Derivative Computation using an Adjoint Based Goal Oriented Iterative Multiscale Lagrangian Framework

Application to Reservoir Simulation

W. de Zeeuw

Technische Universiteit Delft

TU Delft

Derivative Computation using an Adjoint-Based Goal-Oriented Iterative Multiscale Lagrangian Framework

Application to Reservoir Simulation

by

Wessel de Zeeuw

to obtain the degree of Master of Science in
Applied Mathematics at the Delft University of Technology,
to be defended publicly on Friday August 31, 2018 at 13:30 AM.

Student number:	4216172	
Project duration:	December 1, 2017 – August 31, 2018	
Supervisor:	Prof. dr. ir. A.W. Heemink	
Thesis committee:	Prof. dr. ir. A.W. Heemink,	TU Delft
	Prof. dr. ir. M. Verlaan,	TU Delft
	Prof. dr. ir. J.D. Jansen,	TU Delft
	MSc. R.J. Moraes,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

Even though negative effects on the use of crude oil have surfaced over the past years, our energy matrix still largely relies on this energy source. The production of oil, therefore, plays an important role in our society. Unfortunately, the process of oil production is highly uncertain. There are uncertainties associated on the production strategy, e.g. where and how many wells should be drilled and how these wells should operate because of the uncertainties associated with the limited knowledge about the subsurface. In this thesis we are dealing with the uncertainty of the rock permeability distribution. Typically, rock permeabilities in the rock vary, but from the outside this can't be perceived. If these rock permeabilities are estimated inaccurately, they will result in inaccurate pressure solutions. Then, this can lead to faulty decisions regarding the oil exploitation. To resolve this issue, a data assimilation technique may be applied to correct these model parameters based on mismatch of simulated data and observations. For this optimization technique, often gradient information is required. Since in reservoir simulation the number of parameters generally is extremely high, computation of this information is computationally expensive. Therefore, a multiscale framework is employed to improve the computational efficiency of the forward simulation. Multiscale methods are able to solve the model equations at a computationally efficient coarse scale and can easily interpolate this solution to the fine scale resolution. Next, we use a Lagrangian set-up together with a multiscale framework to re-derive an efficient formulation for the derivative computation. However, as the multiscale method is prone to errors, this derivative computation formulation is recast in an iterative fashion, using a residual based iterative multiscale method to provide control of these errors. In this thesis we show that this method generates accurate gradients. In contract to the high accuracy of the method, this method comprises a computationally heavy smoothing step. This issue can be resolved by making smart use of the Lagrange multipliers, to re-derive an efficient iterative multiscale solution strategy. The multipliers are used to identify important domains of the region for which smoothing is required and for which regions we may neglect the smoothing. We show that the newly proposed iterative multiscale goal oriented method is computationally more efficient and we show that method is promising for efficient derivative computation, but that more work is required to fully demonstrate the benefit of this method.

Preface

Before you, you find my graduation thesis. This thesis is the final requirement to fulfil all graduation requirements for the degree Master of Science in Applied Mathematics, with a specialization in Computational Sciences and Engineering. The research for this project was executed at Delft University of Technology at the Delft Institute of Applied Mathematics (DIAM), in collaboration with the Petroleum Engineering section of the Geosciences and Engineering Department of Civil Engineering and Geosciences faculty. In this preface I would like to sincerely thank several people that have made this project successful.

To start of, I would like to start by thanking my supervisors. Rafael, without you this project would probably never have finished. Thank you for your enormous investment, and the possibility for me to ask all the questions I want. Your open, energetic and yet critical view on the project inspired me to push myself further each time. Arnold Heemink, thank you for taking time to explain all the different methods, for giving me useful feedback and support throughout the entire process.

Secondly, I would like to thank my friends and family. Dear friends, you have made the previous period a very enjoyable one. Thank you for listening to all my struggles, even though you were going through your own rough periods too. Because of your energy, interest and good conversations, I was able to relieve my stress levels and work harder the day after. My dear family, thank you for having all the patience in me. My college years was a period of ups and downs, but you have always been there for me, no matter what! Without you all, these previous years wouldn't have been the same.

Finally, I would like to thank prof.ir. J.D. Jansen and prof.ir. M. Verlaan for their time, energy and willingness to take place in my graduation committee.

I have had a blast working on this project, and I wish you as much pleasure in reading my Master thesis as I had working on this project.

Wessel de Zeeuw
Delft, August 2018

Contents

List of Figures

List of Tables

1 Introduction

1.1. Reservoir Simulation

1.1.1. Industry

Crude oil is one of the energy sources with extremely high consumer rates. It is estimated that only in the United States alone around 71.9 billion barrels of oil are consumed per year. This accounts for 37% of the total energy consumption. Oil can be used for different ends. For example, oil is used to propel vehicles, heat buildings and produce electricity [35]. Even though throughout the years negative effects on the use of crude oil surfaced, the industry still remains a billion dollar industry. Large companies such as Shell form a revenue of USD 450 billion in 2016 [27].

The process of oil production is a highly uncertain process. Oil is exploited by drilling wells into rocks of the earth that may, or may not, contain crude oil. Then, under high pressure water is pumped through the pores of this media. Due to pressure differences oil flows from one side to another. Production wells bring the oil to surface level [31]. This process is visualized in Figure [1.1]. These rocks often contain natural gas and water next to oil, and therefore also can be produced as side products. So choosing the location where you produce for crude oil can be critical.

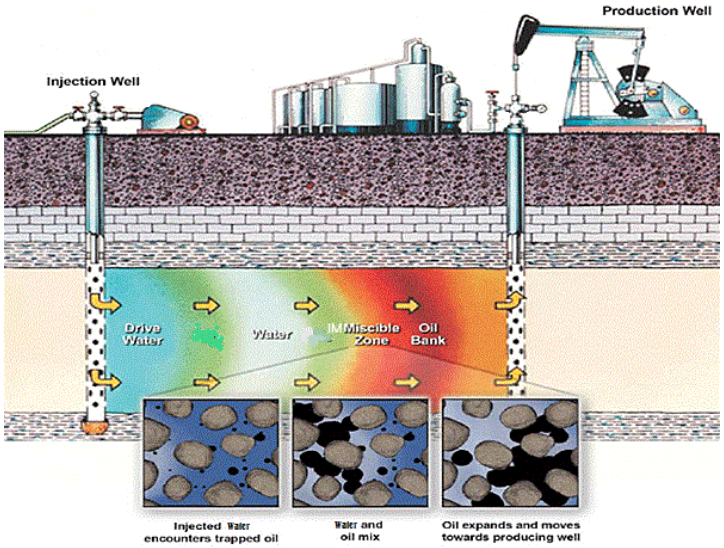


Figure 1.1: Illustration of exploiting of oil

In secondary recovery [34], one uses two or three different types of wells. There are injection, production and observation wells. At the injection wells, water is injected into the natural formation under high pressure. This injection causes the oil and water to be pushed through the pores of the rock. At the

production wells, the wells use pressure to bring the liquids or gas to ground level, all with the goal of producing as much oil and as little of water as possible. Observation wells are wells that give information of different quantities, such as pressure. It is clear that in this process there are many different variables. Firstly, where should we position the wells? How many wells should we use? At which pressure rates should the wells operate? To answer these questions optimization methods are used. In these models, they calculate the optimal parameters to maximize an economic variable such as a net present value. A net present value is a measure of profit by substracting the cash outflow (costs for exploiting, or penalties for producing water or gas) and adding the cash inflow (income generated from oil productions).

1.1.2. Data assimilation

It is known that for rocks containing crude oil, the permeability across these rocks is highly heterogeneous. Permeability is a measure of how the pores are connected throughout the rock, and hence gives a direct relation to the flow of oil. This heterogeneous property has as effect that it makes is hard to know the complete permeability distribution across the entire rock. For reservoir simulation, this poses an issue that the parameters for the model, i.e. the permeabilities, are unknown and need to be estimated to the best of our knowledge. However, estimating these parameters may lead to very inaccurate models and predictions. To resolve this issue, we apply data assimilation techniques to improve our estimations of the permeability distribution. Data assimilation [17], is a mathematical technique that aims to minimize the mismatch or misfit between the measured data and model responses. Typical measurements are pressures at the observation wells. The model responses are computed by running the forward model.

In the view of uncertainty in permeability, the goal of this technique is to adjust the permeabilities parameters in such manner, that the generated model responses are better aligned with the observations. In other words, we minimize the misfit between these values. This method creates predictions that are more accurate and reliable.

1.2. Derivative Computation

Now that we know that we want to minimize the misfit between model responses and measured data, we now ask ourselves the question how we determine for which parameters a correction is required and how big this correction should be. The model solutions can be used to evaluate an function of interest, called the objective function. An example is the economic Net Present Value function. To choose an optimal value of the parameters we must first know the effect to the objective function when a change in the parameters is made. If the effect is large, we make a large adjustment, if the effect is small we make a small adjustment. This rate of change in the model parameters is given by the derivative of the objective function with respect to the model parameters.

For reservoir data assimilation techniques often gradient information is required to find the optimal value. As we will later discuss, reservoir models are often described by a high number of parameters, therefore making the computational demand for computing the gradient information high. To resolve the issue of high computational costs, we will introduce a multiscale framework [43, 44]. The multiscale method is a method that allows us to calculate the solution on a computational domain of much smaller size, therefore decreasing the computational cost. Then, the solution is interpolated back to the original domain. In conclusion, this framework enables us to efficiently and very accurately represent the solution in the original computation domain, however with less computational effort.

Now, in order to calculate the gradient information in a multiscale framework, we recast the multiscale derivative computation introduced in [25] with aid of the Lagrange Multiplier Method [36]. Apart from giving us a mathematical formulation for the gradient information, this method has another advantage. The multipliers calculated during this method prove to be beneficial for later utilization.

The multiscale method is prone to generate errors in the solution and therefore may also lead to inaccurate gradient estimation. To resolve this issue, previous work by [30] has shown that an iterative multiscale method may provide the solution to resolve the errors generated in the gradient computation.

The iterative multiscale method [12, 13] is an iterative algorithm that reduces the error of the solution generated by the multiscale method. It gives more accurate solutions and in this thesis we will show that this means that it also leads to accurate gradients.

As mentioned before, the Lagrange Multipliers can be extremely beneficial and in our setting, and may be used for different purposes. Because of their interpretation, Lagrange Multipliers can be used for different strategies to reduce the computational costs. They are used for grid-refinement techniques [7], for reducing computational effort and for adaptive coarsening of the multiscale model [39]. In this thesis we focus on reducing the computational effort of the iterative multiscale method. Because even though the solution from this method is extremely accurate, the method uses a computationally heavy smoothing step. By introducing the adjoint [3, 19], we define a method to assess the sensitivity of the computational domain with respect to the optimization goal. For the domains that have a low effect, we accept the solution to be accurate enough, and leave out the smoothing stage. This new method is called the goal oriented iterative multiscale method.

1.3. Research and outline

This research was conducted at Delft University of Technology as collaboration between the departments of Applied Mathematics and Applied Geo Sciences. In order to successfully complete the master thesis project, a set of research questions were posed at the start of the thesis:

1. How can we recast the multiscale gradient computation by using a Lagrange Multiplier framework and how can we benefit from the multipliers in a goal oriented iterative multiscale method?

2. Does the iterative multiscale model provide accurate gradients when compared to the full fine scale gradient?

3. Can we improve the computational efficiency of the iterative multiscale method by using the Lagrange Multipliers to select the domain that is most sensitive to errors with respect to a user defined goal?

To answer these questions, we have structured this thesis in as follows. After this introduction, Chapter two: *Mathematical Formulation of Flow in Porous Media* will explain the mathematical background of reservoir simulation. We explain important concepts that are used throughout this thesis. Assumptions are made and validated. Also, we show how to solve a system of equation using the multiscale model.

Next, Chapter three: *Continuous Optimization*, briefly describes the concepts and theory behind optimizational techniques. Here, we explain the Lagrange Multiplier method and we introduce the reader to gradient-based optimization algorithms.

Then, Chapter four: *Multiscale Methods*, deals with all the theory around multiscale methods. An extensive derivation of the method is given. Next, we describe the algebraic multiscale method that is used to solve the forward model equations in a multiscale setting.

Consecutively, we will develop iterative multiscale methods in Chapter five: *Iterative Multiscale Models*. Here the framework to deal with the errors as generated by the multiscale methods is described.

In Chapter six: *Derivation of the Multiscale Lagrange Multiplier Gradient Computation Framework*, we will bring the theory of the Lagrange multiplier method and the multiscale method into practice and we will derive an analytical expression for the gradient information.

Now that we have found an analytical expression for the gradient information, Chapter seven: *Derivation of the Iterative Multiscale Gradient Computation Framework*, will provide algorithms to compute the analytical terms. Two different methods are provided in both an iterative- and multiscale framework.

Chapter eight: *Numerical i-MS gradient computation experiments*, focusses on numerical experiments proving the robustness and accuracy of the iterative Multiscale gradient information, compared to the

fine scale gradient information. Multiple test cases are considered, with increasing complexity.

Then, Chapter nine: *Goal-oriented Adjoint based Optimization*, introduces the Goal-Oriented method. This method aims to identify the spatial regions of the reservoir where errors in the solution lead to big differences in a specified goal. The other positions are regarded as less important and can therefore be disregarded in the smoothing stage of the iterative multiscale model, hence decreasing the computational costs even further.

Chapter ten: *Numerical Goal-oriented Adjoint based Optimization experiments*, focusses on numerical experiments to show the computational efficiency with respect to the previous model. Also we will investigate the accuracy of the gradient computation with this new model.

Finally, we will finish this thesis report with a conclusion about the results generated in the paper. After this, we give some recommendations for future work.

2

Mathematical Formulation of Flow in Porous Media

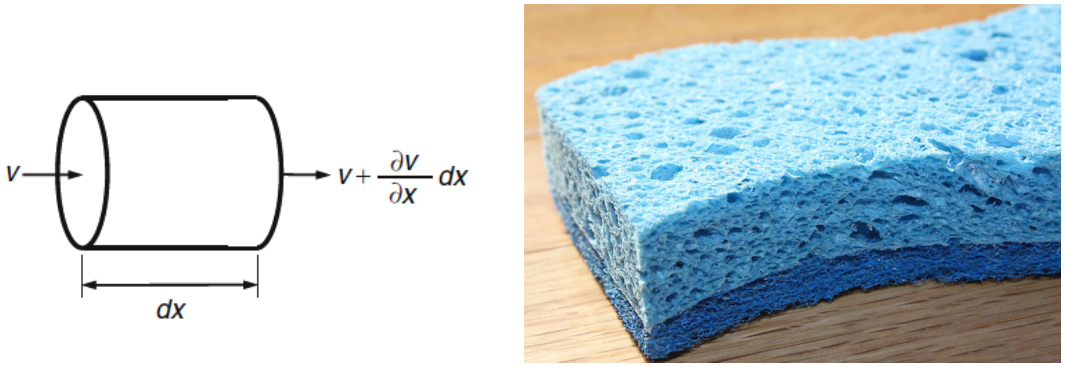
In this chapter we will discuss the mathematical formulation behind the reservoir simulation models. Reservoir simulation is about simulation flow of one, or more, fluids through porous media. We present the mathematical equations that form the basis of the model. During this entire thesis, we consider single phase flow. Throughout this chapter we follow the works of [16, 18, 45].

2.1. Single phase flow

We start the derivation for the governing equations by considering a one-dimensional, horizontal flow of a compressible single-phase fluid through a constant cross-sectional area compressible porous media. To derive the flow equations we consider the mass balance per unit of time for a control volume with length dx. This idea is illustrated in Figure [2.1a]. Mass conservation leads to the equation

$$A\rho v - A\left(\rho + \frac{\partial \rho}{\partial x}dx\right)\left(v + \frac{\partial v}{\partial x}dx\right) - A\frac{\partial \rho\phi}{\partial t}dx + \rho q = 0, \tag{2.1}$$

where A represents the cross-sectional area, $\rho(\mathbf{x}, t)$ the fluid density, $v(\mathbf{x}, t)$ the Darcy velocity and source term $q(\mathbf{x}, t)$. We use spatial coordinate $\mathbf{x}$ and use symbol t to indicate time dependency. For the one-dimensional case we have $\mathbf{x} = x$. Finally we have porosity $\phi(\mathbf{x}, t)$. Porosity as used in this thesis, is defined as the ratio between pore volume and the total volume of a material. A good example of a material with high porosity is a scourer, as depicted in Figure [2.1b]



(a) Control Volume to derive Mass Balance.

(b) Example of Pores in a scourer [1].

Figure 2.1: Left: Control Volume to derive Mass Balance. Right: Visualization of Pores in a scourer.

Now we consider all first order terms, and neglect all higher-order derivative terms of equation (2.1).

We expand and simplify to find

$$\frac{(\partial \rho v)}{\partial x} + \frac{\partial \rho \phi}{\partial t} - \rho q''' = 0, \tag{2.2}$$

where $q''' = \frac{q}{Adx}$ is the new source term. If we repeat this procedure for the additional spatial positions y and z so that $\mathbf{x} = (x, y, z)$, we can generalize equation (2.2) into a three-dimensional equation

$$\nabla \cdot (\rho \mathbf{v}) + \frac{\partial \rho \phi}{\partial t} - \rho q''' = 0, \tag{2.3}$$

where $\mathbf{v} = (v_x, v_y, v_z)^\top$ represents the Darcy velocity with respect to the spatial direction and where $\nabla = (\partial_x, \partial_y, \partial_z)^\top$ is the gradient vector. The divergence operator is defined as the operation $\nabla \cdot (\bullet) = \sum_{i=1}^{n} \frac{\partial (\bullet)_i}{\partial x_i}$, where n represents the number of dimensions. The Darcy velocity is a fictional velocity that would occur if the entire cross-section would be open to flow, i.e. if the entire medium would be porous, hence $\phi = 1$ [37]. The Darcy Velocity is given by [37]

$$\mathbf{v} = -\frac{1}{\mu} \vec{\mathbf{K}} (\nabla p - \rho g \nabla d), \tag{2.4}$$

where $\vec{\mathbf{K}}(\mathbf{x})$ is the permeability tensor and may be written as a diagonal matrix, $\vec{\mathbf{K}} = \text{diag}(K_x, K_y, K_z)$. The permeability tensor is an indication of the ease with which fluids can pass through the porous media. Logically, the permeability is related to the porosity, in the sense that permeability gives an indication of how well the pores in the material are connected. However, we highlight that even though if the porosity is high, but the pores are very ill connected, the permeability is low [14]. A visualization of this explanation is found in Figure [2.2].

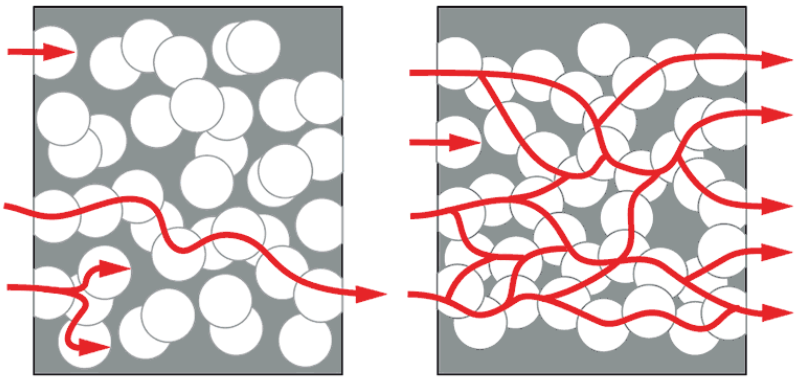


Figure 2.2: Visualization of the relationship between porosity and permeability. Left: good porosity, but poor permeability. Right: good porosity and good permeability [14].

Furthermore, in equation (2.4) we have g as the acceleration of gravity and $d(\mathbf{x})$ is the depth of the reservoir. Combining equations (2.3) and (2.4), this results in

$$-\nabla \cdot \left[\frac{\rho}{\mu} \vec{\mathbf{K}} (\nabla p - \rho g \nabla d) \right] + \frac{\partial (\rho \phi)}{\partial t} - \rho q''' = 0 \tag{2.5}$$

We may decompose this equation into three parts. The first is the flux term, which indicates the flow rate per unit of area. Next, we have the accumulation term, which is an indication of how much is accumulated in the system. Finally we have a source term, indicating the systems in- or outflow. The different terms are indicated in the following equation

$$\underbrace{-\nabla \cdot \left[\frac{\rho}{\mu} \vec{\mathbf{K}} (\nabla p - \rho g \nabla d) \right]}_{\text{Flux-term}} + \underbrace{\frac{\partial (\rho \phi)}{\partial t}}_{\text{Accumulation term}} - \underbrace{\rho q'''}_{\text{Source term}} = 0 \tag{2.6}$$

Equation (2.6), is also called the mass balance equation, as it describes velocity change and hence, the displacement of the fluids through the porous media. Here, multiple variables may be dependent on the pressure. In practice one generally assumes that the variables $\vec{\mathbf{K}}, \mu$ have a small pressure dependency. Hence, from now on we will assume independence of these variables on pressure. The variables ρ and ϕ are in relationship with the pressure. The relationship between p and ρ is given by the equations of state. This equation describes the state of the phase in presence of physical parameters such as pressure and temperature. We have that our fluid phase has a compressibility $c_l(p)$ and is defined as

$$c_l(p) = \frac{1}{\rho}\frac{\partial \rho}{\partial p}\bigg|_{T_0}, \tag{2.7}$$

where T_0 is a constant reference temperature. In the same way we have a relationship between ϕ and p given by

$$c_r(p) = \frac{1}{\phi}\frac{\partial \phi}{\partial p}\bigg|_{T_0} \tag{2.8}$$

This equation, which is called the rock compressibility equation, indicates how the porosity of the porous media changes as a function of pressure. Since equations (2.7) and (2.8) are first-order differential equations, we require to specify the integration constants. We pose

$$\rho|_{p=p_0} = \rho_0, \quad \phi|_{p=p_0} = \phi_0 \tag{2.9}$$

With these equations we are able to rewrite the accumulation term as

$$\frac{\partial(\rho\phi)}{\partial t} = \left(\rho\frac{\partial \phi}{\partial p} + \phi\frac{\partial \rho}{\partial p}\right)\frac{\partial p}{\partial t} = \rho\phi(c_l + c_r)\frac{\partial p}{\partial t} = \rho\phi c_t\frac{\partial p}{\partial t}, \tag{2.10}$$

where $c_t = (c_l + c_r)$ is referred to as the total compressibility. If we assume slight compressibility, we allow for the utilization of the compressibility factor. Now equation (2.6) can be written as

$$\underbrace{-\nabla\cdot\left[\frac{\rho}{\mu}\vec{\mathbf{K}}(\nabla p - \rho g\nabla d)\right]}_{\text{Flux-term}} + \underbrace{\rho\phi c_t\frac{\partial p}{\partial t}}_{\text{Accumulation term}} - \underbrace{\rho q'''}_{\text{Source term}} = 0. \tag{2.11}$$

For the rest of this thesis, we will assume fluid and rock incompressibility. This means that the accumulation term vanishes. Therefore we are left with the following equation

$$-\nabla\cdot\left[\frac{\rho}{\mu}\vec{\mathbf{K}}(\nabla p - \rho g\nabla d)\right] - \rho q''' = 0. \tag{2.12}$$

Finally, we will ignore gravitational effects and capillary effects leading to

$$-\nabla\cdot\left[\frac{\rho}{\mu}\vec{\mathbf{K}}(\nabla p)\right] - \rho q''' = 0, \tag{2.13}$$

or

$$-\nabla\cdot(\boldsymbol{\lambda}_t\cdot\nabla p) = q, \tag{2.14}$$

where

$$\boldsymbol{\lambda}_t = \frac{\vec{\mathbf{K}}}{\mu}, \tag{2.15}$$

is referred to as the mobility tensor.

This equation is of second-order in the spatial coordinate $\mathbf{x}$. Also, since there is no time derivative, this is a stationary equation. Hence, to solve this equation we require two boundary conditions. Multiple choices are possible.

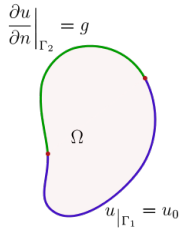


Figure 2.3: Visualization of mixed boundary conditions.

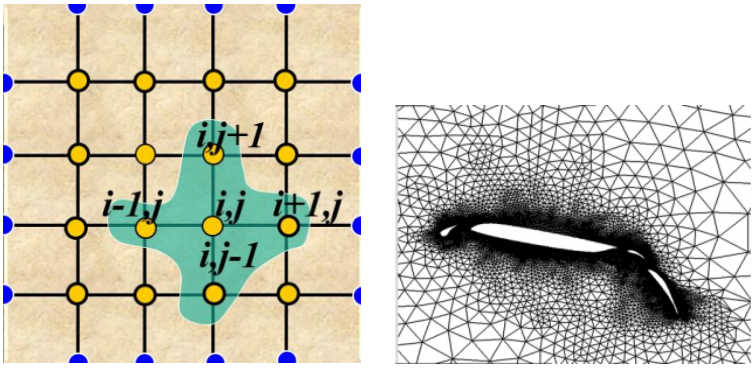
Common boundary conditions are prescribed conditions, known as Dirichlet conditions or prescribed outflow at the boundary, known as Neumann conditions. Also mixed boundary conditions are possible, which are referred to as Robin conditions, but in practice these are generally not applied. It is also possible to have different boundary conditions on different sections of the boundary. The total computational domain, or porous media, is written as Ω and the boundary of this domain is expressed as $\partial\Omega$. We assume a simple domain, i.e. a domain that is unfractured and has no holes or discontinuous regions. An example of a boundary setting is found in Figure [2.3][38]. In summary we have the following conditions.

$$\begin{cases} p(\mathbf{x},t)\big|_{\Gamma_0} & = p^0(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega \\ \frac{\partial p(\mathbf{x},t)}{\partial \mathbf{n}}\Big|_{\Gamma_1} & = p^1(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega \end{cases} \tag{2.16}$$

For reservoir simulation one typically employs no-flow boundary conditions.

