gamma_w{}^{n+1} Z_{i,j-1,k}\Big) \\
&+ T_{wz_{i,j,k+\frac{1}{2}}}{}^{n+1}\Big(P_{o_{i,j,k+1}}{}^{n+1} - P_{cow_{i,j,k+1}}{}^{n+1} \\
&- \gamma_w{}^{n+1} Z_{i,j,k+1} - P_{o_{i,j,k}}{}^{n+1} + P_{cow_{i,j,k}}{}^{n+1} \\
&+ \gamma_w{}^{n+1} Z_{i,j,k}\Big) \\
&- T_{wz_{i,j,k-\frac{1}{2}}}{}^{n+1}\Big(P_{o_{i,j,k}}{}^{n+1} - P_{cow_{i,j,k}}{}^{n+1} - \gamma_w{}^{n+1} Z_{i,j,k} \\
&- P_{o_{i,j,k-1}}{}^{n+1} + P_{cow_{i,j,k-1}}{}^{n+1} + \gamma_w{}^{n+1} Z_{i,j,k-1}\Big) \\
&= \frac{\forall_B}{\Delta t}\left[\frac{\phi'}{B_w{}^n} + \phi^{n+1}\left(\frac{1}{B_w}\right)'\right]_{i,j,k} \cdot S_{w_{i,j,k}}{}^n \\
&\cdot \left(P_{o_{i,j,k}}{}^{n+1} - P_{o_{i,j,k}}{}^n\right) + \frac{\forall_B}{\Delta t}\left(\frac{\phi}{B_w}\right)_{i,j,k}{}^{n+1} \\
&\cdot \left(S_{w_{i,j,k}}{}^{n+1} - S_{w_{i,j,k}}{}^n\right) - q_{WSC_{i,j,k}}{}^{n+1}
\end{aligned}
\tag{A.2}
$$