2.2. Solving the fine scale pressure equation

Before we are able to discuss the framework of multiscale methods, we need to understand how to solve the pressure equation on a single resolution. Multiple strategies and techniques exist to solve equation (2.14) on the domain Ω, which for reservoir simulation will be the reservoir domain. Among these techniques one can find the Finite Volume Method (FVM), the Finite Difference Method (FDM) and the Finite Element Method (FEM). Each method has it's advantages and disadvantages. In this thesis we will not discuss these differences, but instead we refer to [20, 38], for detailed explanations of the distinct methods. All methods, however, work by imposing a grid, or mesh, on the domain Ω. This divides the full domain into smaller sub-domains, or cells, that we refer to as sub-intervals (FDM), control volumes (FVM) or elements (FEM), that interact with other cells in the mesh. See Figure [2.4]. Throughout this thesis we will use the finite volume method as this method preserves mass balance, however we note that the other numerical techniques may also be applied.



(a) Example of Mesh used FVM (b) Example of Mesh used in FEM

Figure 2.4: Different possibilities of meshes of different domains Ω [9].

One may recognize a steady-state heat equation without external sources in equation (2.14). In the heat equation $\boldsymbol{\lambda}_t$ is replaced with κ, being the heat conduction. Generally, the heat conduction is taken as a constant over the complete domain. However, instead of heat we consider pressure in this elliptic equation. This is directly where the challenge in solving equation (2.14) lies.

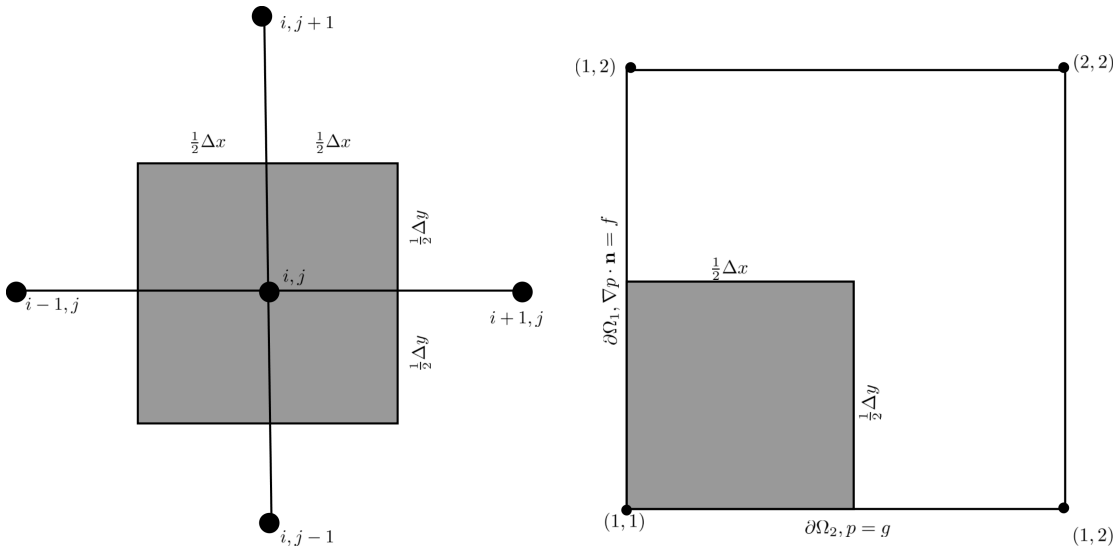
We consider the elliptic problem

$$-\nabla \cdot (\boldsymbol{\lambda} \cdot \nabla p) = q, \tag{2.17}$$

on a arbitrary domain Ω, $\partial\Omega = \partial\Omega_0 \cap \partial\Omega_1$, with boundary conditions, $\nabla p \cdot \mathbf{n} = p^1(\mathbf{x})$ and $p(\mathbf{x}) = p^0(\mathbf{x})$ on boundaries $\partial\Omega_1, \partial\Omega_0$ respectively. Let $\Omega \subset \mathbb{R}^F$ be the fine grid consisting of F-control volumes. For simplicity we assume for now that $\Omega = L \times L$ square domain and we impose an equidistant grid, that is $\Delta x = \Delta y$. In this manner the grid is of the same form as in Figure [2.4a].

To solve equation (2.17), we integrate over all F control volumes Ω_i. We distinguish three different categories of control volumes.

1. Internal cells: These are cells that have no connection to a boundary.
2. Boundary cells: These are cells that have exactly one connection to a boundary.
3. Corner cells: These are cells that have exactly two connections to two different boundaries.



(a) Control Volume of an interior cell. (b) Control volume of a corner cell.

Figure 2.5: Visualizations of two distinct control volumes used for integration in the FVM method.

In Figure [2.5], two examples of control volumes are given. Together with Gauss' theorem, we find for an internal cell (i, j), or control volume number i,

$$\int_{\Omega_i} \nabla \cdot (\boldsymbol{\lambda} \cdot \nabla p) d\Omega_i = \int_{\Omega_i} q d\Omega_i \tag{2.18a}$$

$$\Leftrightarrow \int_{\partial\Omega_i} (\boldsymbol{\lambda} \cdot \nabla p) \cdot \mathbf{n} d\Gamma_i = \int_{\Omega_i} q d\Omega_i \tag{2.18b}$$

$$\Leftrightarrow \sum_o \int_{\partial\Omega_i^o} (\boldsymbol{\lambda} \cdot \nabla p) \cdot \mathbf{n} d\Gamma_o = \int_{\Omega_i} q d\Omega_i \tag{2.18c}$$

Where $o = \{S, W, N, E\}$ is the set of edges of control volume i. As example, if we take the west edge of control volume i we have $n = (-1, 0)^\top$, using the central differences approximation we have

$$\int_{\partial\Omega_i^W} (\boldsymbol{\lambda} \cdot \nabla p) \cdot (-1, 0)^\top dy \approx -\boldsymbol{\lambda}_{i-\frac{1}{2},j} \left(\frac{p_{i-1,j} - p_{i,j}}{\Delta x} \right) \Delta y, \tag{2.19}$$

where $\boldsymbol{\lambda}_{i-\frac{1}{2},j}$ is defined as the harmonically averaged permeability from cell (i, j) to cell $(i-1, j)$,

$$\boldsymbol{\lambda}_{i-\frac{1}{2},j} = \frac{2}{\frac{1}{\boldsymbol{\lambda}_{i-1,j}} + \frac{1}{\boldsymbol{\lambda}_{i,j}}} \tag{2.20}$$

Computing the other terms leads to an approximation of

$$-\boldsymbol{\lambda}_{i-\frac{1}{2},j}\left(\frac{p_{i-1,j}-p_{i,j}}{\Delta x}\right)\Delta y-\boldsymbol{\lambda}_{i+\frac{1}{2},j}\left(\frac{p_{i+1,j}-p_{i,j}}{\Delta x}\right)\Delta y-\boldsymbol{\lambda}_{i,j-\frac{1}{2}}\left(\frac{p_{i,j-1}-p_{i,j}}{\Delta y}\right)\Delta x-\boldsymbol{\lambda}_{i,j+\frac{1}{2}}\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta y}\right)\Delta x = q_{i,j}\Delta x\Delta y \tag{2.21}$$

By setting

$$T_{i-\frac{1}{2},j}=\boldsymbol{\lambda}_{i-\frac{1}{2},j}\frac{\Delta y}{\Delta x},\quad T_{i,j-\frac{1}{2}}=\boldsymbol{\lambda}_{i,j-\frac{1}{2}}\frac{\Delta x}{\Delta y}, \tag{2.22}$$

we find an equation for each internal cell (i,j).

$$\left(T_{i-\frac{1}{2},j}+T_{i+\frac{1}{2},j}+T_{i,j-\frac{1}{2}}+T_{i,j+\frac{1}{2}}\right)p_{i,j}-T_{i-\frac{1}{2},j}p_{i-1,j}-T_{i+\frac{1}{2},j}p_{i+1,j}-T_{i,j-\frac{1}{2}}p_{i,j-1}-T_{i,j+\frac{1}{2}}p_{i,j+1}=q_{i,j}V \tag{2.23}$$

Note that equation (2.23) can also be expressed as an matrix-vector multiplication

$$\begin{bmatrix} -T_{i,j-\frac{1}{2}} & \cdots & -T_{i-\frac{1}{2},j} & \left(T_{i-\frac{1}{2},j}+T_{i+\frac{1}{2},j}+T_{i,j-\frac{1}{2}}+T_{i,j+\frac{1}{2}}\right) & -T_{i+\frac{1}{2},j} & \cdots & -T_{i,j+\frac{1}{2}} \end{bmatrix}\begin{bmatrix} p_{i,j-1}\\ \vdots\\ p_{i-1,j}\\ p_{i,j}\\ p_{i+1,j}\\ \vdots\\ p_{i,j+1}\end{bmatrix}=q_{i,j}V \tag{2.24}$$

Now that we have an equation for the internal cells, we must also find equations for the boundary and corner cells. For illustration, we only give the derivation of a corner cell since boundary cell have an equivalent derivation. Consider corner point (i,j). Suppose that in our control volume, the west boundary corresponds to $\partial\Omega_1$ and the south boundary corresponds to $\partial\Omega_2$. Then,

$$\int_{\Omega_i^W}(\boldsymbol{\lambda}\cdot\nabla p)\cdot\mathbf{n}d\Gamma_W=\int_{\Omega_i^W}\boldsymbol{\lambda}\frac{\partial p}{\partial\mathbf{n}}dy\approx\boldsymbol{\lambda}_{i,j+\frac{1}{2}}f_i\Delta y, \tag{2.25}$$

and

$$\int_{\Omega_i^S}(\boldsymbol{\lambda}\cdot\nabla p)\mathbf{n}d\Gamma_S=\int_{\Omega_i^S}(\boldsymbol{\lambda}\cdot\nabla p^0(\mathbf{x}))\mathbf{n}d\Gamma_S\approx\boldsymbol{\lambda}_{i+\frac{1}{2},j}\frac{p_{i,j}^0-p_{i-1,j}^0}{\Delta y}\Delta x \tag{2.26}$$

And so we are able to find the following equation for the left corner point $(1,1)$

$$\left(T_{1+\frac{1}{2},1}+T_{1,1+\frac{1}{2}}\right)p_{1,1}-T_{1+\frac{1}{2},j}p_{2,1}-T_{1,1+\frac{1}{2}}p_{1,2}=q_{1,1}V+\boldsymbol{\lambda}_{1,1-\frac{1}{2}}f_{1,1}\Delta y+\boldsymbol{\lambda}_{1+\frac{1}{2},1}\frac{p_{1,1}^0-p_{2,1}^0}{\Delta y}\Delta x \tag{2.27}$$

Also, (2.27), may be expressed with a matrix-vector multiplication

$$\begin{bmatrix}\left(T_{1+\frac{1}{2},1}+T_{1,1+\frac{1}{2}}\right) & -T_{1+\frac{1}{2},1} & \cdots & -T_{1,1+\frac{1}{2}}\end{bmatrix}\begin{bmatrix}p_{1,1}\\ p_{2,1}\\ \vdots\\ p_{1,2}\end{bmatrix}=q_{1,1}V+\boldsymbol{\lambda}_{1,1-\frac{1}{2}}f_{1,1}\Delta y+\boldsymbol{\lambda}_{1+\frac{1}{2},1}\frac{p_{1,1}^0-p_{2,1}^0}{\Delta y}\Delta x \tag{2.28}$$

If we combine all the equations that correspond to the internal cells, boundary cells and different corner cells, We find a system

$$\mathbf{Tp}=\mathbf{q} \tag{2.29}$$

In literature the matrix $\mathbf{T}$ is also referred to as the transmissibility matrix. From this system we can calculate our unknown pressure $\mathbf{p}$.

As discussed in [22], it is known that physically $\boldsymbol{\lambda}_t$ generally has highly heterogeneous geological properties. To capture the effect of this physical parameter we require a highly accurate discretized solution. Higher accuracy is achieved by imposing a more dense grid, or fine scale grid, i.e. a grid that consists of a high number of sub-domains. Current high-resolution geomodels may consist 10^6-10^{10} cells.

However, imposing a denser grid comes with a growth in computational cost. Since at this time conventional simulators are capable of handling a maximum of $10^5 - 10^6$ cells, there is a gap in the resolution. Thus, we can argue that for reservoir simulation, the costs to achieve an accurate enough solution is simply too high. Therefore we will investigate solutions to this problem in Chapter [4].

3

Continuous Optimization

In this chapter we will discuss different optimization methods for constrained systems. Firstly, the mathematical formulation of the framework used in the continuous optimization setting is explained. Here, we discuss the difference between Gradient-free and Gradient-based optimization algorithms, and two examples of algorithms of the second category are presented.

3.1. Introduction

3.2. Types of optimization methods

One can distinguish two type of optimization methods. The first set are the optimization problems for which the optimal value may be computed directly and analytically. One way of doing this is via the Lagrange Multiplier Method. For large scale constrained discretized partial differential equations however, it is not possible to analytically solve the optimization problem. Usually an iterative procedure is required to compute the optimal solution [17, 36]. Generally, one can categorize the iterative methods into two different types, either gradient-based methods or gradient-free methods. Gradient-based methods are methods that require the gradient of an objective function with respect to the to parameters that you want to optimize for. Gradient-based methods generally find local optima, instead of global optima. Gradient free methods however, aim at finding a global optimum. The price we pay for this is the fact that many more function evaluations are required [21, 41]. In the setting of reservoir simulation a function evaluation is equivalent to performing a full simulation run and therefore is computationally very expensive. As result, in reservoir simulation we prefer the gradient-based optimization methods above the gradient-free methods. In this thesis we will only discuss gradient-bases methods. In this section we will explain the basic knowledge required for optimization purposes.

3.2.1. Introduction into optimization theory

First of all, we briefly explain the basics of optimization techniques. For further details and mathematical justification we refer to the works found at [8, 17]. We consider unconstrained and constrained optimization.

Unconstrained optimization

The optimization problem of an objective function with respect to the variable $\mathbf{x}$ in most general form can be written as

$$\min_{\mathbf{x}\in\mathcal{X}} \mathcal{J}(\mathbf{x}) \tag{3.1}$$

In words this means; find the minimum value of function $\mathcal{J}$, amongst all possible values of $\mathbf{x} \in \mathcal{X}$. The set $\mathcal{X}$ is referred to as the Feasible Region and we say that $\mathbf{x}$ is called feasible if $\mathbf{x} \in \mathcal{X}$, else, $\mathbf{x}$ is called infeasible. The objective function is in most cases a multivariate. Since there are no constraints or demands that need to be met, this is also referred to as unconstrained optimization. Note that the problem for maximizing the objective function is similar to minimizing the function $-\mathcal{J}$. An optimal point

$\mathbf{x}^0$ is called a stationary (or critical) point and satisfies the first-order necessary condition

$$\nabla \mathcal{J}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}^0} = \left[\frac{\partial \mathcal{J}}{\partial x_1} \cdots \frac{\partial \mathcal{J}}{\partial x_n}\right]\Bigg|_{\mathbf{x}=\mathbf{x}^0} = \mathbf{0}, \tag{3.2}$$

and the second-order sufficient condition

$$\frac{\partial^2 \mathcal{J}(\mathbf{x})}{\partial \mathbf{x}^2}\Bigg|_{\mathbf{x}=\mathbf{x}^0} = \begin{bmatrix} \frac{\partial^2 \mathcal{J}}{\partial x_1^2} & \frac{\partial^2 \mathcal{J}}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 \mathcal{J}}{\partial x_1 \partial x_n} \\ \frac{\partial^2 \mathcal{J}}{\partial x_2 \partial x_1} & \frac{\partial^2 \mathcal{J}}{\partial x_2^2} & \cdots & \frac{\partial^2 \mathcal{J}}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{J}}{\partial x_n \partial x_1} & \frac{\partial^2 \mathcal{J}}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 \mathcal{J}}{\partial x_n^2} \end{bmatrix}\Bigg|_{\mathbf{x}=\mathbf{x}^0} \geq 0 \tag{3.3}$$

The matrix as in equation (3.3) is called the Hessian. If the Hessian has negative eigenvalues, the point $\mathbf{x}^0$ indicates an maximum, while positive eigenvalues indicates an minimum. If among the eigenvalues there are eigenvalues with 0 value, $\mathbf{x}^0$ constitutes a saddle point.

A convex set S is defined as an set in where $s = \beta s_1 + (1-\beta)s_2 \in S, \forall s_1, s_2 \in S, \beta \in (0,1)$. The epigraph of a function is the set of all positions above this graph. A convex function is a function if the epigraph forms a convex set. If not, the set is called concave. Examples of both sets are found in Figure [3.1]. If a set is convex the necessary condition is also an sufficient condition. In reservoir simulation however, we rarely deal with convex problems.

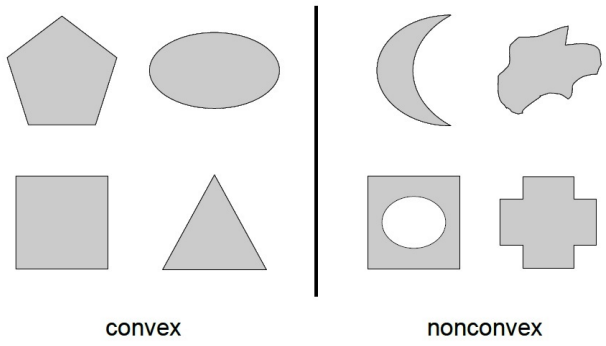


Figure 3.1: Left: example of a convex set. Right: example of a concave set, not all combinations between two points are contained in the set [17].

Equality constrained optimization

In contrast with the fact that we rarely deal with convex problem, it is very common to find optimization problems subjected to constraints. These constraints often have practical or physical interpretations. Even though in this thesis we do not consider these types of constraints, in the view of reservoir simulation optimization, one can think of examples of typical constraints as bounds on the pressures or injections rates. The effect of posing extra constraints is that the set of all feasible solutions $\mathcal{X}$ is limited to a smaller set of feasible solutions $\mathcal{X}^*$. See Figure [3.2]. This optimization problem can be denoted as

$$\begin{aligned} &\underset{\mathbf{x} \in \mathcal{X}^*}{\text{minimize}} \quad \mathcal{J}(\mathbf{x}) \\ &\text{subject to} \quad c_v(\mathbf{x}) = 0 \quad v = 1 \cdots V \end{aligned} \tag{3.4}$$

where $c_v(\mathbf{x})$ indicate the constraints equations. In total there is a number of V constraints. Solving this problem also requires handling of the constraints and hence, we will have to find a different method of calculating the optimum.

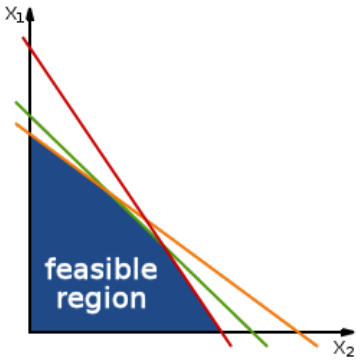


Figure 3.2: Here we can see an example of a feasible region. The feasible region is bounded by the three constraints. Without these constraints the feasible region would comprise of the entire upper-right quadrant.

One method to solving (3.4) is by using the Lagrange Multipliers Method. In this method one augments the objective function $\mathcal{J}$ with the different constraints, which are multiplied by the Lagrange multipliers $\boldsymbol{\lambda}_v \in \boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_V)^\mathsf{T}$. The obtained equation is called the Lagrangian, and may be written in general form as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \mathcal{J}(\mathbf{x}) + \sum_{v=1}^{V} \lambda_v c_v(\mathbf{x}) \tag{3.5}$$

Note that if $x \in \mathcal{X}^*$, the constraints are met and therefore equation (3.5) leads to the same minimum as in (3.1). The first-order necessary conditions for problem (3.4) are given by stationarity of the augmented objective function, just as stationarity provided the necessary conditions for problem (3.1). This leads to the following system of equations.

$$\begin{cases} \frac{\partial L(\mathbf{x},\boldsymbol{\lambda})}{\partial \mathbf{x}} = \mathbf{0} \\ \frac{\partial L(\mathbf{x},\boldsymbol{\lambda})}{\partial \lambda_v} = \mathbf{0} \quad \forall v \in [1, V] \end{cases}$$

This system of equations is also referred to as the Euler-Lagrange equations. To give an interpretation of the meaning of the Lagrange Multipliers we write our constraints as a vector $\mathbf{c} = (c_1, \cdots, c_V)^\mathsf{T}$ and we consider a perturbation in the constraints as

$$\hat{\mathbf{c}}(\mathbf{x}) = \mathbf{c}(\mathbf{x}) + \delta\mathbf{c}(\mathbf{x}), \tag{3.6}$$

which leads to a perturbation in the modified objective function

$$\hat{L}(\mathbf{x}, \boldsymbol{\lambda}) = \mathcal{J} + \boldsymbol{\lambda}^\mathsf{T} \left[\mathbf{c}(\mathbf{x}) + \delta\mathbf{c}(\mathbf{x})\right] \tag{3.7}$$

The difference between non-perturbed and perturbed constraint equations of the augmented objective function is given by

$$\delta L(\mathbf{x}, \boldsymbol{\lambda}) = \hat{L}(\mathbf{x}, \boldsymbol{\lambda}) - L(\mathbf{x}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^\mathsf{T} \delta\mathbf{c}(\mathbf{x}) \tag{3.8}$$

In the non-perturbed optima we may write $L(\mathbf{x}^0, \boldsymbol{\lambda}^0) = \mathcal{J}(\mathbf{x}^0)$ and in the perturbed optima we can write $\hat{L}(\hat{\mathbf{x}}^0, \hat{\boldsymbol{\lambda}}^0) = \mathcal{J}(\hat{\mathbf{x}}^0)$ such that we also have that

$$\mathcal{J}(\hat{\mathbf{x}}^0) - \mathcal{J}(\mathbf{x}^0) = \boldsymbol{\lambda}^\mathsf{T} \delta\mathbf{c}, \tag{3.9}$$

or,

$$\delta\mathcal{J} = \boldsymbol{\lambda}^\mathsf{T} \delta\mathbf{c} \tag{3.10}$$

Thus we can conclude that for all constraints the Lagrange multipliers give a measure of the effect of perturbing the constraint $i \in [1, \cdots, V]$, or state of the system, on the objective function. Moving along $c_i(\mathbf{x})$ one unit will change the objective function with value λ_i.

3.3. Gradient-based numerical optimization methods

3.3.1. Introduction

The Lagrange Multiplier method is generally used when the solution can be computed directly, i.e. if the problem is linear and small. However, when the number of parameters is high and the system of equations is large, this is not a trivial task. For the reservoir simulation models, this is usually the case. In this section we will briefly explain two numerical optimization methods used to cope with this issue. These frameworks of numerical iterative optimization methods rely on a search direction and a step size.

Both methods use the same underlying principle. Suppose that we seek a minimizer (or maximizer) $\mathbf{x}^*$ of $f(\mathbf{x})$. In an iterative framework we want to compute a sequence $\mathbf{x}_0, \cdots, \mathbf{x}_k, \cdots$ such that

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k), \qquad \lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}^* \tag{3.11}$$

In other words, we aim to get closer to the minimum with every iteration step, where the sequence is calculated with aid of the gradient of the objective function. Hence, this class of algorithms is called a gradient-based optimization methods. A visualization of this procedure is found in Figure [3.3],[28]. A classic metaphor for the gradient-based maximization procedure is of climbing down of the top of a hill as fast as possible, whilst being surrounded by fog. This is known as the steepest decent method, where steps are taken in the steepest downwards direction. In this case, the slope of the mountain can be seen as the gradient. This method is based on the interpretation where the descent direction enables the maximum decrease in the value of $f(\mathbf{x})$ possible. If we take enough steps, eventually we would reach the minimum. Now if we assume $\mathbf{x}_k = \mathbf{x}$ and we have that $\nabla f(\mathbf{x}) \neq 0$, or in other words, that $\mathbf{x}$ is not minimizing our function f, then we take a vector $\mathbf{p}$ such that

$$\langle \mathbf{p}, \nabla f(\mathbf{x}) \rangle = \mathbf{p}^\mathsf{T} \nabla f(\mathbf{x}) < 0 \tag{3.12}$$

Then, $\mathbf{p}$ is proportional to the directional derivative of $f(\mathbf{x})$, and because of equation (3.12), this implies that we can reduce the value of $f(\mathbf{x})$ by moving in the direction of $\mathbf{p}$. Therefore, $\mathbf{p}$ is also called the descent direction. Now that we have a descent direction, we can find a positive constant α, which in literature is referred to as the step length parameter such that

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p} \tag{3.13}$$

satisfies the monotonic decreasing property as in equation (3.11). Now we have found a iterative framework for finding a minimum. There are multiple conditions to define the convergence of the algorithm and therefore set the stopping criteria. The question that remains is; how do we choose $\mathbf{p}$ and α? We will discuss two basic methods. For this master thesis however, only the framework of the gradient-based methods is important. No study is performed amongst the pros and cons of the different methods.