PUC-Rio - Certificação Digital Nº 1612782/CA

(iii) Gas

$$
\begin{aligned}
&T_{gx_{i+\frac{1}{2},j,k}}{}^{n+1}\left(P_{o_{i+1,j,k}}{}^{n+1}+P_{cgo_{i+1,j,k}}{}^{n+1}-\gamma_g{}^{n+1}Z_{i+1,j,k}-P_{o_{i,j,k}}{}^{n+1}-P_{cgo_{i,j,k}}{}^{n+1}+\gamma_g{}^{n+1}Z_{i,j,k}\right)\\
&\quad-T_{gx_{i-\frac{1}{2},j,k}}{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}+P_{cgo_{i,j,k}}-\gamma_g{}^{n+1}Z_{i,j,k}-P_{o_{i-1,j,k}}{}^{n+1}-P_{cgo_{i-1,j,k}}{}^{n+1}\right.\\
&\quad\left.+\gamma_g{}^{n+1}Z_{i-1,j,k}\right)\\
&\quad+T_{ox_{i+\frac{1}{2},j,k}}{}^{n+1}R_s{}^{n+1}\left(P_{o_{i+1,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i+1,j,k}-P_{o_{i,j,k}}{}^{n+1}\right.\\
&\quad\left.+\gamma_o{}^{n+1}Z_{i,j,k}\right)\\
&\quad-T_{ox_{i-\frac{1}{2},j,k}}{}^{n+1}R_s{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k}-P_{o_{i-1,j,k}}{}^{n+1}+\gamma_o{}^{n+1}Z_{i-1,j,k}\right)\\
&\quad+T_{gy_{i,j+\frac{1}{2},k}}{}^{n+1}\left(P_{o_{i,j+1,k}}{}^{n+1}+P_{cgo_{i,j+1,k}}{}^{n+1}-\gamma_g{}^{n+1}Z_{i,j+1,k}-P_{o_{i,j,k}}{}^{n+1}\right.\\
&\quad\left.-P_{cgo_{i,j,k}}{}^{n+1}+\gamma_g{}^{n+1}Z_{i,j,k}\right)\\
&\quad-T_{gy_{i,j-\frac{1}{2},k}}{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}+P_{cgo_{i,j,k}}{}^{n+1}-\gamma_g{}^{n+1}Z_{i,j,k}-P_{o_{i,j-1,k}}{}^{n+1}\right.\\
&\quad\left.-P_{cgo_{i,j-1,k}}{}^{n+1}+\gamma_g{}^{n+1}Z_{i,j-1,k}\right)\\
&\quad+T_{oy_{i,j+\frac{1}{2},k}}{}^{n+1}R_s{}^{n+1}\left(P_{o_{i,j+1,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j+1,k}-P_{o_{i,j,k}}{}^{n+1}+\gamma_o{}^{n+1}Z_{i,j,k}\right)\\
&\quad-T_{oy_{i,j-\frac{1}{2},k}}{}^{n+1}R_s{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k}-P_{o_{i,j-1,k}}{}^{n+1}+\gamma_o{}^{n+1}Z_{i,j-1,k}\right)\\
&\quad+T_{gz_{i,j,k+\frac{1}{2}}}{}^{n+1}\left(P_{o_{i,j,k+1}}{}^{n+1}+P_{cgo_{i,j,k+1}}{}^{n+1}-\gamma_g{}^{n+1}Z_{i,j,k+1}-P_{o_{i,j,k}}{}^{n+1}\right.\\
&\quad\left.-P_{cgo_{i,j,k}}{}^{n+1}+\gamma_g{}^{n+1}Z_{i,j,k}\right)\\
&\quad-T_{gz_{i,j,k-\frac{1}{2}}}{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}+P_{cgo_{i,j,k}}{}^{n+1}-\gamma_g{}^{n+1}Z_{i,j,k}-P_{o_{i,j,k-1}}{}^{n+1}\right.\\
&\quad\left.-P_{cgo_{i,j,k-1}}{}^{n+1}+\gamma_g{}^{n+1}Z_{i,j,k-1}\right)\\
&\quad+T_{oz_{i,j,k+\frac{1}{2}}}{}^{n+1}R_s{}^{n+1}\left(P_{o_{i,j,k+1}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k+1}-P_{o_{i,j,k}}{}^{n+1}+\gamma_o{}^{n+1}Z_{i,j,k}\right)\\
&\quad-T_{oz_{i,j,k-\frac{1}{2}}}{}^{n+1}R_s{}^{n+1}\left(P_{o_{i,j,k}}{}^{n+1}-\gamma_o{}^{n+1}Z_{i,j,k}-P_{o_{i,j,k-1}}{}^{n+1}+\gamma_o{}^{n+1}Z_{i,j,k-1}\right)\\
&\quad=\frac{\forall_B}{\Delta t}\left(\left\{\left[\frac{\phi'}{B_o{}^n}+\phi^{n+1}\left(\frac{1}{B_o}\right)'\right]\cdot R_s{}^n+\left(\frac{\phi}{B_o}\right)^{n+1}\cdot R_s{}'\right\}_{i,j,k}\right.\\
&\quad\left.\times\left(1-S_{w_{i,j,k}}{}^n-S_{g_{i,j,k}}{}^n\right)+\left[\frac{\phi'}{B_g{}^n}+\phi^{n+1}\left(\frac{1}{B_g}\right)'\right]_{i,j,k}\cdot S_{g_{i,j,k}}{}^n\right)\\
&\quad\cdot\left(P_{o_{i,j,k}}{}^{n+1}-P_{o_{i,j,k}}{}^n\right)-\frac{\forall_B}{\Delta t}\left[\left(\frac{\phi}{B_o}\right)^{n+1}\cdot R_s{}^{n+1}\right]_{i,j,k}\cdot\left(S_{w_{i,j,k}}{}^{n+1}-S_{w_{i,j,k}}{}^n\right)\\
&\quad+\frac{\forall_B}{\Delta t}\left[\left(\frac{\phi}{B_g}\right)^{n+1}-\left(\frac{\phi}{B_o}\right)^{n+1}\cdot R_s{}^{n+1}\right]_{i,j,k}\cdot\left(S_{g_{i,j,k}}{}^{n+1}-S_{g_{i,j,k}}{}^n\right)-q_{GSC_{i,j,k}}{}^{n+1}
\end{aligned}
\tag{A.3}
$$

PUC-Rio - Certificação Digital Nº 1612782/CA

A.2
Jacobian Matrix Entries

(i) Oil Derivatives with respect to Neighboring Cells

$$\left(\frac{\partial R_{o_n}}{\partial P_{o_m}}\right)^{(\nu)} = \left[T_{o_{n,m}}{}^{(\nu)} + \left[\Delta_m P_o{}^{(\nu)} - \bar{\gamma}^n{}_{o_{n,m}} \Delta_m Z\right] \left(\frac{\partial T_{o_{n,m}}}{\partial P_{o_m}}\right)^{(\nu)}\right] \quad \text{(A.4)}$$

$$\left(\frac{\partial R_{o_n}}{\partial S_{w_m}}\right)^{(\nu)} = \left[\left[\Delta_m P_o{}^{(\nu)} - \bar{\gamma}^n{}_{o_{n,m}} \Delta_m Z\right] \left(\frac{\partial T_{o_{n,m}}}{\partial S_{w_m}}\right)^{(\nu)}\right] \quad \text{(A.5)}$$

$$\left(\frac{\partial R_{o_n}}{\partial S_{g_m}}\right)^{(\nu)} = \left[\left[\Delta_m P_o{}^{(\nu)} - \bar{\gamma}^n{}_{o_{n,m}} \Delta_m Z\right] \left(\frac{\partial T_{o_{n,m}}}{\partial S_{g_m}}\right)^{(\nu)}\right] \quad \text{(A.6)}$$

(ii) Water Derivatives with respect to Neighboring Cells

$$\left(\frac{\partial R_{w_n}}{\partial P_{o_m}}\right)^{(\nu)} = \left[T_{w_{n,m}}{}^{(\nu)} + \left[\Delta_m P_o{}^{(\nu)} - \Delta_m P_{cow}{}^{(\nu)} - \bar{\gamma}^n{}_{w_{n,m}} \Delta_m Z\right] \left(\frac{\partial T_{w_{n,m}}}{\partial P_{o_m}}\right)^{(\nu)}\right] \quad \text{(A.7)}$$

$$\left(\frac{\partial R_{w_n}}{\partial S_{w_m}}\right)^{(\nu)} = \left[\left[\Delta_m P_o{}^{(\nu)} - \Delta_m P_{cow}{}^{(\nu)} - \bar{\gamma}^n{}_{w_{n,m}} \Delta_m Z\right] \left(\frac{\partial T_{w_{n,m}}}{\partial S_{w_m}}\right)^{(\nu)} - T_{w_{n,m}}{}^{(\nu)} P'{}_{cow_m}{}^{(\nu)}\right] \quad \text{(A.8)}$$

$$\left(\frac{\partial R_{w_n}}{\partial S_{g_m}}\right)^{(\nu)} = 0 \quad \text{(A.9)}$$

(iii) Gas Derivatives with respect to Neighboring Cells

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\left(\frac{\partial R_{g_n}}{\partial P_{o_m}}\right)^{(\nu)} = \Bigg[{T_{g_{n,m}}}^{(\nu)} + \Big[{\Delta_m P_o}^{(\nu)} + {\Delta_m P_{cgo}}^{(\nu)} - {\bar{\gamma}^n}_{g_{n,m}} \Delta_m Z \Big] \left(\frac{\partial T_{g_{n,m}}}{\partial P_{o_m}}\right)^{(\nu)} + {(T_o R_s)_{n,m}}^{(\nu)} + \Big[{\Delta_m P_o}^{(\nu)} - {\bar{\gamma}^n}_{o_{n,m}} \Delta_m Z \Big] \left(\frac{\partial (T_o R_s)_{n,m}}{\partial P_{o_m}}\right)^{(\nu)} \Bigg] \tag{A.10}$$