3.3.2. Steepest decent method

The first of the two methods we will discuss is the steepest decent method [4]. If we recall that $\nabla f(\mathbf{x})$ represents the direction of maximum rate change in $f(\mathbf{x})$, we easily see that $\mathbf{p} = -\nabla f(\mathbf{x})$ would guarentee a maximum reduction locally. Then

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}) \tag{3.14}$$

is known as the steepest descent method. What remains is how to find the best value of α_k at the current operation point and along the chosen direction $-\nabla f(\mathbf{x})$. In other words, which value of α decreases the value of $f(\mathbf{x} - \alpha \nabla f(\mathbf{x}))$ most? If we define $z(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}))$, then finding α_k corresponds to minimizing $z(\alpha)$. Hence we solve the one-dimensional minimization problem, which is sometimes called the line search,

$$\frac{dz}{d\alpha} = -\left[\nabla f(\mathbf{x} - \alpha \nabla f(\mathbf{x}))\right]^\mathsf{T} \nabla f(\mathbf{x}) \tag{3.15}$$

This method is proven to guarantee the location of a minimum, if one exists. Unfortunately, the drawback of the method is its slow convergence rate.

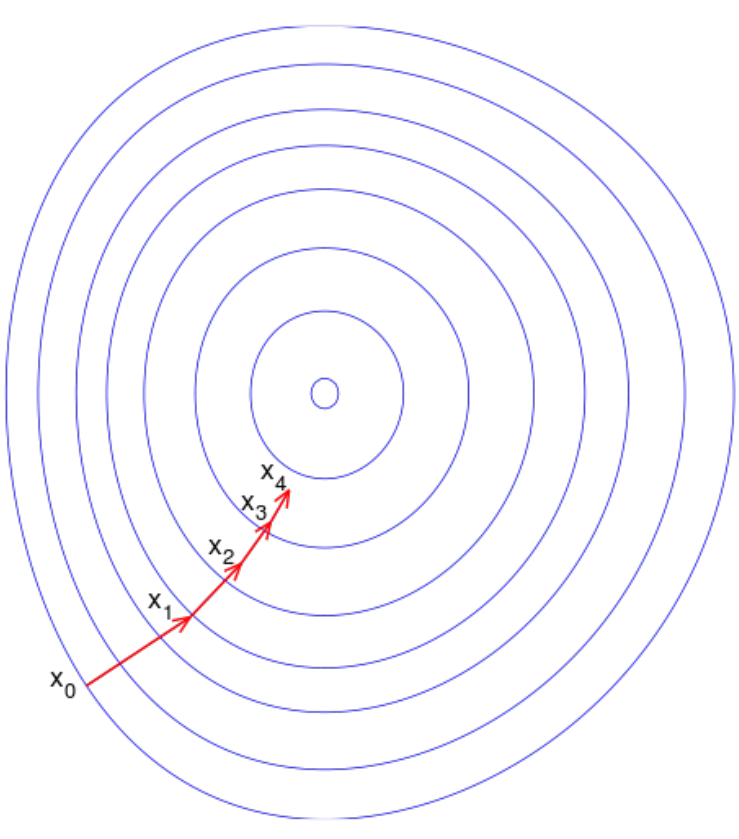


Figure 3.3: Visualization of a numerical iterative gradient optimization procedure on the level sets of a function. Note how each iteration a different direction and step length is determined to find the next point.

3.3.3. Newton's method

The second method is based upon the idea to approximate $f(\mathbf{x})$ by a quadratic second-order Taylor expansion and to minimize this function. The minimizer of this function will define the new operating point. This method is called Newton's method. It uses the same two quantities, α being the step size and $\mathbf{p}$, being the search direction. We define

$$m(\mathbf{p}) = f(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + [\nabla f(\mathbf{x}_k)]^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top [\nabla^2 f(\mathbf{x}_k)]\mathbf{p}, \tag{3.16}$$

as the second order quadratic approximation. Then the gradient of this function is given by

$$\nabla m(\mathbf{p}) = \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\mathbf{p} \tag{3.17}$$

Also we have the condition

$$\nabla^2 m(\mathbf{p}) = \nabla^2 f(\mathbf{x}_k). \tag{3.18}$$

If we want to compute the minimum of $m(\mathbf{p})$, we force the gradient to be zero. Hence we find a system of linear equations to be solved

$$\nabla^2 f(\mathbf{x}_k)\mathbf{p} = -\nabla f(\mathbf{x}_k), \tag{3.19}$$

which is also known as the Newton's equation. Solving this equation gives the Newton direction $\mathbf{p}*$ from which the new point is generated

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p_k}*, \tag{3.20}$$

where α is found in the same manner as in the Steepest Descent method. The benefit of this method is a faster convergence rate compared to the Steepest Descent Method. However, the drawback is that computing $\nabla^2 f(\mathbf{x}_k)$ is extremely expensive and that the method is know to be unstable and doesn't always converge [21].

3.3.4. Gradient accuracy and convergence

In the previous section we have seen that both methods requires gradient information. However, in some cases the computation of gradient information can be computationally very expensive. This generally is the case when a lot of parameters are involved. If this is the case, using these methods can be computationally inefficient.

To resolve this issue, we note that from a optimization point of view, we do not require the exact direction of the gradient, a rough estimation will lead to similar results [10, 25]. As long as the gradient points to the correct up- or downwards direction, the optimization process is able to converge to the correct maximum or minimum. This is especially true for the first part of the iteration process, or the steep region, where big steps towards the optimum are taken. As the iterative process approaches the optimum however, more accurate gradients need to be computed, in order to evaluate a precise stopping criteria [26].

Different ways of computing the derivative information are available. First of all we can compute the gradient on the full model domain. This is called the fine-scale gradient. Because of the fact that the number of parameters is high, computing this information is computationally demanding. To resolve this issue we can estimate the gradient by using a multiscale framework [25]. This decreases the accuracy of the gradient, but improves the computational effort. In this thesis we investigate if we can control the accuracy of our gradient estimation by introducing an iterative multiscale framework [26]. This increases the accuracy compared to the multiscale gradient, but as it uses a heavy computational smoothing step we trade the increase in accuracy for a decrease in the computational benefits of using a multiscale framework. The computational costs of the smoothing step of the iterative multiscale model can be decreased by using a iterative goal oriented multiscale strategy. In this strategy, only positions relevant to a certain goal are used in the smoothing step. In this thesis we will formulate the framework for this method and we show the possible benefits of employing this method.

4

Multiscale Methods

In this section we introduce the upscaling and multiscale methods (MS) and explain to the reader why these methods have been developed. Next the properties of upscaling methods and the difficulties present in these models are discussed. Then the mathematical framework for the multiscale methods is presented and their advantages to upscaling methods is discussed. After this, we present the mathematical framework for the Algebraic Multiscale Method (AMS). After the Multiscale framework has been stated, we use the knowledge of chapter [2] to state a formulation of the multiscale model that we aim to solve.

4.1. Upscaling methods

Work from [22] introduces a solution to resolve the problems associated with a small grid-block size. These methods are called upscaling methods. In upscaling methods, the equations are solved on a coarser grid, using upscaled parameters, i.e. parameters that represent the system at a coarser level. There are multiple possibilities of representing the fine scale heterogeneities at a coarse scale. One can take combinations of arithmetic, geometric or harmonic averaging. One can apply power averaging or one can apply equivalent permeabilities. This process can be visualized in Figure (4.1). There are many difficulties when considering the upscaling method, such as the loss of detail, lack of robustness. Furthermore it is known that the method has trouble dealing with complex grids, i.e. grids with holes or fractures.

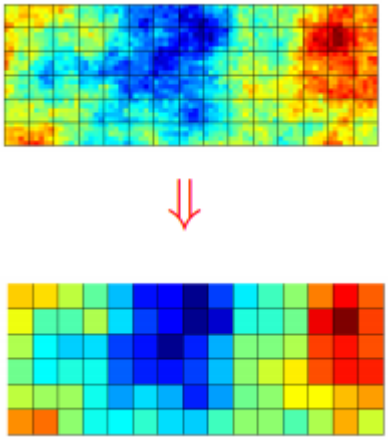


Figure 4.1: Capturing the fine scale heterogeneities at a coarse scale mesh [13].

4.2. Multiscale methods

4.2.1. Introduction

To improve the upscaling methods, multiscale methods have been developed [22, 24]. Similar to the upscaling methods, multiscale methods aim to reduce the computational cost by solving the governing equations on a coarse grid. Yet, where the upscaling methods lack detail, the multiscale method allows us to reconstruct the fine scale pressure field from a coarse scale pressure solution and hence, achieve the same resolution as the fine scale solution, but with lower computational costs. The multiscale methods are designed to compute accurate solutions for highly heterogeneous coefficients $\boldsymbol{\lambda}$. The permeability coefficient has the same resolution as the fine scale solution, that is, this coefficient is different in each different cell, but is constant inside of each cell. Multiple categories of multiscale methods exist. There are multiscale finite-element (MSFE) methods [23], mixed multiscale finite-elements (MMSFE) methods [2, 23] and multiscale finite-volume (MSFV) methods [23, 43, 44]. The two latter provide us with conservative fine scale veloctiy fields, which is a requirement to solve the transport equations. The accuracy of the method has been demonstrated by multiple examples [13]. However, the error present are most prominent in the examples with extremely high permeability contrasts, fractures, obstacles or oddly-shaped domains. These errors mainly come from the so-called localization error, which we will discuss in later sections more thoroughly. To decrease the errors by we can employ an iterative multiscale strategy, which is discussed in chapter [5].

4.2.2. Grids

For the derivation of the multiscale method, we consider the MSFV-method and follow the line of reasoning as presented in [12, 13, 43]. We impose a coarse grid $\bar{\Omega}_k, k \in [1, M]$ consisting of M control volumes. Next to this we also impose a dual coarse grid $\tilde{\Omega}^h, h \in [1, N]$ consisting of N control volumes. Each coarse control volume $\bar{\Omega}_k$ contains exactly one node $\mathbf{x}_k$ of the dual coarse grid in it's interior. The dual coarse grid is constructed by connecting the centers of the coarse grid volumes. Each dual control volume consists of 2^d blocks, where d is the dimension of the problem. In Figure [4.2] one can find an example of the different grids present in the multiscale discretization.

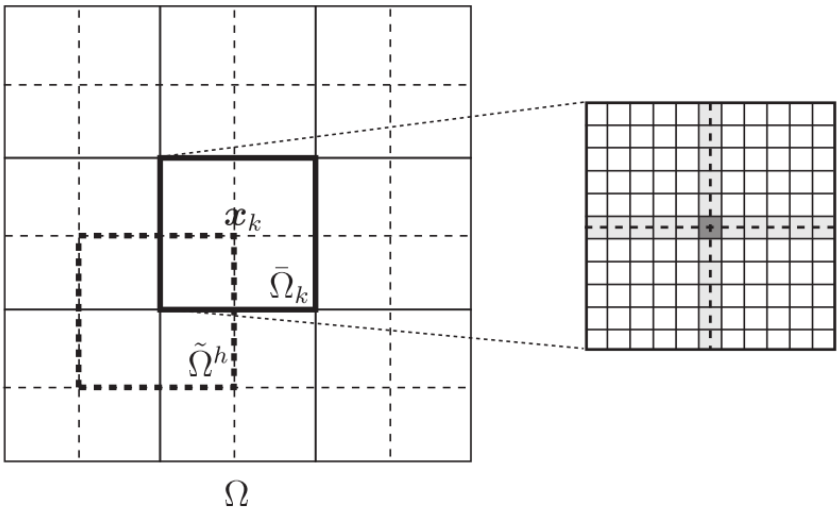


Figure 4.2: The grids employed in the finite volume multiscale method. The coarse scale grid is indicated with solid black lines. The dual coarse grid are indicated with the dashed lines. To the right we have an enlarged coarse cell, containing the fine cells and exactly one coarse nod $\mathbf{x}_k$ [12].

4.2.3. Basis functions

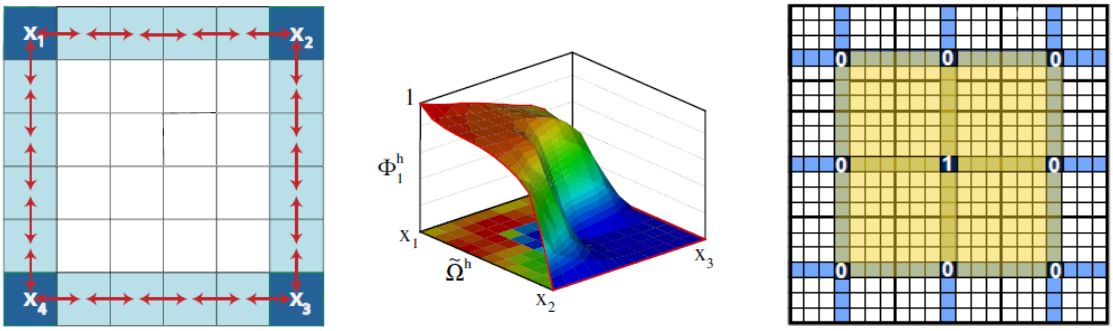
Next to the coarse and dual grids, basis functions play an important role in the multiscale method. These basis function can be seen as interpolators that are able to interpolate the coarse scale solution

to the fine scale resolution. The basis functions are obtained by solving

$$\begin{cases} \nabla \cdot (\boldsymbol{\lambda} \cdot \nabla \phi_k^h) = 0 & \text{in } \tilde{\Omega}^h \\ \nabla_{||} \cdot (\boldsymbol{\lambda} \cdot \nabla \phi_k^h)_{||} = 0 & \text{on } \partial\tilde{\Omega}^h \\ \phi_k^h(\mathbf{x}_j) = \delta_{hj} & \forall j \in [1, M] \end{cases} \tag{4.1}$$

where ϕ_k^h is the basis function associated with coarse node k in dual coarse block h. Solving equation (4.1) means that locally we solve equation (2.17) without the right-hand side. This is also called the localization assumption as the boundary condition reduces the problem a dimension. This assumption is the main contributing factor that causes error in this method. The Dirichlet condition $\phi_k^h(\mathbf{x}_j) = \delta_{hj}$ has a support over a maximum of 2^d dual grid cells. In the cells where there is no support we may easily conclude that $\phi_k^h = 0$. In Figure [4.3], one finds a visualization for the reduced boundary problem, a basis function example and the support of the basis function [13]. Furthermore, this also means that for each coarse node k there are 2^d non-zero basis functions ϕ_k^h. This means that the basis function at coarse node k is given by

$$\phi_k = \sum_{h=1}^{N} \phi_k^h \tag{4.2}$$



(a) Visualization of the reduced boundary problem.

(b) Visualization of an example basis function ϕ_1^h.

(c) Visualization of the support of basis function ϕ_k.

Figure 4.3: In this Figure one can find visualizations of key concepts regarding basis functions.

4.2.4. Compute the coarse grid solution

When we have calculated our basis functions, the next step is to calculate our coarse scale solution. In this section we will describe how to calculate the coarse scale pressure. We use the work of [44].

Denote the fine scale pressure solution with $\mathbf{p}'$ and the coarse scale pressure solution with $\bar{\mathbf{p}}$. To define a coarse scale system, we use previous knowledge to derive an expression for the coarse scale solution in terms of the fine scale solution. This is done with help of the basis functions. We may write the fine scale pressure solution, which we have solved in Chapter [2], as a product of the interpolative basis functions as

$$\mathbf{p} = \mathbf{p}' \approx \sum_{k=1}^{M} \phi_k \bar{\mathbf{p}}_k = \sum_{h=1}^{N} \left[\sum_{k=1}^{M} \phi_k^h \bar{\mathbf{p}}_k \right], \tag{4.3}$$

where we note that in equation (4.3), we do not use a correction factor such as presented in [13]. The correction factor is supposed to deal with the errors generated from the localization assumption, but since [43] states that the gain in convergence does not compensate for the additional computational costs, we decide to leave this out from out method completely.

Next, we substitute equation (4.3) into expression (2.17) and we integrate over a coarse cell $\bar{\Omega}_j, j \in [1, M]$. This gives

$$-\int_{\bar{\Omega}_j} \nabla \cdot (\boldsymbol{\lambda} \cdot \nabla p) d\bar{\Omega}_j = -\int_{\bar{\Omega}_j} \nabla \cdot \left(\boldsymbol{\lambda} \cdot \nabla \left(\sum_{h=1}^{N} \left[\sum_{k=1}^{M} \phi_k^h \bar{\mathbf{p}}_k \right] \right) \right) d\bar{\Omega}_j = -\int_{\bar{\Omega}_j} q d\bar{\Omega}_j \tag{4.4a}$$

$$= \int_{\partial \bar{\Omega}_j} \left(-\boldsymbol{\lambda} \cdot \nabla \left(\sum_{h=1}^{N} \left[\sum_{k=1}^{M} \phi_k^h \bar{\mathbf{p}}_k \right] \right) \right) \cdot \bar{\mathbf{n}}_j d\Gamma_j = -\int_{\bar{\Omega}_j} q d\bar{\Omega}_j \tag{4.4b}$$

$$\tag{4.4c}$$

where we use Gauss' theorem. We can simplify this, by interchanging the summations and the integral over coarse node j. This gives us $\forall j \in [1, M]$,

$$-\int_{\bar{\Omega}_j} \nabla \cdot (\boldsymbol{\lambda} \cdot \nabla p) d\bar{\Omega}_j = \sum_{k=1}^{M} \bar{\mathbf{p}}_k \sum_{h=1}^{N} \left(\int_{\partial \bar{\Omega}_j} \left(-\boldsymbol{\lambda} \cdot \nabla \phi_k^h \right) \cdot \bar{\mathbf{n}}_j d\Gamma_j \right) = -\int_{\bar{\Omega}_j} q d\bar{\Omega}_j \tag{4.5}$$

Now set

$$\mathbf{T}_{jk}^c = \sum_{h=1}^{N} \left(\int_{\partial \bar{\Omega}_j} \left(\boldsymbol{\lambda} \cdot \nabla \phi_k^h \right) \cdot \bar{\mathbf{n}}_j d\Gamma_j \right), \quad \mathbf{q}_j^c = \int_{\bar{\Omega}_j} q d\bar{\Omega}_j, \tag{4.6}$$

Then this leads to a linear system of the form

$$\mathbf{T}^c \bar{\mathbf{p}} = \mathbf{q}^c \tag{4.7}$$

with solution

$$\bar{\mathbf{p}} = (\mathbf{T}^c)^{-1} \mathbf{q}^c, \tag{4.8}$$

Note that $\mathbf{T}^c = \bar{\mathbf{T}}$, which represents the coarse scale transmissibility tensor. Furthermore, $\mathbf{q}^C = \bar{\mathbf{q}}$ represents the coarse scale source term. Finally, the fine scale pressure solution is solved from equation (4.3).

4.3. Algebraic multiscale method

To introduce the Algebraic Multiscale Method (AMS), we remind the reader that relation (4.3) holds, which algebraically boils down to

$$\mathbf{p}' \approx \mathbf{P} \bar{\mathbf{p}}, \quad \mathbf{P} = [\phi_1 \cdots \phi_M] \tag{4.9}$$

where $\mathbf{P}$ is referred to as the prolongation operator, since it prolongs the coarse scale solution to the fine scale grid. This means that if we know all the basis functions, one can simply perform a matrix-vector calculation to retrieve the fine scale solution. However in practice, solving (4.1) is not trivial. Therefore, in this section, we will present an algebraic method to calculate the prolongation operator, or equivalently the basis functions. We follow the work of [43].

4.3.1. Calculating the basis functions

If we consider a fine scale model, discretizing equation (2.17) leads to a system of the form $\mathbf{T}\mathbf{p} = \mathbf{q}$. For simplicity we consider a 2-dimensional problem. Then, the dual-coarse grid divides the fine cells into three distinct categories being: I (internal cells), E (edge cells) and V (vertex cells). An illustration of the different cells can be found in Figure [4.4].

The system $\mathbf{T}\mathbf{p} = \mathbf{q}$ is reordered in a wire-basket fashion [42] using a permutation matrix $\mathcal{G}$, such that it can be expressed as

$$\begin{bmatrix} \mathbf{T}_{II} & \mathbf{T}_{IE} & \mathbf{0} \\ \mathbf{T}_{EI} & \mathbf{T}_{EE} & \mathbf{T}_{EV} \\ \mathbf{0} & \mathbf{T}_{VE} & \mathbf{T}_{VV} \end{bmatrix} \begin{bmatrix} \mathbf{p}_I \\ \mathbf{p}_E \\ \mathbf{p}_V \end{bmatrix} = \begin{bmatrix} \mathbf{q}_I \\ \mathbf{q}_E \\ \mathbf{q}_V \end{bmatrix}, \tag{4.10}$$

where sub matrix $\mathbf{T}_{ij}$ represents the contribution of category i to category j. For example, $\mathbf{T}_{II}$ consists of the matrix-values of $\mathbf{T}$ that correspond to the cells that only have interaction with other internal cells.

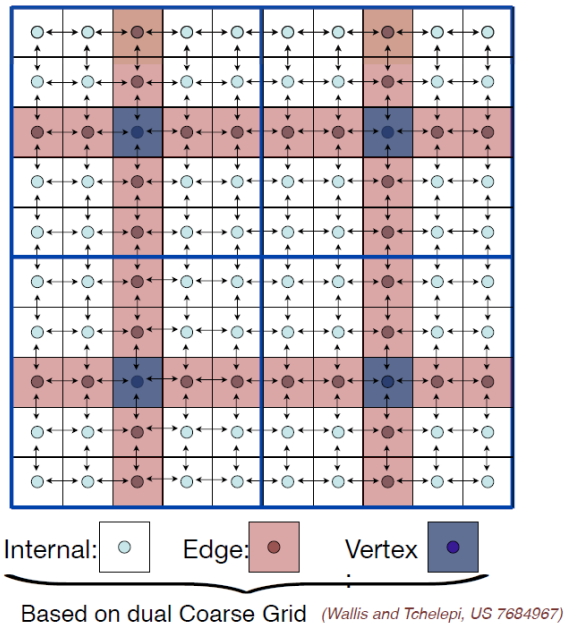


Figure 4.4: The three different cell categories imposed by the dual-coarse grid. The arrows indicate the contribution of cell k w.r.t. neighbouring cells

Now we modify the stencil for the edge cell, so that they represent the localization assumption. This means that the edge cells only contribute to other edge cells and vertex cells. This is done by setting $\mathbf{T}_{EI}$ to zero and it's corresponding part in $\mathbf{T}_{EE}$ to zero giving matrix $\hat{\mathbf{T}}_{EE}$. This leads to an expression for the multiscale approximation

$$\begin{bmatrix} \mathbf{T}_{II} & \mathbf{T}_{IE} & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{T}}_{EE} & \mathbf{T}_{EV} \\ \mathbf{0} & \mathbf{0} & \bar{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{p}_I \\ \mathbf{p}_E \\ \bar{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_I \\ \mathbf{q}_E \\ \bar{\mathbf{q}} \end{bmatrix}, \tag{4.11}$$

where $\bar{\mathbf{T}}\bar{\mathbf{p}} = \bar{\mathbf{q}}$ is exactly the coarse scale solution as expressed in the previous section. Since (4.11) is an upper-triangular system we know that the inverse exists and the inverse can efficiently be computed. Once we have our coarse scale solution, we can then calculate the pressures for the edges and interior cells by using backward substitution, hence

$$\mathbf{p}'_V = \bar{\mathbf{p}} \tag{4.12a}$$

$$\mathbf{p}'_E = (\hat{\mathbf{T}}_{EE})^{-1}\left(\mathbf{T}_{EV}\mathbf{p}'_V - \mathbf{q}_E\right) \tag{4.12b}$$

$$\mathbf{p}'_I = (\mathbf{T}_{II})^{-1}\left(\mathbf{T}_{IE}\mathbf{p}'_E - \mathbf{q}_I\right) = (\mathbf{T}_{II})^{-1}\left(\mathbf{T}_{IE}(\hat{\mathbf{T}}_{EE})^{-1}\left(\mathbf{T}_{EV}\mathbf{p}'_V - \mathbf{q}_E\right) - \mathbf{q}_I\right) \tag{4.12c}$$

In total, we can therefore express the solution as

$$\mathbf{p}' = \begin{bmatrix} \mathbf{p}'_I \\ \mathbf{p}'_E \\ \mathbf{p}'_V \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{II}^{-1}\mathbf{T}_{IE}\hat{\mathbf{T}}_{EE}^{-1}\mathbf{T}_{EV} \\ \hat{\mathbf{T}}_{EE}^{-1}\mathbf{T}_{EV} \\ \mathbf{I}_{VV} \end{bmatrix} \mathbf{p}'_V + \mathbf{T}_{corr}\mathbf{q}, \tag{4.13}$$

where $\mathbf{I}_{VV}$ is an $M \times M$ identity matrix and $\mathbf{T}_{corr}$ is a matrix corresponding to the correction function. However, just as in previous section we will neglect the contribution of this function. Now we compare equation (4.9) with equation (4.13) and remind ourselves that $\mathbf{p}'_V = \bar{\mathbf{p}}$ we may conclude that the prolongation operator and/or basis function algebraically can be expressed by

$$\mathbf{P} = \mathcal{G}\begin{bmatrix} \mathbf{T}_{II}^{-1}\mathbf{T}_{IE}\hat{\mathbf{T}}_{EE}^{-1}\mathbf{T}_{EV} \\ \hat{\mathbf{T}}_{EE}^{-1}\mathbf{T}_{EV} \\ \mathbf{I}_{VV} \end{bmatrix}, \tag{4.14}$$

where the pre-multiplication by $\mathcal{G}$ reorders the system to the natural ordering.