$$\left(\frac{\partial R_{g_n}}{\partial S_{w_m}}\right)^{(\nu)} = \left[\Big[{\Delta_m P_o}^{(\nu)} - {\bar{\gamma}^n}_{o_{n,m}} \Delta_m Z \Big] \left(\frac{\partial (T_o R_s)_{n,m}}{\partial S_{w_m}}\right)^{(\nu)} \right] \tag{A.11}$$

$$\left(\frac{\partial R_{g_n}}{\partial S_{g_m}}\right)^{(\nu)} = \Bigg[\Big[{\Delta_m P_o}^{(\nu)} + {\Delta_m P_{cgo}}^{(\nu)} - {\bar{\gamma}^n}_{g_{n,m}} \Delta_m Z \Big] \left(\frac{\partial T_{g_{n,m}}}{\partial S_{g_m}}\right)^{(\nu)} + {T_{g_{n,m}}}^{(\nu)} {P'_{cgo}}_m^{(\nu)} + \Big[{\Delta_m P_o}^{(\nu)} - {\bar{\gamma}^n}_{o_{n,m}} \Delta_m Z \Big] \left(\frac{\partial (T_o R_s)_{n,m}}{\partial S_{g_m}}\right)^{(\nu)} \Bigg] \tag{A.12}$$

(iv) Oil Derivatives with respect to Current Cell

$$\left(\frac{\partial R_{o_n}}{\partial P_{o_n}}\right)^{(\nu)} = \Bigg[\sum_{m \in \psi_n} \Bigg(-{T_{o_{n,m}}}^{(\nu)} + \Big[{\Delta_m P_o}^{(\nu)} - {\bar{\gamma}^n}_{o_{n,m}} \Delta_m Z \Big] \left(\frac{\partial T_{o_{n,m}}}{\partial P_{o_n}}\right)^{(\nu)} \Bigg) - {C_{op_n}}^{(\nu)} + \left(\frac{\partial q_{osc_n}}{\partial P_{o_n}}\right)^{(\nu)} \Bigg] \tag{A.13}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\left(\frac{\partial R_{o_n}}{\partial S_{w_n}}\right)^{(\nu)} = \left[\sum_{m\in\psi_n}\left(\left[\Delta_m P_o{}^{(\nu)} - \bar{\gamma}^n{}_{o_{n,m}}\Delta_m Z\right]\left(\frac{\partial T_{o_{n,m}}}{\partial S_{w_n}}\right)^{(\nu)}\right) - C_{ow_n}{}^{(\nu)} + \left(\frac{\partial q_{OSC_n}}{\partial S_{w_n}}\right)^{(\nu)}\right] \tag{A.14}$$

$$\left(\frac{\partial R_{o_n}}{\partial S_{g_n}}\right)^{(\nu)} = \left[\sum_{m\in\psi_n}\left(\left[\Delta_m P_o{}^{(\nu)} - \bar{\gamma}^n{}_{o_{n,m}}\Delta_m Z\right]\left(\frac{\partial T_{o_{n,m}}}{\partial S_{g_n}}\right)^{(\nu)}\right) - C_{og_n}{}^{(\nu)} + \left(\frac{\partial q_{OSC_n}}{\partial S_{g_n}}\right)^{(\nu)}\right] \tag{A.15}$$

(v) Water Derivatives with respect to Current Cell

$$\left(\frac{\partial R_{w_n}}{\partial P_{o_n}}\right)^{(\nu)} = \left[\sum_{m\in\psi_n}\left(-T_{w_{n,m}}{}^{(\nu)} + \left[\Delta_m P_o{}^{(\nu)} - \Delta_m P_{cow}{}^{(\nu)} - \bar{\gamma}^n{}_{w_{n,m}}\Delta_m Z\right]\left(\frac{\partial T_{w_{n,m}}}{\partial P_{o_n}}\right)^{(\nu)}\right) - C_{wp_n}{}^{(\nu)} + \left(\frac{\partial q_{WSC_n}}{\partial P_{o_n}}\right)^{(\nu)}\right] \tag{A.16}$$