4.3.2. Calculating the coarse scale solution

In the previous section we explained a finite volume method to compute the coarse scale matrix $\bar{\mathbf{T}}$ and corresponding right hand side vector $\bar{\mathbf{q}}$. Once more we suppose that there is a fine scale discretization $\mathbf{Tp} = \mathbf{q}$. We introduce the restriction operator $\mathbf{R}$. This operator has the inverse function of the prolongation operator and takes you from the fine scale to the coarse scale. It may be defined as

$$\mathbf{R}(i,j) = \begin{cases} 1 & \text{if } \Omega_j \subset \bar{\Omega}_i \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1,M], j \in [1,F] \tag{4.15}$$

With this operator we are able to algebraically give an expression for the coarse scale system.

$$\mathbf{Tp}' = \mathbf{q} \xrightarrow{\text{Restriction}} \mathbf{RTp}' = \mathbf{Rq} \xrightarrow{\mathbf{p}'=\mathbf{P}\bar{\mathbf{p}}} (\mathbf{RTP})\bar{\mathbf{p}} = \mathbf{Rq} \Rightarrow \bar{\mathbf{T}}\bar{\mathbf{p}} = \bar{\mathbf{q}} \tag{4.16}$$

As a result, we easily have that

$$\bar{\mathbf{p}} = (\bar{\mathbf{T}})^{-1}\bar{\mathbf{q}} = (\mathbf{RTP})^{-1}(\mathbf{Rq}) \tag{4.17}$$

Now, the fine scale solution can be computed

$$\mathbf{p}' = \mathbf{P}\bar{\mathbf{p}} = \mathbf{P}(\mathbf{RTP})^{-1}(\mathbf{Rq}) \tag{4.18}$$

To conclude, we have now fully described an algebraic formulation to solve an elliptic pressure equation in a multiscale framework.

4.4. The multiscale forward model

Now that we have stated and explained the general multiscale method and found an efficient manner to compute our basis functions, in this section we will focus on the forward model that we aim to solve with a multiscale strategy. By using the physical model as in chapter [2], we write these equations in a generic form of

$$\mathbf{g_F}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{0}, \tag{4.19}$$

with $\mathbf{g}_F : \mathbb{R}^{N_F} \times \mathbb{R}^{N_\theta} \to \mathbb{R}^{N_F}$ representing the algebraic simulator equations. Here, $\mathbf{x} \in \mathbb{R}^{N_F}$ is the state vector representing the solution of our model, which typically is the pressure solution, and $\boldsymbol{\theta} \in \mathbb{R}^{N_{\boldsymbol{\theta}}}$ is the vector of parameters. In the next section the subscript F represents 'fine scale'. There are N_F fine scale cells and a total of $N_{\boldsymbol{\theta}}$ parameters. Implicitly, it is assumed that the state is dependent on the parameters, such that $\mathbf{x} = \mathbf{x}(\boldsymbol{\theta})$ and that we can write $\mathbf{g}_F$ as

$$\mathbf{g}_F = \mathbf{A}(\boldsymbol{\theta})\mathbf{x} - \mathbf{q}(\boldsymbol{\theta}) \tag{4.20}$$

Where $\mathbf{A}$ is a matrix of dimensions $N_F \times N_F$ and $\mathbf{q}$ a vector of length N_F. Next to model states, we can compute the observable responses generated from the forward simulation. These responses do not only depend on the model state, but also on the parameters such that

$$\mathbf{y}_F = \mathbf{h}_F(\mathbf{x}, \boldsymbol{\theta}), \tag{4.21}$$

where $\mathbf{h}_F : \mathbb{R}^{N_F} \times \mathbb{R}^{N_{\boldsymbol{\theta}}} \to \mathbb{R}^{N_y}$ represent the output equations.

We aim to compute a multiscale solution. Assume that there are N_C coarse-grid nodes ($N_C << N_F$), then for each coarse-grid node we have one basis function φ_i . The prolongation matrix is the matrix constructed from the basis functions

$$\mathbf{P} = \mathbf{P}(\boldsymbol{\theta}) = \begin{bmatrix} \varphi_1(\boldsymbol{\theta}) & | & \varphi_2(\boldsymbol{\theta}) & | & \cdots & | & \varphi_{N_C}(\boldsymbol{\theta}) \end{bmatrix}, \tag{4.22}$$

Where $\mathbf{P}$ is of dimensions $N_F \times N_C$ and is used to interpolate the coarse scale solution to the fine scale solution. Restriction operator $\mathbf{R}$, with dimensions $N_C \times N_F$, maps the fine scale solution to a coarse scale resolution. For simplicity, we assume that $\mathbf{R}$ is independent of $\boldsymbol{\theta}$. Let $\check{\mathbf{x}} \in \mathbb{R}^{N_C}$ be a solution on the coarse-grid. Then, $\check{\mathbf{x}}$ is obtained from solving

$$\check{\mathbf{g}} = (\mathbf{RAP})\check{\mathbf{x}} - (\mathbf{Rq}) = \mathbf{0} \tag{4.23}$$

Finally, we can interpolate the coarse grid solution to the finer scale, giving the solution $\mathbf{x}'$

$$\mathbf{x}' = \mathbf{P}\check{\mathbf{x}} \tag{4.24}$$

5

Iterative Multiscale Methods

Even though multiple examples have been able to demonstrate the power and accuracy of the Multiscale Methods, errors are still generated by the multiscale method. The error mainly originates from the localisation assumptions to solve the local fine scale problems [12, 13]. However, for models with large cohesive structures with high permeability contrasts it is difficult to find an accurate localization assumption, thus leading to large errors. Typical examples are models with shale layers, fracture or difficult geometries. For this reason, the iterative Multiscale Finite Volume Method (i-MSFV) has been developed. In this chapter we we will explain the framework of the Iterative Multiscale Method.

In each iteration the i-MSFV method gives a solution corresponding to fine scale results. These solutions are achieved by an iterative improvement of the high-frequency errors generated in the multiscale procedure. To remove these errors, we use a smoothing step as in [30]. Corrections are made each iteration until the desired tolerance is reached. In this section we will directly derive this method upon the forward model equations as defined in section [4.4]. In this section we will follow the works of [30].

The iterative multiscale methods are residual based. Solving a system of equation with aid of numerical solvers generate errors [41] and therefore lead to residuals. Let $\mathbf{x}^{\nu-1}$ be an approximate solution to equation (4.20) and denote $\mathbf{r} \in \mathbb{R}^{N_F}$ as the corresponding residual. At iteration level $\nu - 1$ it is defined as

$$\mathbf{r}^{\nu-1} = \mathbf{q} - \mathbf{A}\mathbf{x}^{\nu-1} \tag{5.1}$$

Note that at when $\nu = 0, \mathbf{r}^0 = \mathbf{q}$. A multiscale improvement can be made, by rewriting equation (4.23) in residual form. If we denote the coarse scale correction at iteration level ν as $\delta\check{\mathbf{x}}^{\nu}$, it can be computed by solving

$$\check{\mathbf{g}}^{\nu}(\delta\check{\mathbf{x}}^{\nu}, \mathbf{x}^{\nu-1}, \boldsymbol{\theta}) = (\mathbf{RAP})\delta\check{\mathbf{x}}^{\nu} - \check{\mathbf{r}}^{\nu-1} = \mathbf{0} \tag{5.2}$$

The coarse scale residual $\check{\mathbf{r}} \in \mathbb{R}^{N_C}$ may be written as $\mathbf{R}\mathbf{r}^{\nu}$, hence applying the definition of the residual as in equation (5.1), we find

$$\check{\mathbf{g}}^{\nu}(\delta\check{\mathbf{x}}^{\nu}, \mathbf{x}^{\nu-1}, \boldsymbol{\theta}) = (\mathbf{RAP})\delta\check{\mathbf{x}}^{\nu} - \mathbf{R}\left(\mathbf{q} - \mathbf{A}\mathbf{x}^{\nu-1}\right) = \mathbf{0} \tag{5.3}$$

Consecutively, we define the fine scale correction as $\delta\mathbf{x}'^{\nu}$. We can calculate this correction by redefining equation (4.24) to

$$\mathbf{g}'^{\nu}(\delta\check{\mathbf{x}}^{\nu}, \delta\mathbf{x}'^{\nu}, \boldsymbol{\theta}) = \delta\mathbf{x}'^{\nu} - \mathbf{P}\delta\check{\mathbf{x}}^{\nu} = \mathbf{0} \tag{5.4}$$

After this, there is a smoothing stage. For now, we assume that the smoothing stage operates as a black-box as opposed to previous work. Denote the smoothing correction as $\delta\mathbf{x}_{\sigma}^{\nu}$. The smoothed solution is calculated from

$$\mathbf{g}_{\sigma}^{\nu}(\delta\mathbf{x}'^{\nu}, \delta\mathbf{x}_{\sigma}^{\nu}, \mathbf{x}^{\nu-1}, \boldsymbol{\theta}) = \mathbf{A}\delta\mathbf{x}_{\sigma}^{\nu} - \mathbf{r}_{\sigma}^{\nu-1} = \mathbf{0}, \tag{5.5}$$

where $\mathbf{r}_{\sigma} \in \mathbb{R}^{N_F}$ is the residual at previous iteration after the smoothing stage. That is the fine scale residual at previous iteration minus the correction made at the current iteration, i.e.

$$\mathbf{r}_{\sigma}^{\nu} = \mathbf{r}^{\nu-1} - \mathbf{A}\delta\mathbf{x}'^{\nu} = \mathbf{q} - \mathbf{A}\mathbf{x}^{\nu-1} - \mathbf{A}\delta\mathbf{x}'^{\nu} \tag{5.6}$$

Plugging in the expression for the smoothed residual gives us

$$\mathbf{g}_\sigma^\nu(\boldsymbol{\delta}\mathbf{x}'^\nu, \boldsymbol{\delta}\mathbf{x}_\sigma^\nu, \mathbf{x}^{\nu-1}, \boldsymbol{\theta}) = \mathbf{A}\boldsymbol{\delta}\mathbf{x}_\sigma^\nu - \mathbf{q} + \mathbf{A}\mathbf{x}^{\nu-1} + \mathbf{A}(\boldsymbol{\delta}\mathbf{x}'^\nu) = \mathbf{0} \tag{5.7}$$

Finally, the updated estimation for $\mathbf{x}^\nu$ is given by

$$\mathbf{g}_\mathbf{x}^\nu(\mathbf{x}^\nu, \mathbf{x}^{\nu-1}, \boldsymbol{\delta}\mathbf{x}'^\nu, \boldsymbol{\delta}\mathbf{x}_\sigma^\nu, \boldsymbol{\theta}) = \mathbf{x}^{\nu-1} + \boldsymbol{\delta}\mathbf{x}'^\nu + \delta\mathbf{x}_\sigma^\nu = \mathbf{0} \tag{5.8}$$

We note that $\mathbf{x}^{\nu-1}$ in this iterative fashion may be written as

$$\mathbf{x}^{\nu-1} = \mathbf{x}^{\nu-2} + \boldsymbol{\delta}\mathbf{x}'^{\nu-1} + \boldsymbol{\delta}\mathbf{x}_\sigma^{\nu-1} = \cdots = \mathbf{x}^0 + \sum_{\tau=1}^{\nu-1}(\boldsymbol{\delta}\mathbf{x}'^\tau + \boldsymbol{\delta}\mathbf{x}_\sigma^\tau) \tag{5.9}$$

This results allows us to rewrite equation (5.3) to find

$$\breve{\mathbf{g}}^\nu(\boldsymbol{\delta}\breve{\mathbf{x}}^\nu, \boldsymbol{\delta}\mathbf{x}'^{\nu-1}, \boldsymbol{\delta}\mathbf{x}_\sigma^{\nu-1}, \boldsymbol{\theta}) = \mathbf{R}\mathbf{A}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_\sigma)^{\nu-1} + (\mathbf{R}\mathbf{A}\mathbf{P})\boldsymbol{\delta}\breve{\mathbf{x}}^\nu - \mathbf{R}\left(\mathbf{q} - \mathbf{A}\sum_{\tau=1}^{\nu-2}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_\sigma)^\tau\right) = \mathbf{0}, \tag{5.10}$$

and equation (5.5) to find

$$\mathbf{g}_\sigma^\nu(\boldsymbol{\delta}\mathbf{x}'^\nu, \boldsymbol{\delta}\mathbf{x}_\sigma^\nu, \boldsymbol{\delta}\mathbf{x}'^{\nu-1}, \boldsymbol{\delta}\mathbf{x}_\sigma^{\nu-1}, \boldsymbol{\theta}) = \mathbf{A}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_\sigma)^\nu - \mathbf{q} + \mathbf{A}\left(\sum_{\tau=1}^{\nu-1}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_\sigma)^\tau\right) = \mathbf{0} \tag{5.11}$$

Then, equations (5.8),(5.10), (5.4) and (5.11) give rise to a system of equations that needs to be solved for every iteration level ν.

$$\mathbf{A}\mathbf{x}^\nu = \mathbf{q}^\nu, \tag{5.12}$$

where

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{RA} & \mathbf{RA} & \mathbf{0} & \mathbf{RAP} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{P} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{A} & \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & -\mathbf{I} & -\mathbf{I} & \mathbf{I} \end{pmatrix}, \quad \mathbf{q}^\nu = \begin{pmatrix} \mathbf{R}\left(\mathbf{q} - \mathbf{A}\sum_{\tau=1}^{\nu-2}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_\sigma)^\tau\right) \\ \mathbf{0} \\ \mathbf{q} - \mathbf{A}\sum_{\tau=1}^{\nu-2}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_\sigma)^\tau) \\ \mathbf{0} \end{pmatrix} \tag{5.13}$$

This iterative process is stopped when the residual has achieved the requested accuracy ϵ. Hence the number of iterations ν is controlled by

$$\text{If } \frac{||\mathbf{r}^\nu||}{||\mathbf{q}||} > \epsilon, \text{ continue, set } \nu = \nu + 1, \text{ else stop.} \tag{5.14}$$

In the same way, the smoothing stage is stopped when the smoothed residual has achieved the requested accuracy ϵ_σ

$$\text{If } \frac{||\mathbf{r}_\sigma^{\nu-1}||}{||\mathbf{r}^\nu||} > \epsilon_\sigma, \text{ repeat smoothing step, else stop.} \tag{5.15}$$

The algorithm can be summarized into algorithm [1] as in [26]. After convergence we write the solution in a different notation. The super-vector notation [19, 32] offers a convenient way of writing all solution into a compact form. This form will prove its benefits in chapter [7]. In this notation we redefine $\mathbf{x}$ and source terms $\mathbf{q}$ as

$$\mathbf{x}(\boldsymbol{\theta}) = \begin{pmatrix} \mathbf{x}^0 \\ \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^{\nu-1} \\ \mathbf{x}^\nu \end{pmatrix}, \text{ where } \mathbf{x}^i = \begin{pmatrix} \boldsymbol{\delta}\breve{\mathbf{x}}^i \\ \boldsymbol{\delta}\mathbf{x}' \\ \boldsymbol{\delta}\mathbf{x}_\sigma^i \\ \mathbf{x}^i \end{pmatrix} \quad \text{and} \quad \mathbf{q}(\boldsymbol{\theta}) = \begin{pmatrix} \mathbf{q}^0 \\ \mathbf{q}^1 \\ \vdots \\ \mathbf{q}^{\nu-1} \\ \mathbf{q}^\nu \end{pmatrix} \tag{5.16}$$

Likewise, we rewrite the forward model equations as a super vector $\mathbf{g}$,

$$\mathbf{g}(x,\boldsymbol{\theta}) = \begin{pmatrix} \mathbf{g}^0(x,\boldsymbol{\theta}) \\ \mathbf{g}^1(x,\boldsymbol{\theta}) \\ \vdots \\ \mathbf{g}^{\nu-1}(x,\boldsymbol{\theta}) \\ \mathbf{g}^{\nu}(x,\boldsymbol{\theta}) \end{pmatrix} = \mathbf{0}, \text{ where } \quad \mathbf{g}^i = \begin{pmatrix} \check{\mathbf{g}}(\boldsymbol{\delta}\check{\mathbf{x}}^i, \boldsymbol{\delta}\mathbf{x}'^{i-1}, \boldsymbol{\delta}\mathbf{x}_\sigma^{i-1}, \boldsymbol{\theta}) \\ \mathbf{g}'(\boldsymbol{\delta}\check{\mathbf{x}}^i, \boldsymbol{\delta}\mathbf{x}'^{i}, \boldsymbol{\theta}) \\ \mathbf{g}_\sigma(\boldsymbol{\delta}\mathbf{x}'^{i}, \boldsymbol{\delta}\mathbf{x}_\sigma^{i}, \boldsymbol{\delta}\mathbf{x}'^{i-1}, \boldsymbol{\delta}\mathbf{x}_\sigma^{i-1}, \boldsymbol{\theta}) \\ \mathbf{g}_x(\boldsymbol{\delta}\mathbf{x}'^{i}, \boldsymbol{\delta}\mathbf{x}_\sigma^{i}, \mathbf{x}^{i-1}, \mathbf{x}^{i}, \boldsymbol{\theta}) \end{pmatrix} = \mathbf{0} \tag{5.17}$$

6

Derivation of the Multiscale Lagrange Multiplier Gradient Computation Framework

6.1. Introduction

As seen in chapter [3] the gradient of an objective function plays an important role, but we have also seen that the computational effort to compute the gradient can become high. In this section we will recast the multiscale gradient computation introduced by [25] in a Lagrange Multiplier setting to obtain a mathematical formulation of the gradient. First of all, we will explain how we can calculate gradient information by applying the method of Lagrange Multipliers as previously discussed in chapter [3]. For this we distinguish two distinct categories of objective functions. Before the multiscale derivation of the gradient computation is given, we first discuss the derivation of the computation of gradient information on a single, fine scale grid. Finally, we will consider the multiscale framework where we use the forward multiscale equations as presented in section [4.4].

6.2. Sensitivity matrix and Lagrange Multipliers

The sensitivity matrix is defined as the total derivative of the response function $\mathbf{h}(\mathbf{x}, \boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ [25]. It is denoted as

$$\mathbf{G} = \frac{d\mathbf{h}}{d\boldsymbol{\theta}}. \tag{6.1}$$

The sensitivity matrix shows us how much our responses will change if we perturb the parameters.

To find a formulation for the sensitivity matrix we use the Lagrange Multiplier method. Here we use a general objective function $O(\mathbf{x}, \boldsymbol{\theta})$. Now we distinguish two different categories of objective functions; scalar- and vectorial objective functions.

Now, the only difference between the scalar- and vectorial objective functions is of the size of the Lagrangian and the Lagrange multipliers, and thus the notation of these quantities. The method however, is valid for both types [33]. One may conclude that the strategy of augmenting the objective function by Lagrange Multipliers and applying the first-order necessary conditions results in comparable results. The exact dimensions are denoted in table [6.1].

For illustration, we explain the Lagrange Multiplier method using a scalar objective function. In both categories, we augment the objective function with our model equations multiplied by the Lagrange multipliers. We note that in this sense, the constraint equations as in chapter [3] have a different interpretation. They do not represent a bound on the pressure or injection rates. However, the same theory still applies as we want to optimize the value of an objective function where the solution is

Different notations and dimensions				
	Lagrange Multipliers	dimensions	Lagrangian	dimensions
Scalar OF	$\boldsymbol{\lambda}$	$\mathbb{R}^{N_F}$	L	$\mathbb{R}$
Vector OF	$\boldsymbol{\Lambda}$	$\mathbb{R}^{N_F} \times \mathbb{R}^{N_y}$	$\mathbf{L}$	$\mathbb{R}^{N_y}$

Table 6.1: Dimensions of Lagrangian and Multipliers

'constrained' by the forward model equations. Then, we find the Lagrangian

$$L(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = O(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\lambda}^{\mathsf{T}} \mathbf{g} \tag{6.2}$$

To find a maximum or minimum we apply the first-order necessary conditions to find the system of equations

$$\nabla_{\boldsymbol{\lambda}} L = \mathbf{0} = \mathbf{g} \tag{6.3}$$

$$\nabla_{\mathbf{x}} L = \mathbf{0} = \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} + \boldsymbol{\lambda}^{\mathsf{T}} \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \tag{6.4}$$

$$\nabla_{\boldsymbol{\theta}} L = \mathbf{0} = \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda}^{\mathsf{T}} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right) \tag{6.5}$$

Now, we note that because we have that $\mathbf{g} = 0$,

$$\nabla_{\boldsymbol{\theta}} L = \nabla_{\boldsymbol{\theta}} \left(O(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\lambda}^{\mathsf{T}} \mathbf{g} \right) = \nabla_{\boldsymbol{\theta}} O(\mathbf{x}, \boldsymbol{\theta}), \tag{6.6}$$

hence if we take $O(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{h}$, then we see that this method enables us to compute $\nabla_{\boldsymbol{\theta}} \mathbf{h} = \mathbf{G}$.

6.3. Gradient formulation in a single fine scale model

In this section we will derive a mathematical formulation of the sensitivity matrix on a fine scale system.

6.3.1. Considering a scalar objective function

For the scalar objective function we assume a function $O(\mathbf{x}, \boldsymbol{\theta}) : \mathbb{R}^{N_F} \times \mathbb{R}^{N_{\boldsymbol{\theta}}} \to \mathbb{R}$ to be minimized on the fine scale model. Our Lagrangian can be written as

$$L(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = O(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\lambda}^{\mathsf{T}} \mathbf{g}, \quad L : \mathbb{R}^{N_F} \times \mathbb{R}^{N_{\boldsymbol{\theta}}} \times \mathbb{R}^{N_F} \to \mathbb{R} \tag{6.7}$$

and to find a maximum or minimum we apply the first-order necessary conditions. From (6.4) we clearly see that $\nabla_{\mathbf{x}} L$ vanishes if

$$\boldsymbol{\lambda}^{\mathsf{T}} \frac{\partial \mathbf{g}}{\partial \mathbf{x}} = -\frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \Rightarrow \boldsymbol{\lambda}^{\mathsf{T}} = \left(-\frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \tag{6.8}$$

The values for the Lagrange Multipliers can be substituted into the condition (6.5), giving

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \left(\frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right) \quad \Rightarrow \tag{6.9}$$

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} - \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \left(\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right) \tag{6.10}$$

The latter expression may be simplified to

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial O(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \left(\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right) \tag{6.11}$$

From section [4.4] we have that $\mathbf{g} = \mathbf{A}(\boldsymbol{\theta})\mathbf{x} - \mathbf{q}(\boldsymbol{\theta})$. This gives,

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \mathbf{A}(\boldsymbol{\theta}), \quad \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{x} + \mathbf{A} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \tag{6.12}$$

where we apply the chain-rule. Note that the derivative of $\mathbf{A}$ with respect to $\boldsymbol{\theta}$ results in a third-order tensor. An interpretation of this can be found in [25]. Substitution of the formulations of (6.12) into (6.11) gives us the final expression

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O(\mathbf{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial O(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}} (\mathbf{A})^{-1} \left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{x} + \mathbf{A} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \right), \tag{6.13}$$

or in other words, if we take $O = \mathbf{h}$, which represents a single model response in this case,

$$\mathbf{G} = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{h}}{\partial \mathbf{x}} (\mathbf{A})^{-1} \left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{x} + \mathbf{A} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \right) \tag{6.14}$$

6.3.2. Considering an vectorial objective function

In contrast to the previous section, we employ the same single scale strategy, but we consider a vectorial objective function. This objective function is assumed to be written as $\mathbf{O}(\mathbf{x},\boldsymbol{\theta}) : \mathbb{R}^{N_F} \times \mathbb{R}^{N_{\boldsymbol{\theta}}} \rightarrow \mathbb{R}^{N_y}$, where $\mathbf{O} = (O_1, \cdots, O_y)^\top$. Next, we introduce the augmented objective function as

$$\mathbf{L}(\mathbf{x},\boldsymbol{\theta},\boldsymbol{\Lambda}) = \mathbf{O}(\mathbf{x},\boldsymbol{\theta}) + \boldsymbol{\Lambda}^\top \mathbf{g}, \tag{6.15}$$

where for the first time we notice the different dimensions of $\mathbf{L}, \boldsymbol{\Lambda}$. The first-order necessary equations can be written as

$$\nabla_{\boldsymbol{\Lambda}} \mathbf{L} = \mathbf{0} = \mathbf{g} \tag{6.16}$$

$$\nabla_{\mathbf{x}} \mathbf{L} = \mathbf{0} = \frac{\partial O(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}} + \boldsymbol{\Lambda}^\top \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \tag{6.17}$$

$$\nabla_{\boldsymbol{\theta}} \mathbf{L} = \mathbf{0} = \frac{\partial O(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O(\mathbf{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \boldsymbol{\Lambda}^\top \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right) \tag{6.18}$$

Clearly, $\nabla_{\mathbf{x}} \mathbf{L}$ vanishes if

$$\boldsymbol{\Lambda}^\top = -\frac{\partial \mathbf{O}(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \tag{6.19}$$

Then, one can easily conclude that

$$(\boldsymbol{\lambda}_i)^\top = \frac{\partial \mathbf{O}_i(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}} \mathbf{A}^{-1}, \quad \forall i \in \{1, \cdots, y\} \tag{6.20}$$

Before we substitute the values for our Lagrange Multipliers into condition (6.18), we notice that the structure of the Lagrange Multipliers is similar to the Lagrange Multipliers in (6.8), and the subscript i directly relates to $i-th$ component of $\mathbf{L}$. Hence, we may state that

$$\nabla_{\boldsymbol{\theta}} \mathbf{L}_i = \frac{\partial \mathbf{O}_i(\mathbf{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{O}_i(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}} (\mathbf{A})^{-1} \left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{x} + \mathbf{A} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \right), \quad \forall i \in \{1, \cdots, y\} \tag{6.21}$$

which can be rewritten into vector notation as

$$\nabla_{\boldsymbol{\theta}} \mathbf{L} = \frac{\partial \mathbf{O}(\mathbf{x},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{O}(\mathbf{x},\boldsymbol{\theta})}{\partial \mathbf{x}} (\mathbf{A})^{-1} \left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{x} + \mathbf{A} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \right) \tag{6.22}$$

But then, if we take $O = \mathbf{h}$, which represent multiple model responses.

$$\mathbf{G} = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{h}}{\partial \mathbf{x}} (\mathbf{A})^{-1} \left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{x} + \mathbf{A} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \right) \tag{6.23}$$

6.4. Gradient formulation in a multiscale framework

As opposed to the previous section where calculated the sensitivity matrix on a fine scale only, in this section we will derive a different formulation using the computational benefits of the Multiscale Framework. First we will consider an scalar objective function. Consecutively we will consider a multi-valued objective function.

6.4.1. Considering a scalar objective function

In order to compute the multiscale derivatives with respect to the parameters, we will follow the same line of reasoning as in the previous section. However, in the multiscale method the gradient with respect to the parameters is build from the coarse scale solution instead of the computationally heavy fine scale approach. A key aspect in this method is using the framework in section [4.4] to define

$$\mathbf{x} = \begin{bmatrix} \check{\mathbf{x}} \\ \mathbf{x}' \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \check{\mathbf{g}} \\ \mathbf{g}' \end{bmatrix} = \begin{bmatrix} (\mathbf{RAP})\check{\mathbf{x}} - \mathbf{Rq} \\ \mathbf{x}' - \mathbf{P}\check{\mathbf{x}} \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} \check{\boldsymbol{\lambda}} \\ \boldsymbol{\lambda}' \end{bmatrix} \tag{6.24}$$

In this case we can describe the state of our model not only in fine scale, but also on the coarse scale. Note that in this case $\mathbf{x}, \mathbf{g}, \boldsymbol{\lambda} \in \mathbb{R}^{N_c N_F}$. Once more we consider an objective function of the form $O(\mathbf{x}, \boldsymbol{\theta}) : \mathbb{R}^{N_c N_F} \times \mathbb{R}^{N_{\boldsymbol{\theta}}} \to \mathbb{R}$, and similarly we write our Lagrangian as

$$L(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = O(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\lambda}^\top \mathbf{g}, \quad L : \mathbb{R}^{N_F} \times \mathbb{R}^{N_{\boldsymbol{\theta}}} \times \mathbb{R}^{N_F} \to \mathbb{R}, \tag{6.25}$$

In accordance to the fine scale, we are able to calculate the coefficients $\boldsymbol{\lambda}$ by forcing $\nabla_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbf{0}^\top$. For simplicity, we do not write the dependency of O on $\mathbf{x}$ and $\boldsymbol{\theta}$. By doing this, we obtain the following system.

$$\begin{aligned} \frac{\partial O}{\partial \mathbf{x}} + \boldsymbol{\lambda}^\top \frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \mathbf{0}^\top &\Leftrightarrow \begin{pmatrix} \frac{\partial O}{\partial \check{\mathbf{x}}} & \frac{\partial O}{\partial \mathbf{x}'} \end{pmatrix} + \begin{pmatrix} (\check{\boldsymbol{\lambda}})^\top & (\boldsymbol{\lambda}')^\top \end{pmatrix} \begin{pmatrix} \frac{\partial \check{\mathbf{g}}}{\partial \check{\mathbf{x}}} & \frac{\partial \check{\mathbf{g}}}{\partial \mathbf{x}'} \\ \frac{\partial \mathbf{g}'}{\partial \check{\mathbf{x}}} & \frac{\partial \mathbf{g}'}{\partial \mathbf{x}'} \end{pmatrix} = \mathbf{0}^\top \\ &\Leftrightarrow \begin{pmatrix} (\check{\boldsymbol{\lambda}})^\top & (\boldsymbol{\lambda}')^\top \end{pmatrix} \begin{pmatrix} \mathbf{RAP} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{pmatrix} = -\begin{pmatrix} \frac{\partial O}{\partial \check{\mathbf{x}}} & \frac{\partial O}{\partial \mathbf{x}'} \end{pmatrix} \\ &\Leftrightarrow \begin{pmatrix} (\check{\boldsymbol{\lambda}})^\top & (\boldsymbol{\lambda}')^\top \end{pmatrix} = \begin{pmatrix} -\frac{\partial O}{\partial \check{\mathbf{x}}} & -\frac{\partial O}{\partial \mathbf{x}'} \end{pmatrix} \begin{pmatrix} (\mathbf{RAP})^{-1} & \mathbf{0} \\ \mathbf{P}(\mathbf{RAP})^{-1} & \mathbf{I} \end{pmatrix} \end{aligned} \tag{6.26}$$

Then, we find an expression for our Lagrange multipliers on coarse- and fine scale as

$$(\check{\boldsymbol{\lambda}})^\top = -\left(\frac{\partial O}{\partial \check{\mathbf{x}}} + \frac{\partial O}{\partial \mathbf{x}'} \mathbf{P} \right) (\mathbf{RAP})^{-1} \tag{6.27}$$

$$(\boldsymbol{\lambda}')^\top = -\frac{\partial O}{\partial \mathbf{x}'} \tag{6.28}$$

In the next step we can plug these values into the third first-order necessary conditions (6.5),

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O}{\partial \boldsymbol{\theta}} + \begin{pmatrix} (\check{\boldsymbol{\lambda}})^\top & (\boldsymbol{\lambda}')^\top \end{pmatrix} \underbrace{\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right)}_{(*)} \tag{6.29}$$

Now, we expand and simplify $(*)$, which gives us the following relation

$$\begin{aligned} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right) &= \begin{pmatrix} \mathbf{RAP} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathbf{x}'}{\partial \boldsymbol{\theta}} \end{pmatrix} + \begin{pmatrix} \frac{\partial \check{\mathbf{g}}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathbf{g}'}{\partial \boldsymbol{\theta}} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{RAP} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathbf{x}'}{\partial \boldsymbol{\theta}} \end{pmatrix} + \begin{pmatrix} \left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}} (\mathbf{AP}) + \mathbf{R} \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{P} + (\mathbf{RA}) \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}} \right] \check{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}} \mathbf{q} - \mathbf{R} \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \\ -\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}} \check{\mathbf{x}} \end{pmatrix} \end{aligned} \tag{6.30}$$

The previous equation can be simplified further to find

$$\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \right) = \begin{bmatrix} (\mathbf{RAP}) \frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} + \left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}} (\mathbf{AP}) + \mathbf{R} \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}} \mathbf{P} + (\mathbf{RA}) \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}} \right] \check{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}} \mathbf{q} - \mathbf{R} \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} \\ -\mathbf{P} \frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{x}'}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}} \check{\mathbf{x}} \end{bmatrix} \tag{6.31}$$

Plugging equation (6.31) into equation (6.29), multiplying with the Lagrange multiplies (6.27) and (6.28), then leaves us with the expression

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O}{\partial \boldsymbol{\theta}} - \left(\frac{\partial O}{\partial \check{\mathbf{x}}} + \frac{\partial O}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\left((\mathbf{RAP})\frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} + \left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}(\mathbf{AP}) + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\check{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}\mathbf{q} - \mathbf{R}\frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}\right) - \frac{\partial O}{\partial \mathbf{x}'}\left(-\mathbf{P}\frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{x}'}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}}\right) \tag{6.32}$$

If we work out the product and collect all terms, we are left with

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O}{\partial \boldsymbol{\theta}} - \left(\frac{\partial O}{\partial \check{\mathbf{x}}} + \frac{\partial O}{\partial \mathbf{x}'}\mathbf{P}\right)\frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} + \frac{\partial O}{\partial \mathbf{x}'}\left(\mathbf{P}\frac{\partial \check{\mathbf{x}}}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{x}'}{\partial \boldsymbol{\theta}}\right) - \left(\frac{\partial O}{\partial \check{\mathbf{x}}} + \frac{\partial O}{\partial \mathbf{x}'}\mathbf{P}\right)\cdot (\mathbf{RAP})^{-1}\left(\left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}(\mathbf{AP}) + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\check{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}\mathbf{q} - \mathbf{R}\frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}\right) + \frac{\partial O}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}} \tag{6.33}$$

The reader can see, that many of the terms cancel out on one another. Therefore, there is one more simplification step required to arrive at the following expression

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O}{\partial \boldsymbol{\theta}} + \frac{\partial O}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}} - \left(\frac{\partial O}{\partial \check{\mathbf{x}}} + \frac{\partial O}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\left(\left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}(\mathbf{AP}) + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\check{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}\mathbf{q} - \mathbf{R}\frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}\right) \tag{6.34}$$

For simplicity we disregard the dependency of $\mathbf{R}, \mathbf{q}$ on $\boldsymbol{\theta}$, hence the final expression is given by

$$\nabla_{\boldsymbol{\theta}} L = \frac{\partial O}{\partial \boldsymbol{\theta}} + \frac{\partial O}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}} - \left(\frac{\partial O}{\partial \check{\mathbf{x}}} + \frac{\partial O}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\check{\mathbf{x}}, \tag{6.35}$$

and thus, if we take $O = \mathbf{h}$,

$$\mathbf{G} = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}} - \left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\check{\mathbf{x}} \tag{6.36}$$

6.4.2. Considering a vectorial objective function

Finally, we employ the same multi-scale strategy as in the previous example, however now we use the second category of the objective functions. For this, we suppose that our objective function is of the form $\mathbf{O}(\mathbf{x}, \boldsymbol{\theta}), \mathbf{O} : \mathbb{R}^{N_c N_F} \times \mathbb{R}^{N_\theta} \to \mathbb{R}^{N_y}$. Thus, $\mathbf{O} = (O_1, \cdots, O_y)^\top$. O, however with different dimensions. If we define $\mathbf{x}, \mathbf{g}, \boldsymbol{\lambda}$ in the same manner and introduce our Lagrangian as

$$L(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbf{O} + \boldsymbol{\lambda}^\top \mathbf{g} \tag{6.37}$$

we easily note that there is a mismatch in the dimensions, since $\mathbf{O}$ is a vector, while $\boldsymbol{\lambda}^\top \mathbf{g}$ is a scalar. To resolve this problem we change the definition of $\boldsymbol{\lambda}$. We define

$$\boldsymbol{\Lambda} = \begin{pmatrix} \check{\boldsymbol{\lambda}}_1 & \check{\boldsymbol{\lambda}}_2 & \cdots & \check{\boldsymbol{\lambda}}_y \\ \boldsymbol{\lambda}'_1 & \boldsymbol{\lambda}'_2 & \cdots & \boldsymbol{\lambda}'_y \end{pmatrix} \tag{6.38}$$

as the $N_c N_F \times N_y$ matrix of Lagrange Multipliers where $\check{\boldsymbol{\lambda}}_i \in \mathbb{R}^{N_c}$ and $\boldsymbol{\lambda}'_i \in \mathbb{R}^{N_F}, \forall i \in \{1, \cdots, y\}$. Now we redefine the Lagrangian as

$$\mathbf{L}(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\Lambda}) = \mathbf{O} + \boldsymbol{\Lambda}^\top \mathbf{g} \tag{6.39}$$

To calculate the values of $\boldsymbol{\Lambda}$ we force $\nabla_{\mathbf{x}} \mathbf{L}(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\Lambda}) = \mathbf{0}^\top$ which leads to the following relation

$$\begin{aligned} \frac{\partial \mathbf{O}}{\partial \mathbf{x}} + \boldsymbol{\Lambda}^\top \frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \mathbf{0}^\top &\Leftrightarrow \boldsymbol{\Lambda}^\top \frac{\partial \mathbf{g}}{\partial \mathbf{x}} = -\frac{\partial \mathbf{O}}{\partial \mathbf{x}} \\ &\Leftrightarrow \boldsymbol{\Lambda}^\top = -\frac{\partial \mathbf{O}}{\partial \mathbf{x}}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)^{-1} \end{aligned} \tag{6.40}$$

We note that $\frac{\partial \mathbf{O}}{\partial \mathbf{x}}$ is a $N_y \times N_c N_F$ matrix. We will write

$$\frac{\partial \mathbf{O}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial O_1}{\partial \breve{\mathbf{x}}} & \frac{\partial O_1}{\partial \mathbf{x}'} \\ \vdots & \vdots \\ \frac{\partial O_y}{\partial \breve{\mathbf{x}}} & \frac{\partial O_y}{\partial \mathbf{x}'} \end{pmatrix}, \quad \frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \begin{pmatrix} \mathbf{RAP} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{pmatrix} \tag{6.41}$$

simplifying [6.40] to

$$\begin{pmatrix} (\breve{\boldsymbol{\lambda}}_1)^\top & (\boldsymbol{\lambda}'_1)^\top \\ \vdots & \vdots \\ (\breve{\boldsymbol{\lambda}}_y)^\top & (\boldsymbol{\lambda}'_y)^\top \end{pmatrix} = \begin{pmatrix} -\frac{\partial O_1}{\partial \breve{\mathbf{x}}} & -\frac{\partial O_1}{\partial \mathbf{x}'} \\ \vdots & \vdots \\ -\frac{\partial O_y}{\partial \breve{\mathbf{x}}} & -\frac{\partial O_y}{\partial \mathbf{x}'} \end{pmatrix} \begin{pmatrix} (\mathbf{RAP})^{-1} & \mathbf{0} \\ \mathbf{P}(\mathbf{RAP})^{-1} & \mathbf{I} \end{pmatrix} \tag{6.42}$$

$$= \begin{pmatrix} -\left(\frac{\partial O_1}{\partial \breve{\mathbf{x}}} + \frac{\partial O_1}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1} & -\frac{\partial O_1}{\partial \mathbf{x}'} \\ \vdots & \vdots \\ -\left(\frac{\partial O_y}{\partial \breve{\mathbf{x}}} + \frac{\partial O_y}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1} & -\frac{\partial O_y}{\partial \mathbf{x}'} \end{pmatrix} \tag{6.43}$$

Hence we find an expression for our Lagrange multipliers

$$\begin{aligned} (\breve{\boldsymbol{\lambda}}_i)^\top &= -\left(\frac{\partial O_i}{\partial \breve{\mathbf{x}}} + \frac{\partial O_i}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1} \\ (\boldsymbol{\lambda}'_i)^\top &= -\frac{\partial O_i}{\partial \mathbf{x}'} \qquad\qquad \forall i \in \{1, \cdots, y\} \end{aligned} \tag{6.44}$$

In similar fashion as in section [6.3.2], we note that the Lagrange Multipliers are exactly the same as the fine scale multipliers computed in the previous section. This means that we do not need to redo all the calculations, but that we have that the $i-th$ component of $\mathbf{L}$ is of the same form as (6.35). This leads to

$$\begin{aligned} (\nabla_{\boldsymbol{\theta}} \mathbf{L})_i = \frac{\partial O_i}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} + \frac{\partial O_i}{\partial \boldsymbol{\theta}} - \left(\frac{\partial O_i}{\partial \breve{\mathbf{x}}} + \frac{\partial O_i}{\partial \mathbf{x}'}\mathbf{P}\right)\frac{\partial \breve{\mathbf{x}}}{\partial \boldsymbol{\theta}} - \left(\frac{\partial O_i}{\partial \breve{\mathbf{x}}} + \frac{\partial O_i}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\left(\left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}(\mathbf{AP}) + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\breve{\mathbf{x}}\right. \\ \left. -\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}\mathbf{q}\right) - \frac{\partial O_i}{\partial \mathbf{x}'}\left(-\mathbf{P}\frac{\partial \breve{\mathbf{x}}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{x}'}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\breve{\mathbf{x}}\right) \end{aligned} \tag{6.45}$$

which can be simplified to find the following expressions for the gradient of $\mathbf{L}$ with respect to the parameters

$$\nabla_{\boldsymbol{\theta}} \mathbf{L} = \begin{pmatrix} \frac{\partial O_1}{\partial \boldsymbol{\theta}} - \left(\frac{\partial O_1}{\partial \breve{\mathbf{x}}} + \frac{\partial O_1}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\left(\left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}(\mathbf{AP}) + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\breve{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}\mathbf{q}\right) + \frac{\partial O_1}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\breve{\mathbf{x}} \\ \vdots \\ \frac{\partial O_y}{\partial \boldsymbol{\theta}} - \left(\frac{\partial O_y}{\partial \breve{\mathbf{x}}} + \frac{\partial O_y}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\left(\left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}(\mathbf{AP}) + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\breve{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}\mathbf{q}\right) + \frac{\partial O_y}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\breve{\mathbf{x}} \end{pmatrix} \tag{6.46}$$

or in matrix-vector notation as

$$\nabla_{\boldsymbol{\theta}} \mathbf{L} = \frac{\partial \mathbf{O}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{O}}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\breve{\mathbf{x}} - \left(\frac{\partial \mathbf{O}}{\partial \breve{\mathbf{x}}} + \frac{\partial \mathbf{O}}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\left(\left[\frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}(\mathbf{AP}) + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\breve{\mathbf{x}} - \frac{\partial \mathbf{R}}{\partial \boldsymbol{\theta}}\mathbf{q} - \mathbf{R}\frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}\right) \tag{6.47}$$

For the final step in the derivation, we once again, neglect the dependence of $\mathbf{R}, \mathbf{q}$ on $\boldsymbol{\theta}$. Next to this, we take the function $\mathbf{h}(\mathbf{x}, \boldsymbol{\theta})$, which represent the measurements produced by the model, as our objective function. Then we find an expression for the sensitivity-matrix

$$\mathbf{G} = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\breve{\mathbf{x}} - \left(\frac{\partial \mathbf{h}}{\partial \breve{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\breve{\mathbf{x}} \tag{6.48}$$

The above results give us a theoretical framework for computing the sensitivity matrix $\mathbf{G}$. It also shows us that using the Lagrange Multiplier Method gives the same formulation for the sensitivity matrix $\mathbf{G}$ as with the implicit differentiation method in [25]. The benefit however, is that now we have found an explicit formulation for the Lagrange Multipliers. These multipliers can be important for reducing computational costs as we will discuss in later chapters. The question that remains is, how to efficiently evaluate this matrix.

7

Derivation of the Iterative Multiscale Gradient Computation Framework

7.1. Introduction

In the previous chapter, we have formulated a theoretical framework for finding the sensitivity matrix $\mathbf{G}$. According to different types of studies, different derivative information has to be provided. Among these studies one finds the Quasi-Newton methods, which require the gradient of the objective function as also discussed in chapter [3]. A different example is the category of history matching algorithms that require the sensitivity matrix. Finally, one finds Gauss-Newton methods that require products of $\mathbf{G}$ and it's conjugate $\mathbf{G}^{\mathsf{T}}$. For general applicability of the method one considers the right- and left multiplication of arbitrary matrices $\mathbf{W}, \mathbf{V}$ respectively. By defining algorithms for calculating $\mathbf{WG}$ and $\mathbf{GV}$, different types of derivative information can be accommodated in a single framework [25]. In this chapter we will focus on a framework for the computation of the formulation for the sensitivity matrix. The chapter is split in three different sections. In the first section we will revise the framework for finding algorithms to compute the gradient in a non-iterative setting. The second section will focus on the same objective, but will be formulated in an iterative setting. Finally, we will discuss some similarities and difference between both methods presented.

7.2. Gradient computation in a multiscale setting

The reader is reminded that in chapter [6] one has found a formulation of our sensitivity matrix $\mathbf{G}$ as

$$\mathbf{G} = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}} - \left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\check{\mathbf{x}} \tag{7.1}$$

where the forward model equations are given as in section [4.4].

7.2.1. Direct method

In this section we will employ the direct or forward method [29, 32]. This method starts by considering the calculation of $\mathbf{GV}$, where $\mathbf{V}$ is an arbitrary matrix of dimensions $N_\theta \times n$. Note that if $\mathbf{V} = \mathbf{I}$ in this method, we would directly compute the sensitivity matrix.

We start with the formulation of $\mathbf{G}$ as

$$\mathbf{G} = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}} - \left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\check{\mathbf{x}} \tag{7.2}$$

Now if we define

$$\mathbf{Z} = -\left[(\mathbf{RAP})^{-1}\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\check{\mathbf{x}}\right]\mathbf{V}, \tag{7.3}$$

we can actually compute this matrix $\mathbf{Z}$ by solving the system of equations

$$(\mathbf{RAP})\mathbf{Z} = -\left[\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\check{\mathbf{x}}\right]\mathbf{V} \tag{7.4}$$

Then, we may write the product $\mathbf{GV}$ as

$$\mathbf{GV} = \left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)\mathbf{Z} + \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}}\right)\mathbf{V} + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}\mathbf{V} \tag{7.5}$$

This procedure can be summarized into algorithm [1].

Algorithm 1: Right multiplication of sensitivity matrix, by arbitrary matrix

input : $\mathbf{R}, \mathbf{A}, \mathbf{P}, \mathbf{x}, \mathbf{V}, \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}, \frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}}, \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}, \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}$
output: The product $\mathbf{GV}$

Compute $\alpha = \left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)$ and $\beta = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P}\check{\mathbf{x}}$;
foreach $j = 1, 2, \cdots, n$ **do**
  Compute $\gamma = \left(\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}}\right)\mathbf{V}_{.,j}$;
  Compute $\delta = \mathbf{R}(\beta \mathbf{V}_{.,j} + \mathbf{A}\gamma)$;
  Solve $\mathbf{Z} = (\mathbf{RAP})^{-1}\delta$;
  Compute $(\mathbf{GV})_{.,j} = \alpha\mathbf{Z} - \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\gamma + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}\mathbf{V}_{.,j}$;

7.2.2. Adjoint method

In a similar manner, one may compute the left multiplication of $\mathbf{G}$ with an arbitrary matrix $\mathbf{W}$ of dimensions $m \times N_Y$. This is known as the backward or adjoint method [5]. If we define

$$\mathbf{Z}^\top = -\mathbf{W}\left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)(\mathbf{RAP})^{-1}, \tag{7.6}$$

where we solve for $\mathbf{Z}$ by solving the system

$$(\mathbf{RAP})^\top \mathbf{Z} = -\left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)^\top \mathbf{W}^\top \tag{7.7}$$

Then may write the multiplication as

$$\mathbf{WG} = \mathbf{Z}^\top\mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{A}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right)\check{\mathbf{x}} + \mathbf{W}\left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}}\right) + \mathbf{W}\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} \tag{7.8}$$

Finally we may rewrite this as

$$\mathbf{WG} = \left(\mathbf{Z}^\top\mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P}\check{\mathbf{x}} + \left(\mathbf{Z}^\top\mathbf{RA} + \mathbf{W}\frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\right)\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right) + \mathbf{W}\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} \tag{7.9}$$

This procedure can be summarized into algorithm [2].

Algorithm 2: Left multiplication of sensitivity matrix, by arbitrary matrix

input : $\mathbf{R}, \mathbf{A}, \mathbf{P}, \mathbf{x}, \mathbf{W}, \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\partial\boldsymbol{\theta}, \frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}}, \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}, \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}$
output: The product $\mathbf{WG}$

Compute $\alpha = \left(\frac{\partial \mathbf{h}}{\partial \check{\mathbf{x}}} + \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\mathbf{P}\right)$ and $\beta = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P}\check{\mathbf{x}}$;
foreach $j = 1, 2, \cdots, n$ **do**
  Solve $\mathbf{Z} = (\mathbf{RAP})^{-\top}\alpha\mathbf{W}_{i,\cdot}^\top$;
  Compute $\mathbf{m}^\top = (\mathbf{Z}^\top\mathbf{RA} - \mathbf{W}_{i,\cdot}\frac{\partial \mathbf{h}}{\partial \mathbf{x}'})$;
  Compute $\gamma = \mathbf{m}^\top\left(\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}}\right)$;
  Compute $(\mathbf{WG})_{i,\cdot} = \mathbf{Z}^\top\beta + \gamma + \mathbf{W}_{i,\cdot}\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}$;

7.3. Gradient computation in an iterative multiscale setting

In this section we will derive the gradient information when an iterative multiscale setting is used. Therefore we consider the forward model equations in an iterative setting as in Chapter [5]. We apply the same strategy as in the previous section. Results are based on the unpublished results of [26].