$$\left(\frac{\partial R_{w_n}}{\partial S_{w_n}}\right)^{(\nu)} = \left[\sum_{m\in\psi_n}\left(\left[\Delta_m P_o{}^{(\nu)} - \Delta_m P_{cow}{}^{(\nu)} - \bar{\gamma}^n{}_{w_{n,m}}\Delta_m Z\right]\left(\frac{\partial T_{w_{n,m}}}{\partial S_{w_n}}\right)^{(\nu)} + T_{w_{n,m}}{}^{(\nu)} P'_{cow_n}{}^{(\nu)}\right) - C_{ww_n}{}^{(\nu)} + \left(\frac{\partial q_{WSC_n}}{\partial S_{w_n}}\right)^{(\nu)}\right] \tag{A.17}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\left(\frac{\partial R_{w_n}}{\partial S_{g_n}}\right)^{(\nu)} = \left[-C_{wg_n}{}^{(\nu)} + \left(\frac{\partial q_{WSC_n}}{\partial S_{g_n}}\right)^{(\nu)}\right] \tag{A.18}$$

(vi) Gas Derivatives with respect to Current Cell

$$\begin{aligned}\left(\frac{\partial R_{g_n}}{\partial P_{o_n}}\right)^{(\nu)} = \Bigg[\sum_{m\in\psi_n}\Bigg(&-T_{g_{n,m}}{}^{(\nu)} \\ &+ \left[\Delta_m P_o{}^{(\nu)} + \Delta_m P_{cgo}{}^{(\nu)}\right. \\ &\left. - \bar{\gamma}^n{}_{g_{n,m}} \Delta_m Z\right]\left(\frac{\partial T_{g_{n,m}}}{\partial P_{o_n}}\right)^{(\nu)} - (T_o R_s)_{n,m}{}^{(\nu)} \\ &+ \left[\Delta_m P_o{}^{(\nu)} - \bar{\gamma}^n{}_{o_{n,m}} \Delta_m Z\right]\left(\frac{\partial (T_o R_s)_{n,m}}{\partial P_{o_n}}\right)^{(\nu)}\Bigg) \\ &- C_{gp_n}{}^{(\nu)} + \left(\frac{\partial q_{GSC_n}}{\partial P_{o_n}}\right)^{(\nu)}\Bigg]\end{aligned} \tag{A.19}$$

$$\begin{aligned}\left(\frac{\partial R_{g_n}}{\partial S_{w_n}}\right)^{(\nu)} = \Bigg[\sum_{m\in\psi_n}\Bigg(&\left[\Delta_m P_o{}^{(\nu)}\right. \\ &\left. - \bar{\gamma}^n{}_{o_{n,m}} \Delta_m Z\right]\left(\frac{\partial (T_o R_s)_{n,m}}{\partial S_{w_n}}\right)^{(\nu)}\Bigg) - C_{gw_n}{}^{(\nu)} \\ &+ \left(\frac{\partial q_{GSC_n}}{\partial S_{w_n}}\right)^{(\nu)}\Bigg]\end{aligned} \tag{A.20}$$

PUC-Rio - Certificação Digital Nº 1612782/CA

$$\left(\frac{\partial R_{g_n}}{\partial S_{g_n}}\right)^{(\nu)} = \left[\sum_{m\in\psi_n}\left(\left[\Delta_m {P_o}^{(\nu)} + \Delta_m {P_{cgo}}^{(\nu)} - \bar{\gamma}^n{}_{g_{n,m}}\Delta_m Z\right]\left(\frac{\partial T_{g_{n,m}}}{\partial S_{g_n}}\right)^{(\nu)} - {T_{g_{n,m}}}^{(\nu)}{P'}_{cgo_n}{}^{(\nu)} + \left[\Delta_m {P_o}^{(\nu)} - \bar{\gamma}^n{}_{o_{n,m}}\Delta_m Z\right]\left(\frac{\partial (T_o R_s)_{n,m}}{\partial S_{g_n}}\right)^{(\nu)}\right) - {C_{gg_n}}^{(\nu)} + \left(\frac{\partial q_{GSC_n}}{\partial S_{g_n}}\right)^{(\nu)}\right] \quad \text{(A.21)}$$

PUC-Rio - Certificação Digital Nº 1612782/CA