If we repeat the same strategy as in the non-iterative setting, we post- and pre multiply the sensitivity matrix $\mathbf{G}$ by matrices $\mathbf{V}$ of order $N_{\boldsymbol{\theta}} \times n$ and $\mathbf{W}$ of order $m \times N_Y$ respectively we compute

$$\mathbf{WGV} = -\mathbf{W}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)^{-1}\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}\mathbf{V} + \mathbf{W}\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}\mathbf{V} \tag{7.10}$$

7.3.1. Direct method

For the Direct Method we neglect the pre multiplication of matrix $\mathbf{W}$ in equation (7.10). Hence, we only post multiply by an arbitrary matrix $\mathbf{V}$ of size $N_{\boldsymbol{\theta}} \times n$.

$$\mathbf{GV} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathbf{Z} + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}\mathbf{V}, \tag{7.11}$$

where

$$\mathbf{Z} = -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)^{-1}\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}\mathbf{V}, \tag{7.12}$$

and is solved from

$$\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)\mathbf{Z} = \frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}\mathbf{V} \tag{7.13}$$

Now, we are able to use the super vector notation also from Chapter [5]. Let ν denote the iteration level and N_ν the number of iterations. If we write $\mathbf{Z} = \left(\mathbf{Z}^0, \cdots, \mathbf{Z}^\nu\right)^\top$, then we may express equation (7.13) in a matrix block-form as

$$\begin{pmatrix} \frac{\partial \mathbf{g}^0}{\partial \mathbf{x}^0} & & & \\ \frac{\partial \mathbf{g}^1}{\partial \mathbf{x}^0} & \frac{\partial \mathbf{g}^1}{\partial \mathbf{x}^1} & & \\ & \ddots & & \\ & & \frac{\partial \mathbf{g}^{N_\nu}}{\partial \mathbf{x}^{N_\nu-1}} & \frac{\partial \mathbf{g}^{N_\nu}}{\partial \mathbf{x}^{N_\nu}} \end{pmatrix}\begin{pmatrix} \mathbf{Z}^0 \\ \mathbf{Z}^1 \\ \vdots \\ \mathbf{Z}^\nu \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{g}^0}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathbf{g}^1}{\partial \boldsymbol{\theta}} \\ \vdots \\ \frac{\partial \mathbf{g}^\nu}{\partial \boldsymbol{\theta}} \end{pmatrix}\mathbf{V} \tag{7.14}$$

For all iterations $\nu \in \{0, \cdots, N_\nu\}$ we have

$$\left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^{\nu-1}}\right)(\mathbf{Z}^{\nu-1}) + \left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^\nu}\right)(\mathbf{Z}^\nu) = -\left(\frac{\partial \mathbf{g}^\nu}{\partial \boldsymbol{\theta}}\right)\mathbf{V} \tag{7.15}$$

From this we clearly can see that this system is solved forward in time. By using the iterative forward model equations as in Chapter [5], we may conclude that

$$\left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^\nu}\right)^\top = \begin{pmatrix} \check{\mathbf{A}}^\top & -\mathbf{P}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}^\top & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}^\top & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}, \quad \left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^{\nu-1}}\right)^\top = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}^\top\mathbf{R}^\top & \mathbf{0} & \mathbf{A}^\top & \mathbf{0} \\ \mathbf{A}^\top\mathbf{R}^\top & \mathbf{0} & \mathbf{A}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \end{pmatrix} \tag{7.16}$$

Then, equation (7.15), gives the following system of equations for all iteration levels ν

$$\begin{aligned} \check{\mathbf{Z}}^\nu &= \check{\mathbf{A}}^{-1}\left(-\frac{\partial \check{\mathbf{g}}^\nu}{\partial \boldsymbol{\theta}}\mathbf{V} - \mathbf{RA}\left(\mathbf{Z}' + \mathbf{Z}_\sigma\right)^{\nu-1}\right) \\ \mathbf{Z}'^\nu &= -\mathbf{P}\check{\mathbf{Z}}^\nu - \frac{\partial \mathbf{g}'^\nu}{\partial \boldsymbol{\theta}}\mathbf{V} \\ \mathbf{Z}_\sigma^\nu &= \mathbf{A}^{-1}\left(-\frac{\partial \mathbf{g}_\sigma^\nu}{\partial \boldsymbol{\theta}}\mathbf{V} - \mathbf{AZ}'^\nu - \mathbf{A}\left(\mathbf{Z}' + \mathbf{Z}_\sigma\right)^{\nu-1}\right) \\ \mathbf{Z}_\mathbf{x}^\nu &= \mathbf{Z}_\mathbf{x}^{\nu-1} + \mathbf{Z}'^\nu + \mathbf{Z}_\sigma^\nu \end{aligned}$$

Note that we neglect the dependency of $\mathbf{R}, \mathbf{q}$ on $\boldsymbol{\theta}$. Then we may write all the partial derivatives of the super vector $\mathbf{g}$ with respect to theta as

$$\frac{\partial \check{\mathbf{g}}^{\nu}}{\partial \boldsymbol{\theta}} = \left[\mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + (\mathbf{RA})\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\right]\boldsymbol{\delta}\check{\mathbf{x}}^{\nu} + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\sum_{\tau=1}^{\nu-2}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_{\sigma})^{\tau}$$

$$\frac{\partial \mathbf{g}'^{\nu}}{\partial \boldsymbol{\theta}} = -\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\boldsymbol{\delta}\check{\mathbf{x}}^{\nu}$$

$$\frac{\partial \mathbf{g}_{\sigma}^{\nu}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\boldsymbol{\delta}\mathbf{x}_{\sigma}^{\nu} + \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\boldsymbol{\delta}\mathbf{x}'^{\nu} + \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\sum_{\tau=1}^{\nu-2}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_{\sigma})^{\tau}$$

$$\frac{\partial \mathbf{g}_{\mathbf{x}}^{\nu}}{\partial \boldsymbol{\theta}} = \mathbf{0}$$

So finally we may compute the product $\mathbf{GV}$ as

$$\mathbf{GV} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}^{N_{\nu}}}\mathbf{Z}_{\mathbf{x}} + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}\mathbf{V} \tag{7.19}$$

The formal algorithm of this method can be found in algorithm [3].

Algorithm 3: Direct method for iterative Multi-Scale gradient computation.

input : $\mathbf{R}, \mathbf{A}, \mathbf{P}, \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}, \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}, \frac{\partial \mathbf{h}}{\partial \mathbf{x}}, \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}, \mathbf{x}, \mathbf{V}$
output: The product $\mathbf{GV}$

Compute $\alpha = (\mathbf{RAP})^{-\mathsf{T}}$;
Compute $\beta = \mathbf{A}^{-\mathsf{T}}$;
for $j = 1, \cdots, n$ **do**
  for $\nu = 0, \cdots, N_{\nu}$ **do**
    Compute $\gamma = \frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\boldsymbol{\delta}\check{\mathbf{x}}^{\nu}\mathbf{V}_{.,j}$;
    Compute $\varphi_1 = \mathbf{RA}\gamma + \mathbf{R}\left(\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P} + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\sum_{\tau=1}^{\nu-2}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_{\sigma})^{\tau}\right)$;
    Compute $\varphi_2 = -\gamma$;
    Compute $\varphi_3 = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\boldsymbol{\delta}\mathbf{x}_{\sigma}^{\nu} + \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\boldsymbol{\delta}\mathbf{x}'^{\nu} + \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\sum_{\tau=1}^{\nu-2}(\boldsymbol{\delta}\mathbf{x}' + \boldsymbol{\delta}\mathbf{x}_{\sigma})^{\tau}$;
    Solve $\check{\mathbf{Z}}_{.,j}^{\nu} = \alpha\left(-\varphi_1\mathbf{V}_{.,j} - \mathbf{RA}\left(\mathbf{Z}'_{.,j} + \mathbf{Z}_{\sigma.,j}\right)^{\nu-1}\right)$;
    Compute $\mathbf{Z}'^{\nu}_{.,j} = -\mathbf{P}\check{\mathbf{Z}}^{\nu}_{.,j} - \varphi_2\mathbf{V}_{.,j}$;
    Solve $\mathbf{Z}_{\sigma}^{\nu} = \beta\left(-\varphi_3\mathbf{V}_{.,j} - \mathbf{A}\mathbf{Z}'^{\nu}_{.,j} - \mathbf{A}\left(\mathbf{Z}'_{.,j} + \mathbf{Z}_{\sigma.,j}\right)^{\nu-1}\right)$;
    Compute $\mathbf{Z}_{\mathbf{x}.,j} = \mathbf{Z}^{\nu-1}_{\mathbf{x}.,j} + \mathbf{Z}'^{\nu}_{.,j} + \mathbf{Z}^{\nu}_{\sigma.,j}$;
  Compute $\mathbf{GV} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}^{\nu}}\mathbf{Z}^{N_{\nu}}_{\mathbf{x}.,j} + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}\mathbf{V}_{.,j}$;

7.3.2. Adjoint Method

For the Adjoint Method we employ a similar strategy, however instead of post mulitplying, we now pre multiply $\mathbf{G}$ with an arbitrary matrix of dimensions $m \times N_Y$. Therefore we find

$$\mathbf{WG} = \mathbf{Z}\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} + \mathbf{W}\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}, \quad \mathbf{Z} = -\mathbf{W}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)^{-1} \tag{7.20}$$

The matrix $\mathbf{Z}$ is solved from

$$\mathbf{Z}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right) = -\mathbf{W}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}, \tag{7.21}$$

and where $\mathbf{Z}$ is also written in super vector notation, hence $\mathbf{Z}^\nu = (\check{\mathbf{Z}}, \mathbf{Z}', \mathbf{Z}_\sigma, \mathbf{Z}_\mathbf{x})^\nu$ Denote ν as the iteration level and N_ν as the total number of iteration levels. Then, we may write this as block form

$$\begin{pmatrix} \mathbf{Z}^0 & \mathbf{Z}^1 & \cdots & \mathbf{Z}^{N_\nu} \end{pmatrix} \begin{pmatrix} \frac{\partial \mathbf{g}^0}{\partial \mathbf{x}^0} & & & \\ \frac{\partial \mathbf{g}^1}{\partial \mathbf{x}^0} & \frac{\partial \mathbf{g}^1}{\partial \mathbf{x}^1} & & \\ & \ddots & & \\ & & \frac{\partial \mathbf{g}^{N_\nu}}{\partial \mathbf{x}^{N_\nu - 1}} & \frac{\partial \mathbf{g}^{N_\nu}}{\partial \mathbf{x}^{N_\nu}} \end{pmatrix} = -\mathbf{W} \begin{pmatrix} \frac{\partial \mathbf{h}}{\partial \mathbf{x}^0} & \frac{\partial \mathbf{h}}{\partial \mathbf{x}^1} & \cdots & \frac{\partial \mathbf{h}}{\partial \mathbf{x}^{N_\nu}} \end{pmatrix}, \tag{7.22}$$

where $\mathbf{Z}^\nu = \left(\check{\mathbf{Z}}, \mathbf{Z}', \mathbf{Z}_\sigma, \mathbf{Z}_\mathbf{x}\right)^\nu$. We can write this in a more convenient form by transposing this system, resulting in

$$\begin{pmatrix} (\frac{\partial \mathbf{g}^0}{\partial \mathbf{x}^0})^\mathsf{T} & (\frac{\partial \mathbf{g}^1}{\partial \mathbf{x}^0})^\mathsf{T} & & \\ & (\frac{\partial \mathbf{g}^1}{\partial \mathbf{x}^1})^\mathsf{T} & \ddots & \\ & & \ddots & (\frac{\partial \mathbf{g}^{N_\nu}}{\partial \mathbf{x}^{N_\nu - 1}})^\mathsf{T} \\ & & & (\frac{\partial \mathbf{g}^{N_\nu}}{\partial \mathbf{x}^{N_\nu}})^\mathsf{T} \end{pmatrix} \begin{pmatrix} (\mathbf{Z}^0)^\mathsf{T} \\ (\mathbf{Z}^1)^\mathsf{T} \\ \vdots \\ (\mathbf{Z}^{N_\nu})^\mathsf{T} \end{pmatrix} = -\begin{pmatrix} (\frac{\partial \mathbf{h}}{\partial \mathbf{x}^0})^\mathsf{T} \\ (\frac{\partial \mathbf{h}}{\partial \mathbf{x}^1})^\mathsf{T} \\ \vdots \\ (\frac{\partial \mathbf{h}}{\partial \mathbf{x}^{N_\nu}})^\mathsf{T} \end{pmatrix} \mathbf{W}^\mathsf{T} \tag{7.23}$$

For all iterations $\nu \in \{0, \cdots, N_\nu\}$ we have

$$\left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^\nu}\right)^\mathsf{T} (\mathbf{Z}^\nu)^\mathsf{T} + \left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^{\nu-1}}\right)^\mathsf{T} (\mathbf{Z}^{\nu+1})^\mathsf{T} = -\left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \tag{7.24}$$

Here we notice the difference with the forward method. This system of equations is not solved forward in time, but backwards. Once more, from Chapter [5], we have that

$$\left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^\nu}\right)^\mathsf{T} = \begin{pmatrix} \check{\mathbf{A}}^\mathsf{T} & -\mathbf{P}^\mathsf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}^\mathsf{T} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}^\mathsf{T} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}, \quad \left(\frac{\partial \mathbf{g}^\nu}{\partial \mathbf{x}^{\nu-1}}\right)^\mathsf{T} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}^\mathsf{T}\mathbf{R}^\mathsf{T} & \mathbf{0} & \mathbf{A}^\mathsf{T} & \mathbf{0} \\ \mathbf{A}^\mathsf{T}\mathbf{R}^\mathsf{T} & \mathbf{0} & \mathbf{A}^\mathsf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \end{pmatrix} \tag{7.25}$$

Plugging (7.25) into (7.24), leaves us with the following system of equations for all iteration steps ν

$$\begin{aligned} \check{\mathbf{A}}^\mathsf{T} \left(\check{\mathbf{Z}}^\nu\right)^\mathsf{T} - \mathbf{P}^\mathsf{T} (\mathbf{Z}'^\nu)^\mathsf{T} &= -\left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta} \check{\mathbf{x}}^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \\ (\mathbf{Z}'^\nu)^\mathsf{T} + \mathbf{A}^\mathsf{T} \left(\mathbf{Z}_\sigma^\nu\right)^\mathsf{T} - \left(\mathbf{Z}_\mathbf{x}^\nu\right)^\mathsf{T} + \mathbf{A}^\mathsf{T}\mathbf{R}^\mathsf{T} \left(\check{\mathbf{Z}}^{\nu+1}\right)^\mathsf{T} + \mathbf{A}^\mathsf{T} \left(\mathbf{Z}_\sigma^{\nu+1}\right)^\mathsf{T} &= -\left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta} \mathbf{x}'^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \\ \mathbf{A}^\mathsf{T} \left(\mathbf{Z}_\sigma^\nu\right)^\mathsf{T} - \left(\mathbf{Z}_\mathbf{x}^\nu\right)^\mathsf{T} + \mathbf{A}^\mathsf{T} \left(\mathbf{Z}_\sigma^{\nu+1}\right)^\mathsf{T} + \mathbf{A}^\mathsf{T}\mathbf{R}^\mathsf{T} \left(\check{\mathbf{Z}}^{\nu+1}\right) &= -\left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta} \mathbf{x}_\sigma^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \\ \left(\mathbf{Z}_\mathbf{x}^\nu\right)^\mathsf{T} - \left(\mathbf{Z}_\mathbf{x}^{\nu+1}\right)^\mathsf{T} &= -\left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \end{aligned}$$

Finally we may compute all the $\mathbf{Z}$ terms by rearranging the terms. We notice that we can make use of backwards substitution to compute all the terms. Hence we solve

$$\begin{aligned} \left(\mathbf{Z}_\mathbf{x}^\nu\right)^\mathsf{T} &= \left(\mathbf{Z}_\mathbf{x}^{\nu+1}\right)^\mathsf{T} - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \\ \left(\mathbf{Z}_\sigma^\nu\right)^\mathsf{T} &= \left(\mathbf{A}^{-\mathsf{T}}\right) \left(\left(\mathbf{Z}_\mathbf{x}^\nu\right)^\mathsf{T} - \mathbf{A}^\mathsf{T} \left(\mathbf{Z}_\sigma^{\nu+1}\right)^\mathsf{T} - \mathbf{A}^\mathsf{T}\mathbf{R}^\mathsf{T} \left(\check{\mathbf{Z}}^{\nu+1}\right) - \left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta} \mathbf{x}_\sigma^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \right) \\ (\mathbf{Z}'^\nu)^\mathsf{T} &= -\mathbf{A}^\mathsf{T} \left(\mathbf{Z}_\sigma^\nu\right)^\mathsf{T} + \left(\mathbf{Z}_\mathbf{x}^\nu\right)^\mathsf{T} - \mathbf{A}^\mathsf{T}\mathbf{R}^\mathsf{T} \left(\check{\mathbf{Z}}^{\nu+1}\right)^\mathsf{T} - \mathbf{A}^\mathsf{T} \left(\mathbf{Z}_\sigma^{\nu+1}\right)^\mathsf{T} - \left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta} \mathbf{x}'^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \\ \left(\check{\mathbf{Z}}^\nu\right)^\mathsf{T} &= \left(\check{\mathbf{A}}^{-\mathsf{T}}\right) \left(\mathbf{P}^\mathsf{T} (\mathbf{Z}'^\nu)^\mathsf{T} - \left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta} \check{\mathbf{x}}^\nu}\right)^\mathsf{T} \mathbf{W}^\mathsf{T} \right) \end{aligned} \tag{7.27a}$$

Algorithm 4: Adjoint method for iterative Multi-Scale gradient computation.

input : $\mathbf{R}, \mathbf{A}, \mathbf{P}, \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}, \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}, \frac{\partial \mathbf{h}}{\partial \mathbf{x}}, \frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}, \mathbf{x}, \mathbf{W}$
output: The product $\mathbf{WG}$

Initialize $\mathbf{WG} = 0$;
Compute $\alpha = (\mathbf{RAP})^{-\mathsf{T}}$;
Compute $\beta = \mathbf{A}^{-\mathsf{T}}$;
for $i = 1, \cdots, m$ **do**
  for $\nu = N_\nu, \cdots, 0$ **do**
    Compute $\psi = \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\mathbf{P}\check{\mathbf{x}}^\nu - \mathbf{R}\frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}} + \mathbf{R}\frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\sum_{\tau=1}^{\nu-2}(\mathbf{x}' + \mathbf{x}_\sigma)^\tau$;
    Compute $\delta = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}}\sum_{\tau=1}^{\nu-1}(\mathbf{x}' + \mathbf{x}_\sigma)^\tau - \frac{\partial \mathbf{q}}{\partial \boldsymbol{\theta}}$;
    if $\nu = N_\nu$ **then**
      Compute $\left(\mathbf{Z}_{\mathbf{x}}^\nu\right)_{i,\cdot}^{\mathsf{T}} = \left(\mathbf{Z}_{\mathbf{x}}^{\nu+1}\right)^{\mathsf{T}} - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}^\nu}\right)^{\mathsf{T}} \mathbf{W}_{i,\cdot}^{\mathsf{T}}$;
    else
      Set $\left(\mathbf{Z}_{\mathbf{x}}^\nu\right)_{i,\cdot}^{\mathsf{T}} = \left(\mathbf{Z}_{\mathbf{x}}^{\nu+1}\right)^{\mathsf{T}}$
    Solve $\left(\mathbf{Z}_\sigma^\nu\right)_{i,\cdot}^{\mathsf{T}} = \beta\left(\left(\mathbf{Z}_{\mathbf{x}}^\nu\right)_{i,\cdot}^{\mathsf{T}} - \mathbf{A}^{\mathsf{T}}\left(\mathbf{Z}_\sigma^{\nu+1}\right)_{i,\cdot}^{\mathsf{T}}\right)$;
    Compute $(\mathbf{Z}'^\nu)_{i,\cdot}^{\mathsf{T}} = -\mathbf{A}^{\mathsf{T}}\left(\mathbf{Z}_\sigma^\nu\right)_{i,\cdot}^{\mathsf{T}} + \left(\mathbf{Z}_{\mathbf{x}}^\nu\right)_{i,\cdot}^{\mathsf{T}} - \mathbf{A}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\left(\check{\mathbf{Z}}^{\nu+1}\right)_{i,\cdot}^{\mathsf{T}} - \mathbf{A}^{\mathsf{T}}\left(\mathbf{Z}_\sigma^{\nu+1}\right)_{i,\cdot}^{\mathsf{T}}$;
    Solve $\left(\check{\mathbf{Z}}^\nu\right)_{i,\cdot}^{\mathsf{T}} = \alpha\left(\mathbf{P}^{\mathsf{T}}\left(\mathbf{Z}'^\nu\right)_{i,\cdot}^{\mathsf{T}}\right)$;
    Compute $\mathbf{m}^{\mathsf{T}} = \check{\mathbf{Z}}_{i,\cdot}^\nu \mathbf{RA} - \mathbf{Z}_{i,\cdot}'^\nu$;
    Compute $\gamma = \mathbf{m}^{\mathsf{T}}\left(\frac{\partial \mathbf{P}}{\partial \boldsymbol{\theta}}\check{\mathbf{x}}^\nu\right)$ as in algorithm 4 found in [25];
    Compute $\eta = \check{\mathbf{Z}}_{i,\cdot}^\nu \psi + \gamma + \mathbf{Z}_{\sigma_{i,\cdot}}^\nu \delta + \mathbf{W}_{i,\cdot}\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}$;
    Compute $(\mathbf{WG})_{i,\cdot} = (\mathbf{WG})_{i,\cdot} + \eta$;

Once $\mathbf{Z}$ is computed we can update the $\mathbf{WG}$ product at each backward iteration by

$$(\mathbf{WG})^\nu = (\mathbf{WG})^{\nu-1} + \check{\mathbf{Z}}^\nu \frac{\partial \check{\mathbf{g}}^\nu}{\partial \boldsymbol{\theta}} + \mathbf{Z}'^\nu \frac{\partial \mathbf{g}'^\nu}{\partial \boldsymbol{\theta}} + \mathbf{Z}_\sigma^\nu \frac{\partial \mathbf{g}_\sigma^\nu}{\partial \boldsymbol{\theta}} \tag{7.28}$$

In conclusion we compute the product as

$$\mathbf{WG} = (\mathbf{WG})^0 + \mathbf{W}\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} \tag{7.29}$$

Now we remark that our response function is only dependent on the converged, or iteratively improved solution. Hence the observations may be describes as

$$\mathbf{h}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{h}(\mathbf{x}^{N_\nu}, \boldsymbol{\theta})$$

In this manner, we can simplify the partial derivative of our response function to

$$\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial \mathbf{h}^0}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathbf{h}^1}{\partial \boldsymbol{\theta}} \\ \vdots \\ \frac{\partial \mathbf{h}^{N_\nu - 1}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathbf{h}^{N_\nu}}{\partial \boldsymbol{\theta}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \frac{\partial \mathbf{h}^{N_\nu}}{\partial \boldsymbol{\theta}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}\check{\mathbf{x}}^{N_\nu}} \\ \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}\mathbf{x}'^{N_\nu}} \\ \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}\mathbf{x}_\sigma^{N_\nu}} \\ \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}\check{\mathbf{x}}'^{N_\nu}} \\ \frac{\partial \mathbf{h}}{\partial \mathbf{x}^{N_\nu}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}\mathbf{x}^{N_\nu}} \end{pmatrix} \tag{7.30}$$

This method is summarized in algorithm [4].

7.4. Comparison between both methods

In this section we will discuss some of the advantages and disadvantages of both methods. The main difference between both methods is explained in terms of the computational efficiency. The Forward

or Direct Method has a computational cost proportional to the columns of $\mathbf{V}$. This can be seen by considering equations (7.5) and (7.12). Since $\mathbf{Z}$ has dimensions $N_C \times n$, in this equations n linear systems need to be solved. One can see that if we need to compute the full sensitivity matrix $\mathbf{G}$, i.e. $\mathbf{V} = \mathbf{I}$, the cost will be proportional to the number of parameters N_θ. For the Backward or Adjoint method we note that $\mathbf{Z}$ has dimensions of $N_c \times m$. From equations (7.7) and (7.21) we may now conclude that the computational cost of this method is proportional to the rows of $\mathbf{W}$ since m linear systems need to be solved. When we want to compute the full sensitivity matrix $\mathbf{G}$, i.e $\mathbf{W} = \mathbf{I}$, the computational cost is proportional to N_Y, or the number of observations. Since in Reservoir Simulation th number of parameters generally is much more higher then the number of responses, from this we may conclude that using the Adjoint Method is more efficient, since less computations are required to compute the product.

However, the Adjoint method also has a drawback. Since we operate backward through the iteration levels, to compute the final product we must store the intermediate results. Therefore, from a storage point of view this is a limitation to the method. The computational efficiency however, trumps the drawback in most situations. Therefore, we are likely to use the adjoint method when dealing with complex reservoir simulation gradient computations.

8

Numerical i-MS Gradient Computation Experiments

8.1. Introduction

In this chapter we will study the performance of the iterative Multiscale Gradient Computation algorithms as described previously. We also refer the reader to the (unpublished) results presented in [26]. The main goal is to show that we can estimate the computationally heavy fine scale gradient by deploying a iterative multiscale method in sense of alignment. In this section we will assume a misfit objective function with no regularization term, i.e.

$$O(\mathbf{h}) = \frac{1}{2}(\mathbf{h}(\mathbf{x},\boldsymbol{\theta}) - d_{obs})^{\top} C^{-1} (\mathbf{h}(\mathbf{x},\boldsymbol{\theta}) - d_{obs}) \tag{8.1}$$

We note that the analytical gradient formulation of equation (8.1) is given by

$$\nabla_{\boldsymbol{\theta}} O(\mathbf{h}) = (\mathbf{h}(\mathbf{x},\boldsymbol{\theta}) - d_{obs})^{\top} C^{-1} \mathbf{G}, \tag{8.2}$$

This also illustrates the benefit using a framework where we pre and post multiplying the sensitivity matrix. If we set $\mathbf{W} = (\mathbf{h}(\mathbf{x},\boldsymbol{\theta}) - d_{obs})^{\top} C^{-1}$, then we are able to use the algorithms in Chapter [7]. In this case, $\mathbf{W}$ is the misfit matrix. Next to this, we assume that the observed quantities d_{obs} are observed only on fine scale, which is comparable to the setup in the previous chapter. Hence we remind the reader that if this is the case, all of the partial derivatives of $\mathbf{h}$ with respect to a component of $\mathbf{x}$ is zero, except for the $\left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}\mathbf{x}^{N_v}}\right)$ term.

In the objective function (8.1) we consider observed quantities d_{obs}. The observations are the pressures measured at the locations of the observation wells and are obtained by taking the pressures from a reference case where the permeability field is taken from an ensemble of heterogeneous fields. In total there are 1000 available geological realisations. These fields are generated using the Principal Component Analysis parametrization. We refer the reader to [15] for more information. Multiple examples of these fields can be found in Figure [8.1]. In all experiment we have grid-block dimensions of $\Delta x = \Delta y = \Delta z = 1m$. Furthermore we have a fluid viscosity of 1.0×10^{-3} Pa s. At the injection wells there is a constant pressure of $p_{inj} = 1.0$ Pa, and at the production well we have $p_{prod} = 0.0$ Pa.

8.2. Validation experiments

In this section we will validate the iMS-gradient method against the numerical differentiation method with a higher order, two-sided Taylor approximation

$$\nabla_{\boldsymbol{\theta}} O_i = \frac{1}{2\delta\theta_i}\left(O(\theta_1,\cdots,\theta_{i-1},\theta_i+\delta\theta_i,\theta_{i+1},\cdots,\theta_{N_\theta}) - O(\theta_1,\cdots,\theta_{i-1},\theta_i-\delta\theta_i,\theta_{i+1},\cdots,\theta_{N_\theta})\right), \tag{8.3}$$

where we consider δ to be a multiplicative parameter perturbation. We define the relative error as

$$\varepsilon = \frac{||\nabla_{\boldsymbol{\theta}} O_{NUM} - \nabla_{\boldsymbol{\theta}} O_{iMS}||_2}{||\nabla_{\boldsymbol{\theta}} O_{iMS}||_2} \tag{8.4}$$

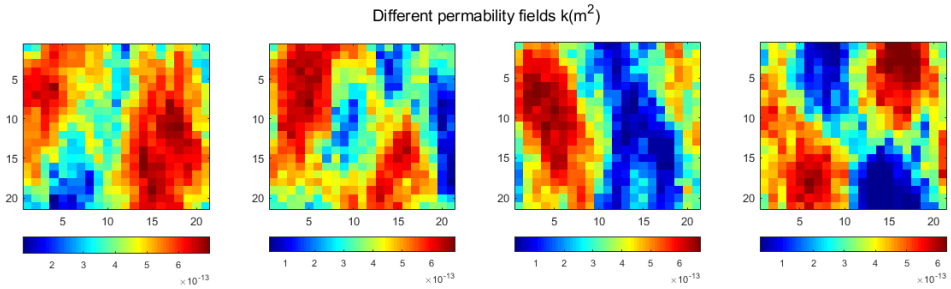
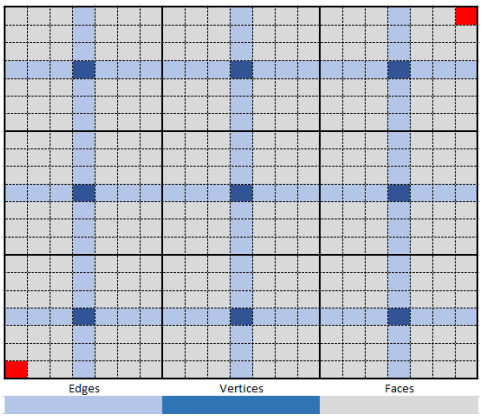


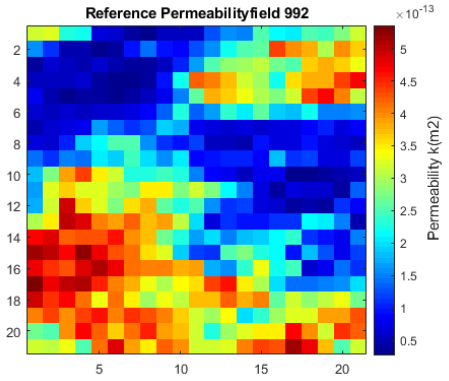
Figure 8.1: Different Realisations of the heterogeneous permeability fields

Here $\nabla_{\boldsymbol{\theta}} O_{NUM}$ is obtained by performing proper amount of iterative multiscale reservoir simulations required to evaluate equation (8.3). $\nabla_{\boldsymbol{\theta}} O_{iMS}$ is obtained by employing the iterative direct- or iterative adjoint gradient method.

To evaluate the correctness of the implementation and the proposed iterative gradient computation methods, we investigate the linear decrease of the relative error ε by decreasing the parameter perturbation δ from 10^{-1} to 10^{-4}. This investigation is carried out in two distinct examples. Both have a fine grid of 21×21 grid blocks. We employ a 7×7 coarsening ratio, giving a 3×3 coarse grid. The reference twin-experiment is generated with permeability realization number 992. Figure [8.2a] illustrates the fine-, coarse- and dual grid cells and the reference permeability is visualized in Figure [8.2b] .



(a) Visualization of grid specification



(b) Visualization of reference permeability field number 992

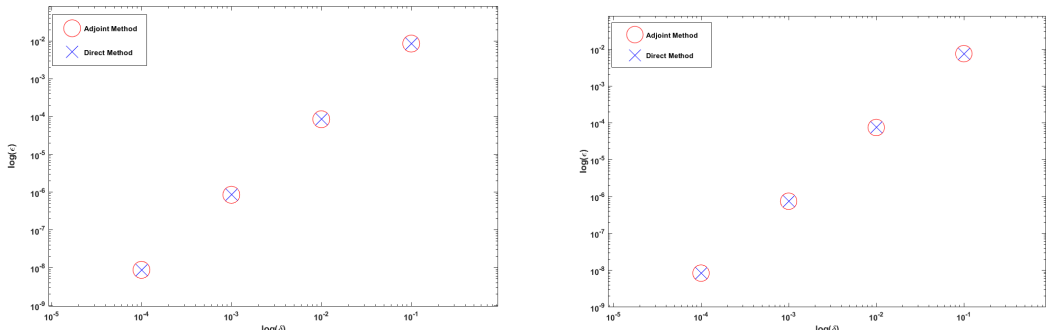
Next we determine the well positions. We use the so-called quarter well spot. Here, two observation wells are placed near operating wells. The full specifications can be found in table [8.1]

Well positions			
Name	**Position**	**Well Type**	**Pressure (Pa)**
INJE	$(1, 1)$	Injection	$1.0 \cdot 10^0$
PROD	$(21, 21)$	Production	$0.0 \cdot 10^0$
OBS1	$(3, 3)$	Observation	
OBS2	$(19, 19)$	Observation	

Table 8.1: Quarter well spot setup.

The results of this experiment are found in Figure [8.3]. Here, we use a single outer iteration. We use a very high smoothing tolerance of $\epsilon_\sigma = 5 \times 10^{-18}$ to ensure that the numerical gradient method produces

accurate enough gradients. First of all we can see that the fine-scale numerical gradient method and the iterative MS-gradient methods are of the same order of accuracy with respect to the perturbation δ, for all different cases considered. In all experiments, we can see the linear decreasing behaviour of the relative error values ε as the perturbation δ decreases. Also we may easily see that the Adjoint- and Direct method are equally accurate. They provide the analytical gradient at the same accuracy. The second experiment indicates the correctness of the method when it is applied to heterogeneous porous media problems.



(a) (b)

Figure 8.3: Validation of the iMS gradient computation method compared with a numerical gradient. (a) represents the homogeneous test case, while (b) represents the heterogeneous test case.

8.3. Gradient accuracy

We perform multiple numerical experiments to assess the quality of the i-MS Gradient algorithms. We will discuss the performance over a realization of 1000 heterogeneous permeability fields. We discuss the quality of the i-MS gradient in comparison to the fine-scale gradient. Suppose that the fine-scale gradient and i-MS gradient are denoted by $\nabla_{\boldsymbol{\theta}} O_{FS}, \nabla_{\boldsymbol{\theta}} O_{iMS}$ respectively. To asses the gradient we use the angle metric between the two gradients. This angle is given by

$$\cos(\alpha) = \frac{\langle \nabla_{\boldsymbol{\theta}} O_{FS}, \nabla_{\boldsymbol{\theta}} O_{iMS} \rangle}{||\nabla_{\boldsymbol{\theta}} O_{FS}||_2 \cdot ||\nabla_{\boldsymbol{\theta}} O_{iMS}||_2} = \nabla_{\boldsymbol{\theta}} \hat{O}_{FS}^{\top} \nabla_{\boldsymbol{\theta}} \hat{O}_{iMS} \tag{8.5}$$

where

$$\nabla_{\boldsymbol{\theta}} \hat{O}_{FS} = \frac{\nabla_{\boldsymbol{\theta}} O_{FS}}{||\nabla_{\boldsymbol{\theta}} O_{FS}||_2}, \quad \nabla_{\boldsymbol{\theta}} \hat{O}_{iMS} = \frac{\nabla_{\boldsymbol{\theta}} O_{iMS}}{||\nabla_{\boldsymbol{\theta}} O_{iMS}||_2} \tag{8.6}$$

Hence, the angle between the two vector may be computed as

$$\alpha = \cos^{-1}\left(\nabla_{\boldsymbol{\theta}} \hat{O}_{FS}^{\top} \nabla_{\boldsymbol{\theta}} \hat{O}_{iMS}\right) \tag{8.7}$$

If the angle is close to zero, this means that the i-MS gradient has the same direction as the fine scale gradient. For the one dimensional case, we have that $\alpha = 0°$, since there are no localization errors in the MS solution, we do not need any iterations and hence, the i-MS solution gradient has no errors. Note that the angle does not tell us if the gradients are equal to one another. If merely tells us if they are aligned. As a minimum requirement, we accept only the iMS gradients that have an angle α much smaller then $90°$ [10]. We investigate the behaviour of the metric as a function of the number of iterations. The number of outer iterations in the iterative process is controlled by the residual $||\mathbf{r}^{\nu}|| > \epsilon$, where the iterative process continues until this pre-set accuracy is achieved. In comparison, the smoothing error is controlled by the smoother tolerance ϵ_{σ}

8.3.1. Investigation of the i-MSFV convergence and gradient quality

In this section we will discuss the behaviour of the i-MSFV gradient computation method and the quality of the gradient. A fine grid of 21×21 is defined. Next to this our coarse grid has dimensions of 7×7. For the well positions we impose an inverted five-well spot with four observation wells close to the production wells. The full specifications of the configuration can be found in table [8.2]. Once more

Well positions			
Name	**Position**	**Well Type**	**Pressure (Pa)**
INJE	$(11, 11)$	Injection	$1.0 \cdot 10^0$
PROD1	$(1, 1)$	Production	$0.0 \cdot 10^0$
PROD2	$(21, 21)$	Production	$0.0 \cdot 10^0$
PROD3	$(1, 21)$	Production	$0.0 \cdot 10^0$
PROD4	$(21, 1)$	Production	$0.0 \cdot 10^0$
OBSWELL1	$(3, 3)$	Observation	
OBSWELL2	$(19, 3)$	Observation	
OBSWELL3	$(3, 19)$	Observation	
OBSWELL4	$(19, 3)$	Observation	

Table 8.2: Inverted five well spot setup

we create observations from a twin experiment.

The results of evaluating the metric with respect to the number of iterations is found in Figure [8.4].

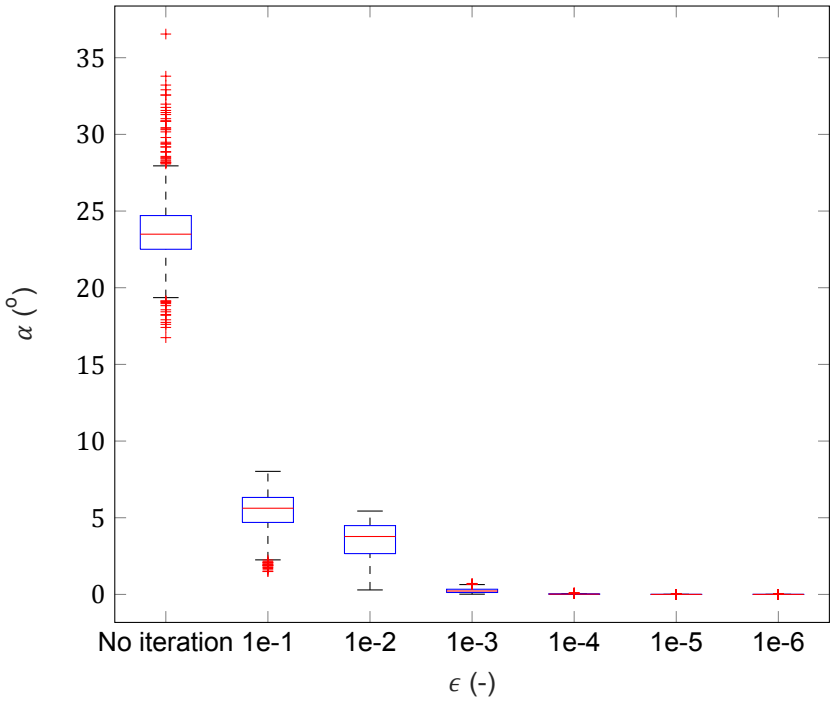


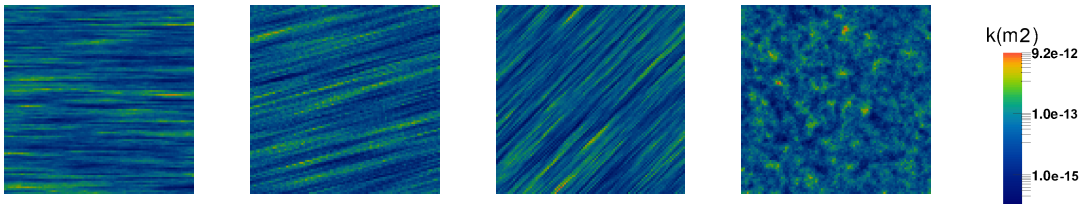
Figure 8.4: Box-plot illustrating the metric α between fine-scale gradient and i-MSFV gradient computed for the $1,000$ member ensemble as a function of the outer-loop tolerance error ϵ. "No iteration" is equivalent to the MSFV gradient computation presented in [25] and as discussed in section 7.2.

From this figure we can clearly see that the angle metric decreases for all realisations if we increase the outer-loop residual tolerance. Next to this we conclude that the variance of the angle decreases significantly and becomes zero if we set $\epsilon = 1 \cdot 10^{-4}$. If the residual is set to $1 \cdot 10^{-5}$ we notice that there is perfect alignment with the fine-scale gradient in all 1000 permeability realizations. However, there is very limited permeability contrast between the different realisations of the ensemble. These permeability fields can be regarded as the uncomplicated geological case. Therefore we assess the robustness for the iterative gradient computation method for geological settings with higher permeability contrasts, and hence increasing the complexity of the problem.

8.3.2. Heterogeneity contrast and distribution

To investigate the robustness of the method on examples with increasing complexity with respect to heterogeneity and distribution, we consider 4 set of 20 equiprobable realizations of log-normally dis-

tributed permeability fields. These are generated using the sequential Gaussian Simulation [40] with a spherical variance of $\psi_1 = 0.5, \psi_2 = 0.02$. In this algorithm you transform data to fit a Gaussian Random Variable. Then you sample multiple sets from this set to obtain the permeability field. For each set we have that the mean of the grid block permeabilities is $\mu = 3.0$ Darcy and the variance is $\sigma^2 = 2.0$ Darcy. We note that if the correlation length ψ is small, there is a patchy, peaky behaviour, whilst if ψ is larger, different permeability streaks or layers appear in different angles. In this experiment we consider three different angles $(0°, 15°, 45°)$ and the patchy permeability sets. Compared with the previous experiment, the permeability contrasts are generally much higher in this experiment. A visualization of the permeability fields is shown in Figure [8.5]



(a) $0°$ (b) $15°$ (c) $45°$ (d) Patchy

Figure 8.5: Visualization of four different realisations taken from the different sets of geostatistically equipobable permeability fields. In figures $(a-c)$ we use different correlation lengths. Also, a patchy field (d) with a small correlation length is considered.

We consider a fine grid of 100×100 grid blocks and the coarse grid contains 20×20 grid blocks. The well setup is similar to the quarter well setup in table [8.1], however the Production well is now in cell number $(100, 100)$ and the second observation well is at grid block $(98, 98)$. Table [8.3]. In this experiment we generate observations from a twin-experiment where the permeability is the first realisation of each set. We set $\epsilon = 1.0 \cdot 10^{-6}$ and $\epsilon_\sigma = 1.0 \cdot 10^{-1}$.

Well positions			
Name	**Position**	**Well Type**	**Pressure (Pa)**
INJE	$(1, 1)$	Injection	$1.0 \cdot 10^0$
PROD	$(100, 100)$	Production	$0.0 \cdot 10^0$
OBSWELL1	$(3, 3)$	Observation	
OBSWELL2	$(98, 98)$	Observation	

Table 8.3: Quarter well spot setup for the heterogeneity distribution experiment.

Now Figure [8.7] shows the value of the angle metric for the different sets, when using the multiscale method and the iterative multiscale method. The box-plot shown in [8.6] summarizes the required total

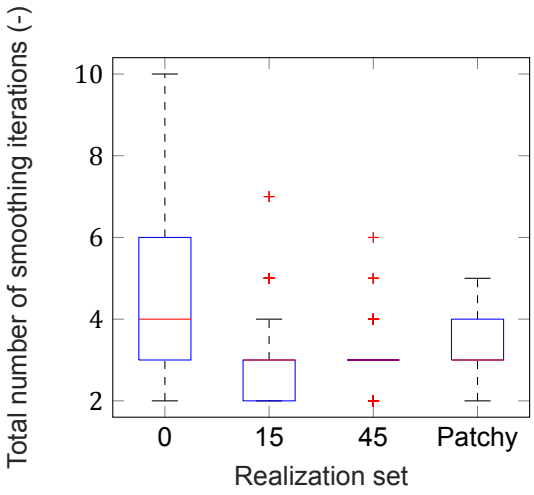
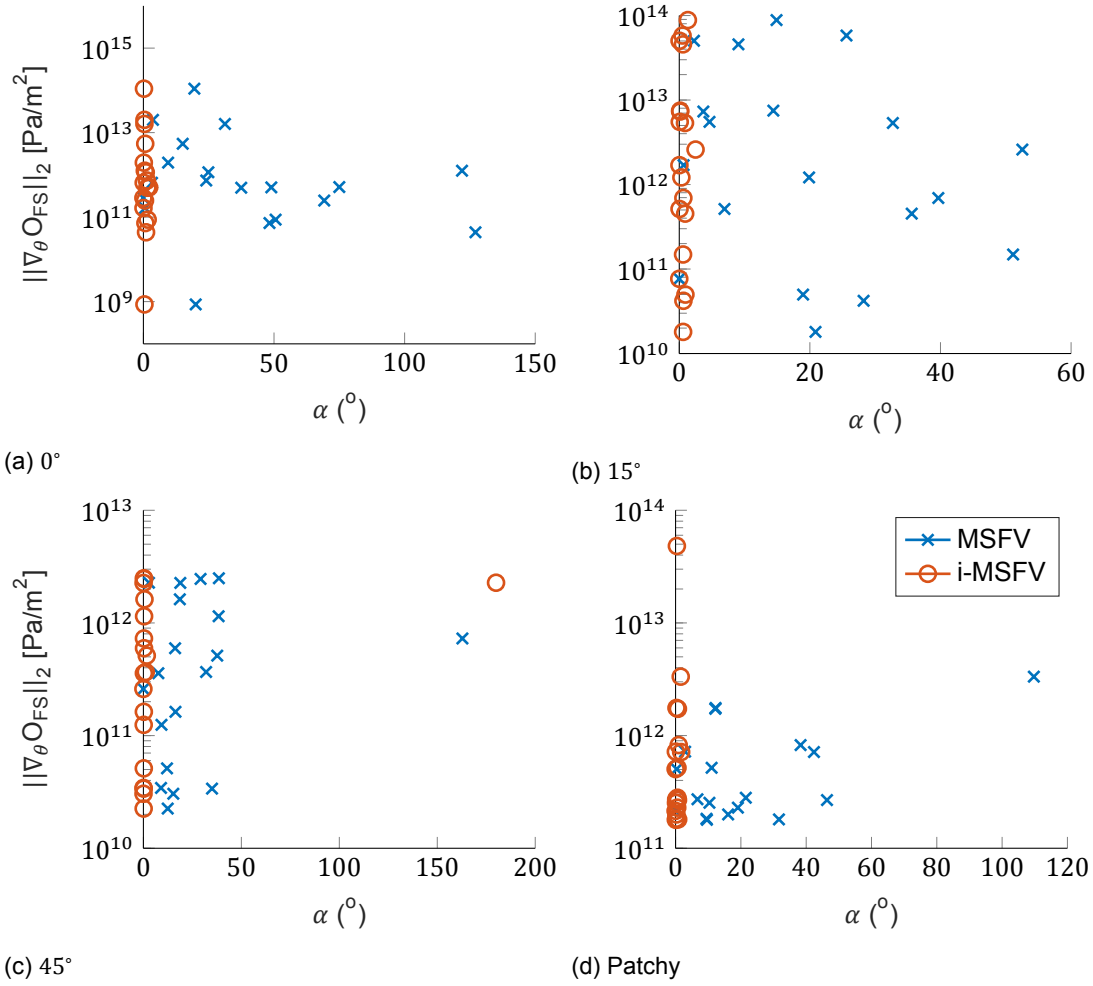


Figure 8.6: Boxplot of the convergence behaviour of the different permeability sets.

number of smoothing steps, for all outer i-MSFV steps, for varying ϵ. We see that if the forward model is more challenging, the computation of the i-MSFV gradient is more challenging to compute as well. However, we can also see that in all cases, almost all of the computed i-MSFV gradient realizations are perfectly aligned with the fine-scale gradient, hence proving the robustness of the method.



(a) 0° (b) 15°

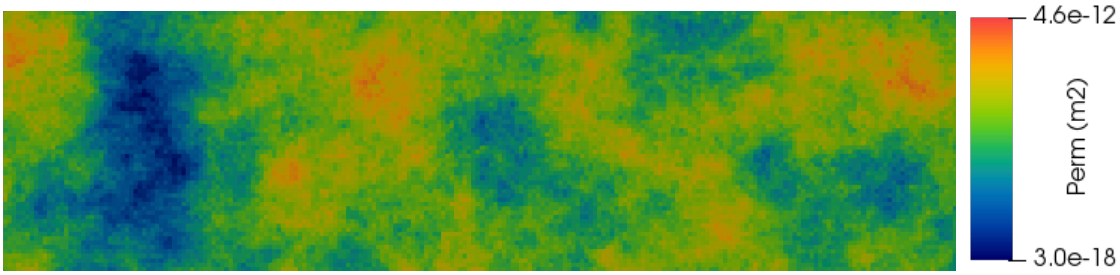
(c) 45° (d) Patchy

Figure 8.7: Illustration of gradient quality improvement when an i-MSFV gradient computation strategy is employed in comparison to a MSFV computation strategy for the different equiprobable permeability fields. The x-axis represent the metric α between fine-scale and the MSFV gradient (illustrated by blue crosses) and the i-MSFV gradient with error tolerance $\epsilon = 10^{-6}$ (illustrated by orange circles).

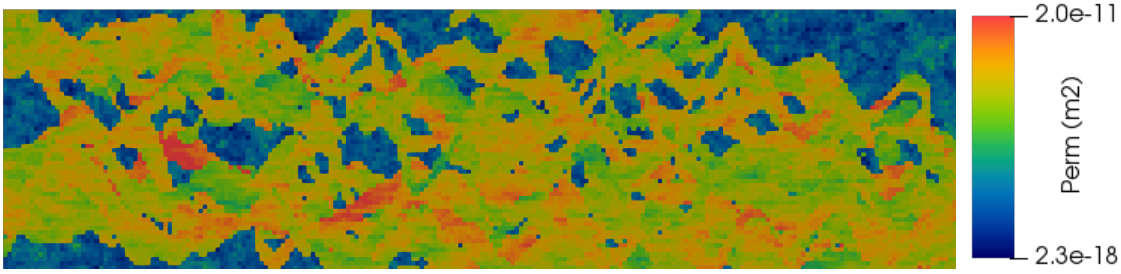
8.3.3. SPE-10 comparative test case

In the final experiment, we investigate the performance of the method it he SPE-10 comparative case [6]. This project provides test data and files for independent comparison of different methods. Previous project focussed on black-oil, horizontal wells and gridding techniques. The aim of the 10^{th} project however, was to compare different upscaling techniques. It considers a fine grid model of $60 \times 220 \times 85$ fine grid cells, with high contrast between permeability regions, throughout the different layers. Therefore, this case is regarded as challenging. In our experiment we consider flow through the top and bottom layer of this model. The corresponding permeability fields can be found in Figure [8.8]. In this permeability field one can see why this model is regarded as complex. There are very high permeability contrasts. Next to this, in the bottom layer there are very specific regions of high- and low contrast adding to the complexity.

We employ a constant coarsening ratio of 5×11, hence generating a coarse grid of 12×20 grid blocks. The observations are generated from a homogeneous twin-experiment with a permeability value of



(a) Top layer



(b) Bottom layer

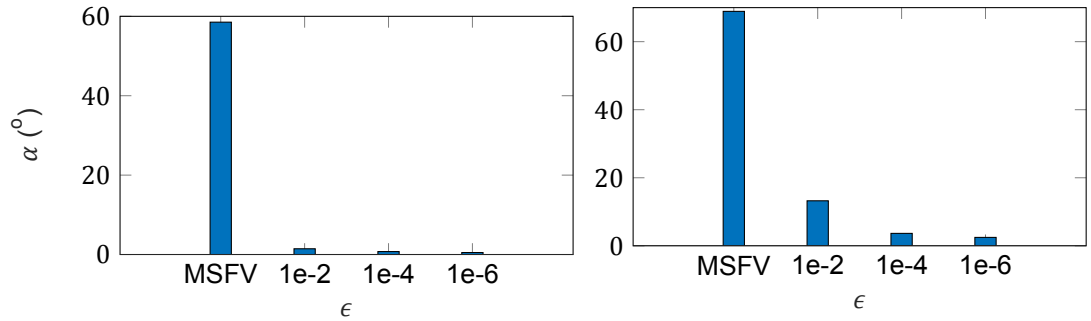
Figure 8.8: SPE-10 comparative test case: top (a) and bottom (b) layer permeability fields.

$k = 1 \cdot 10^{-13} m^2$. The specifications of the well configuration is found in table [8.4]. In this experiment

Well positions			
Name	**Position**	**Well Type**	**Pressure (Pa)**
INJE	$(1, 220)$	Injection	$1.0 \cdot 10^0$
PROD	$(60, 1)$	Production	$0.0 \cdot 10^0$
OBSWELL1	$(33, 5)$	Observation	
OBSWELL2	$(28, 50)$	Observation	
OBSWELL3	$(28, 83)$	Observation	
OSBWELL4	$(43, 204)$	Observation	

Table 8.4: SPE-10 comparative test case well setup.

we fix $\epsilon_\sigma = 1.0 \cdot 10^{-2}$ and we vary the outer residual ϵ with values of $1.0 \cdot 10^{-2}, 1.0 \cdot 10^{-4}$ and $1.0 \cdot 10^{-6}$. Figure [8.9] shows the results for this test case. Once again, we clearly see that for this geological challenging setting we can provide more accurate gradient alignment by using an iterative method when compared to the non-iterative multi-scale gradient. This is for both the top- and the bottom layer, however we do notice that the top layer performs significantly better then the bottom layer, which has more extreme permeability contrasts.



(a) Top layer

(b) Bottom layer

Figure 8.9: i-MSFV gradient quality as a function of residual error ϵ for the SPE-10 top layer (a) and bottom layer (b).

9

Goal-Oriented Adjoint Based Optimization

9.1. Introduction

In the previous chapter we showed the performance of the iterative multiscale gradient computation algorithm. In this algorithm, a smoothing step is applied to remove errors. This step comes with the computational cost of solving a fine scale system, and hence is expensive. In this section we introduce a new method, the iterative multiscale goal oriented method (iMSGo(m_{goal})), with the goal of reducing the computational cost of the fine-scale smoothing. This method identifies the regions of the computational domain that have a high impact with respect to a user defined goal. Then, the smoothing step is performed only on these positions, hence improving the computational efficiency of the system. Selecting these positions mean that we have to permute our system equations to reorder the system accordingly. First, we will describe how to permute the system of equations. Then we determine two types of convergence control criteria. The first being residual-based, the second being the goal oriented adjoint-based convergence criterion. There, we also explain the goal oriented method where two different frameworks are applied. We state the criterion on a fine scale and multiscale basis.

9.2. Adaptive iterative multiscale finite volume method

9.2.1. Permutation of system equations

Fine-scale

Let $\mathcal{P}$ be a $\mathbb{R}^{N_F} \times \mathbb{R}^{N_F}$ permutation matrix [11] that re-orders the original system of equations in a block-wise fashion

$$\left(\mathcal{P}\mathbf{A}\mathcal{P}^T\right)\left(\mathcal{P}\mathbf{x}\right) = \begin{bmatrix}\mathbf{A}_{xx} & \mathbf{A}_{xp} \\ \mathbf{A}_{px} & \mathbf{A}_{pp}\end{bmatrix} \times \begin{bmatrix}\mathbf{x}_x \\ \mathbf{x}_p\end{bmatrix} = \begin{bmatrix}\mathbf{q}_x \\ \mathbf{q}_p\end{bmatrix} = \mathcal{P}\mathbf{q} \tag{9.1}$$

where $\mathbf{A}_{xx}$ is a $\mathbb{R}^{N_x} \times \mathbb{R}^{N_x}$ matrix composed of equations regarding the N_x unknowns selected, $\mathbf{A}_{pp}$ is a $\mathbb{R}^{N_p} \times \mathbb{R}^{N_p}$ matrix composed of equation regarding the equations which N_p variables are considered known according to a given criterion, and $\mathbf{A}_{xp} \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_p}$ and $\mathbf{A}_{px} \in \mathbb{R}^{N_p} \times \mathbb{R}^{N_x}$ are the matrices representing the couplings between known and unknown variables. $\mathbf{x}_x \in \mathbb{R}^{N_x}$ and $\mathbf{x}_p \in \mathbb{R}^{N_p}$ are, respectively, the known and unknown vectors. Also, $\mathbf{q}_x \in \mathbb{R}^{N_x}$ and $\mathbf{q}_p \in \mathbb{R}^{N_p}$ are, respectively, the knowns and unknowns right-hand-side vectors.

Assuming certain variables are known enables us to remove these equations from the system (9.1). The strong assumption is that there is no coupling between known and unknown variables. Hence, under the assumption that $\mathbf{A}_{xp} = \mathbf{0}$, $\mathbf{A}_{px} = \mathbf{0}$ and $\mathbf{A}_{pp} = \mathbf{I}$, from (9.1), it follows that

$$\mathbf{A}_{xx}\mathbf{x}_x = \mathbf{q}_x, \tag{9.2}$$

is the system that should be iterated over in the i-MSFV smoothing stage.

Iterative Multiscale
For the iterative multiscale model the same formulation holds, but instead we deal with known and unknown corrections $\delta\mathbf{x}_x, \delta\mathbf{x}_p$ and their corresponding residuals $\mathbf{r}_x, \mathbf{r}_p$ at different iteration levels.
Note that, because $\mathcal{P}^\nu$ can theoretically be reconstructed at every iteration ν, the size of the operators in (9.1) can also change. Also, because of this dynamic behaviour, the convergence criterion must be carefully defined. For instance, having a norm-2 of the reduced residual after smoothing the reduced system of equations does not imply that the norm-2 of the full system will be also smaller than the tolerance.

9.2.2. Residual-based Convergence Criterion

A logical criterion for the determination of the reduced smoothing system is based on the fine-scale residual. If $\mathbf{r}_i < \epsilon, i = 1 \cdots N_F$, ϵ being the i-MSFV tolerance, implies that fine-scale mass conservation is respected given ϵ sufficiently small. Consequentially, that also implies that $\delta\mathbf{x}_i < \epsilon, i = 1 \cdots N_F$. That allows a rigorous enough criterion so that (9.2) holds.
Therefore $\mathcal{P}$ can be built such that if $\mathbf{r}_{p_i} < \epsilon, \delta\mathbf{x}_i \in \delta\mathbf{x}_p$, otherwise $\delta\mathbf{x}_i \in \delta\mathbf{x}_x$.
The convergence is controlled based on the norm-2 of $\mathbf{r}_x^{\nu-1}$.

9.2.3. Goal Oriented Adjoint-based Convergence Criterion

Fine Scale
Suppose that the final state $\mathbf{x}$ is used to evaluate a quadratic or integral goal $\mathcal{J}(\mathbf{x})$. Examples are Net Present Value, or Misfit functions. Depending on the type of goal imposed on the problem, one can formulate a minimization or maximization problem to find the optimal goal value where the flow discretized system of equations is used as constraint. At a fine scale this gives the problem

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{J}(\mathbf{x}) \\ &\text{subject to} \quad \mathbf{g}_F(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{A}(\boldsymbol{\theta})\mathbf{x} - \mathbf{q}(\boldsymbol{\theta}) = \mathbf{0} \end{aligned} \tag{9.3}$$

Problem (9.3) can by solved using the Adjoint Method [3, 7, 19].

The Adjoint Model has already been defined in Chapter [6]. This method produces so-called Adjoint Variables or Lagrange Multipliers. Futhermore, this method enables us to write the gradient of $\mathcal{J}(\mathbf{x})$ with respect to the parameters as

$$\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}} = \boldsymbol{\lambda}^\top \frac{\partial \mathbf{g}_F}{\partial \boldsymbol{\theta}} \tag{9.4}$$

The Adjoint Variables, or Lagrange Multipliers $\boldsymbol{\lambda}$ is a vector of size $N_F \times 1$, and may be interpreted as a measure of the effect of perturbing the model equations, and thus the state, on the goal $\mathcal{J}(\mathbf{x})$. The matrix $\frac{\partial \mathbf{g}_F}{\partial \boldsymbol{\theta}}$ of size $N_F \times N_{\boldsymbol{\theta}}$ gives for each spatial coordinate the rate of change in model equations when the parameters of the systems are perturbed.

Equation (9.4) is equivalent to

$$\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}} = \boldsymbol{\lambda}^\top \frac{\partial \mathbf{g}_F}{\partial \boldsymbol{\theta}} = \left(\boldsymbol{\lambda} \cdot \frac{\partial \mathbf{g}_F}{\partial \boldsymbol{\theta}_1}, \cdots, \boldsymbol{\lambda} \cdot \frac{\partial \mathbf{g}_F}{\partial \boldsymbol{\theta}_{N_{\boldsymbol{\theta}}}} \right) = \left(\langle \boldsymbol{\lambda}, \frac{\partial \mathbf{g}_F}{\partial \boldsymbol{\theta}_1} \rangle, \cdots, \langle \boldsymbol{\lambda}, \frac{\partial \mathbf{g}_F}{\partial \boldsymbol{\theta}_{N_{\boldsymbol{\theta}}}} \rangle \right), \tag{9.5}$$

hence,

$$\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}} = \left(\sum_{i=1}^{N_F} \boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \boldsymbol{\theta}_1}, \cdots, \sum_{i=1}^{N_F} \boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \boldsymbol{\theta}_{N_{\boldsymbol{\theta}}}} \right) \tag{9.6}$$

From (9.6), we see that for all parameters, the gradient with respect to the parameter can be written as an summation of the product between the effect to the model equations when perturbing parameters and the effect of change in the model equations to the posed goal.

For each degree of freedom $\mathbf{x}_i$ and parameter $\boldsymbol{\theta}_j$, we have the product

$$\boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \boldsymbol{\theta}_j}, \tag{9.7}$$

and, following the interpretation, this gives a measure of perturbing parameter $\boldsymbol{\theta}_j$ for variable $\mathbf{x}_i$ to our posed goal. This measure is called the sensitivity as high values of this measure indicate that perturbing this parameter at this degree of freedom has a high effect to our posed goal. If this measure is low, it has a low effect to our goal.

If there is only one parameter θ, the Goal Oriented Criteria is simple. After evaluation of equation (9.7) for all degrees of freedom, the sensitivity region is defined as all of the positions for which the sensitivity value is above a pre-set tolerance m_{goal}. In this sense one builds the matrix $\mathcal{P}$, by determining for which i one has that $||\boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \theta}||_2 > m_{goal}$. If $||\boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \theta}||_2 > m_{goal}, \mathbf{x}_i \in \mathbf{x}_x$, else $\mathbf{x}_i \in \mathbf{x}_p$.

If however, as often is the case in reservoir simulation, the number of parameters is greater then one, the Goal Oriented Criteria is slightly less trivial. For one parameter, equation (9.7) enables us to visualize the sensitivity of the parameter across the entire domain of our model. If there are more parameters, we conclude that for each parameter we can perform this visualization. However, this means that for all parameters $N_{\boldsymbol{\theta}}$ we get a different pattern for the sensitivity effect in the domain.

Therefore, the strategy for a single parameter doesn't hold as for different parameters we may find different regions of high sensitivity. The question that remains is how we dilute one sensitivity region out of these different patterns?

Naturally, if

$$||\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}_i}|| < ||\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}_j}||$$

for two given parameters, we want to give the sensitivity region for parameter j more importance. Therefore, one could argue that the gradient could serve as a weighting. The weight of parameter $\boldsymbol{\theta}_j$ is given by

$$\mathcal{W}_j = \frac{\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}_j}}{||\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}}||_2} = \frac{\sum_{i=1}^{N_F} \boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \boldsymbol{\theta}_j}}{||\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}}||_2}, \tag{9.8}$$

and can be interpreted as the importance of parameter $\boldsymbol{\theta}_j$ with respect to the other variables. For all variables $\mathbf{x}_i$, the total sensitivity therefore may be written as

$$\sum_{j=1}^{N_{\boldsymbol{\theta}}} \boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \boldsymbol{\theta}_j} \mathcal{W}_j \tag{9.9}$$

We note that by introducing this weighting the total sensitivity can be thought of as stacking all the relative sensitivities at that degree of freedom. We denote

$$\mathcal{S} = \left(\sum_{j=1}^{N_{\boldsymbol{\theta}}} \boldsymbol{\lambda}_1 \frac{\partial \mathbf{g}_{F1}}{\partial \boldsymbol{\theta}_j} \left(\frac{\sum_{i=1}^{N_F} \boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \boldsymbol{\theta}_j}}{||\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}}||_2} \right), \cdots, \sum_{j=1}^{N_{\boldsymbol{\theta}}} \boldsymbol{\lambda}_{N_F} \frac{\partial \mathbf{g}_{FN_F}}{\partial \boldsymbol{\theta}_j} \left(\frac{\sum_{i=1}^{N_F} \boldsymbol{\lambda}_i \frac{\partial \mathbf{g}_{Fi}}{\partial \boldsymbol{\theta}_j}}{||\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}}||_2} \right) \right)^{\mathsf{T}} \tag{9.10}$$

as the vector of spatial sensitivities. As an effect, we have the total sensitivity of all spatial positions. If the value is high, it means that perturbations in one or more parameters has a big effect. In contrast, if the value is low, perturbing the parameters has a little effect to our goal. We may therefore define our criteria based upon $\mathcal{S}$. However, in order to do so, one must first apply a scaling of this matrix. This is because the results of the forward model may vary in orders of magnitude if different settings are applied, so that it is impossible to find a value-based criteria that is generically applicable. Scaling removes these setting dependent differences in magnitude of the solution and gives a possibility to

Algorithm 5: Scaling of Matrix using the Min-Max

input : $\mathcal{S}$
output: Normalized product $\hat{\mathcal{S}}$
Find $MinV = \min \mathcal{S}_{.,i}, \quad \forall i \in \{1, \cdots N_F\}$;
Find $MaxV = \max \mathcal{S}_{.,i}, \quad \forall i \in \{1, \cdots N_F\}$;
if $||MinV|| > MaxV$ **then**
    Set $R_{max} = ||MinV||$;
    Set $R_{min} = MinV$;
else
    Set $R_{max} = MaxV$;
    Set $R_{min} = -MaxV$;
Calculate $\hat{\mathcal{S}} = \frac{2\mathcal{S}}{R_{max} - R_{min}}$

define a measure.

The scaling procedure is found in algorithm [5]. Note that this algorithm takes scales all values according the highest and lowest, whilst still maintaining the correct spread between all values.

Independent from the settings, this scaled vector $\hat{\mathcal{S}}$ may now be used to form the criterion for the determination of the reduced smoothing system.

$$\text{If } |\hat{\mathcal{S}}_i| > m_{goal}, \mathbf{x}_i \in \mathbf{x}_x, \text{ else }, \mathbf{x}_i \in \mathbf{x}_p \tag{9.11}$$

Multiscale
The Goal Oriented Adjoint-Based Convergence Criteria as previously determined, is based upon a fine-scale model. Hence, for complex models, this leads to expensive computations. To resolve this problem, we try to reduce this complexity by employing an Multiscale framework. Here we use the model formulation as in [25],

$$\mathbf{g} = \begin{bmatrix} \breve{\mathbf{g}} \\ \mathbf{g}' \end{bmatrix} = \mathbf{0}, \quad \mathbf{x} = \begin{bmatrix} \breve{\mathbf{x}} \\ \mathbf{x}' \end{bmatrix}, \tag{9.12}$$

where $\mathbf{g} \in \mathbb{R}^{N_C N_F}, \mathbf{x} \in \mathbb{R}^{N_C N_F}$. To derive the Adjoint Model, we maintain a similar strategy. For the Lagrange Multiplier method this leads to an augmented Lagrangian of

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathcal{J}(\mathbf{x}) + \begin{pmatrix} \breve{\boldsymbol{\lambda}}^\top & \boldsymbol{\lambda}'^\top \end{pmatrix} \mathbf{g}, \tag{9.13}$$

where our goal is evaluated only on fine scale level, hence $\frac{\partial \mathcal{J}(\mathbf{x})}{\partial \breve{\mathbf{x}}} = 0$. The Lagrange Multipliers is a structured vector, with $\boldsymbol{\lambda} \in \mathbb{R}^{N_C N_F}$. We are now able to formulate the multiscale equivalent of equation as

$$\begin{pmatrix} \frac{\partial \mathcal{J}(\mathbf{x})}{\partial \breve{\mathbf{x}}} & \frac{\partial \mathcal{J}(\mathbf{x})}{\partial \mathbf{x}'} \end{pmatrix} + \begin{pmatrix} \breve{\boldsymbol{\lambda}}^\top & \boldsymbol{\lambda}'^\top \end{pmatrix} \begin{pmatrix} \breve{\mathbf{A}} & \mathbf{0} \\ -\mathbf{P} & \mathbf{I} \end{pmatrix} = \mathbf{0} \tag{9.14}$$

Rearranging the terms, enables us to find the following expressions

$$\breve{\boldsymbol{\lambda}}^\top = \boldsymbol{\lambda}'^\top \mathbf{P} \breve{\mathbf{A}}^{-1}, \quad \boldsymbol{\lambda}'^\top = -\frac{d\mathcal{J}(\mathbf{x})}{\partial \mathbf{x}'} \tag{9.15}$$

Similarly, the gradient of $\mathcal{J}$ with respect to the may be written as

$$\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}} = \boldsymbol{\lambda}^\top \frac{\partial \mathbf{g}(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \breve{\boldsymbol{\lambda}}^\top \frac{\partial \mathbf{g}(\breve{\mathbf{x}}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda}'^\top \frac{\partial \mathbf{g}(\mathbf{x}', \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \tag{9.16}$$

where, if we apply the same decomposition as in (9.5), we find a relation comparable to (9.6),

$$\frac{d\mathcal{J}(\mathbf{x})}{d\boldsymbol{\theta}} = \left(\sum_{i=1}^{N_C} \breve{\boldsymbol{\lambda}}_i \frac{\partial \breve{\mathbf{g}}_i}{\partial \boldsymbol{\theta}_1} + \sum_{j=1}^{N_F} \boldsymbol{\lambda}'_j \frac{\partial \mathbf{g}'_j}{\partial \boldsymbol{\theta}_1}, \cdots, \sum_{i=1}^{N_C} \breve{\boldsymbol{\lambda}}_i \frac{\partial \breve{\mathbf{g}}_i}{\partial \boldsymbol{\theta}_{N_{\boldsymbol{\theta}}}} + \sum_{j=1}^{N_F} \boldsymbol{\lambda}'_j \frac{\partial \mathbf{g}'_j}{\partial \boldsymbol{\theta}_{N_{\boldsymbol{\theta}}}} \right), \tag{9.17}$$

where the final term in expression (9.17) gives the total sensitivity for each parameter on fine scale. Hence, for each parameter we can split the sensitivity into a spatial-coarse sensitivity $\check{\mathcal{S}}$ and a spatial fine sensitivity $\mathcal{S}'$. In a similar manner as in (9.9), we may write the coarse sensitivity as

$$\check{\mathcal{S}}_n = \sum_{i=1}^{N_{\boldsymbol{\theta}}} \check{\boldsymbol{\lambda}}_i \frac{\partial \check{\mathbf{g}}_i}{\partial \boldsymbol{\theta}_n} \check{\mathcal{W}}_n, \quad \check{\mathcal{W}}_n = \frac{\sum_{i=1}^{N_C} \check{\boldsymbol{\lambda}}_i \frac{\partial \check{\mathbf{g}}_i}{\partial \boldsymbol{\theta}_n}}{||\check{\boldsymbol{\lambda}}^\top \frac{\partial \check{\mathbf{g}}}{\partial \boldsymbol{\theta}}||_2} \tag{9.18}$$

Similarly, the fine-scale sensitivity may be expressed as

$$\mathcal{S}'_n = \sum_{j=1}^{N_{\boldsymbol{\theta}}} \boldsymbol{\lambda}'_j \frac{\partial \mathbf{g}'_j}{\partial \boldsymbol{\theta}_n} \mathcal{W}'_n, \quad \mathcal{W}'_n = \frac{\sum_{j=1}^{N_F} \boldsymbol{\lambda}'_j \frac{\partial \mathbf{g}'_j}{\partial \boldsymbol{\theta}_n}}{||\boldsymbol{\lambda}' \frac{\partial \mathbf{g}'}{\partial \boldsymbol{\theta}}||_2}, \tag{9.19}$$

so that the total sensitivity is given by

$$\mathcal{S}_n = \check{\mathcal{S}}_n + \mathcal{S}'_n \tag{9.20}$$

After normalization as in algorithm (5), the Goal Oriented Adjoint-Based Criterion will be to select all positions $\mathbf{x}'_i$ for which $|\hat{\mathcal{S}}_i| > m_{goal}, \quad i = 1, \cdots, N_F$. Therefore $\mathcal{P}$ in (9.1) can build such that if

$$|\hat{\mathcal{S}}_i| > m_{goal}, \mathbf{x}'_i \in \mathbf{x}_x, \text{ else } \mathbf{x}'_i \in \mathbf{x}_p$$

.

9.2.4. Algorithm

In this section we will present the algorithm used to implement the Goal-oriented adjoint based criterion. We write m_{goal} to denote the goal tolerance. We note that, in our algorithm, we deal with the known positions $\mathbf{x}_p$ as if the solution at these positions is considered to be good enough. Hence, we deal with these positions as if they are Dirichlet Boundary conditions. However, in order to create a valid system of equations, we need to adapt the residual and system matrix accordingly. The Dirichlet condition is $\mathbf{r}_p = \mathbf{0}$. Then, for the system matrix we adapt the entries as follows. For simplicity, assume that there is exactly one known position $\mathbf{x}_p$, which in the unpermuted system corresponds to spatial position $\mathbf{x}_i$. The original system can be represented in general form as

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1i} & \cdots & a_{1N_F} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2i} & \cdots & a_{2N_F} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \cdots & a_{ii} & \cdots & a_{iN_F} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ a_{N_F1} & a_{N_F2} & a_{N_F3} & \cdots & a_{N_fi} & \cdots & a_{N_FN_F} \end{pmatrix}, \tag{9.21}$$

and we incorporate the Dirichlet conditions by adapting the matrix to

$$\bar{\mathbf{A}} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1i} & \cdots & a_{1N_F} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2i} & \cdots & a_{2N_F} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ a_{N_F1} & a_{N_F2} & a_{N_F3} & \cdots & a_{N_Fi} & \cdots & a_{N_FN_F} \end{pmatrix}, \quad \bar{\mathbf{r}}_p = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ 0 \\ \vdots \\ \mathbf{r}_{N_F} \end{pmatrix} \tag{9.22}$$

If there are more positions, then we repeat this procedure for all corresponding spatial positions. Finally, there after we use the system

$$\bar{\mathbf{A}}\mathbf{x} = \bar{\mathbf{r}}_p$$

We also note that by not necessarily selecting all positions the stopping criteria of the algorithm is adjusted too. Instead of using a stopping criteria based on the total residual $\mathbf{r}^\nu$, we define our system to have been converged if the residual belonging to the positions that are smoothed is below a certain tolerance. The process can be seen in algorithm [6].

Algorithm 6: Algorithm for Goal - Oriented iterative Multi Scale Method

input : $\mathbf{A}$, $\mathbf{q}$, m_{tol}, m_{goal}
output: The state of the system $\mathbf{x}$

Set $\mathbf{r}^0 = \tilde{\mathbf{r}}^0 = \mathbf{q}$;
Set $\nu = 1$;
Set $\mathbf{x} = \mathbf{0}$;
Construct Sensitivity Matrix $\mathcal{S}$;
Calculate Permutation Matrix $\mathcal{P}$;
while $||\tilde{\mathbf{r}}^{\nu}||_2 > m_{tol}$ **do**
    Solve $\mathbf{A}\check{\mathbf{x}}^{\nu} = \check{\mathbf{r}}^{\nu-1}$;
    Compute $\mathbf{x}' = \mathbf{P}\check{\mathbf{x}}^{\nu}$;
    Set $\mathbf{x} = \mathbf{x} + \mathbf{x}'^{\nu}$;
    Set $\mathbf{r}^{\nu} = \mathbf{r}^{\nu} - \mathbf{A}\mathbf{x}'^{\nu}$;
    if $m_{goal} \neq 1$ **then**
        Permute matrix $\tilde{\mathbf{A}} = (\mathcal{P}\mathbf{A}\mathcal{P}^{\top})(\mathcal{P}\mathbf{x}'^{\nu})$;
        Permute residual $\tilde{\mathbf{r}}^{\nu} = \mathcal{P}\mathbf{x}'^{\nu}$;
        Smooth and solve for $\tilde{\mathbf{A}}_{\mathbf{xx}}\tilde{\mathbf{x}}^{\nu}_{\sigma\mathbf{x}} = \tilde{\mathbf{r}}$;
        Set $\tilde{\mathbf{r}}^{\nu} = \tilde{\mathbf{r}}^{\nu} - \tilde{\mathbf{A}}\mathbf{x}^{\nu}_{\sigma}$;
        Permute $\mathbf{r}^{\nu} = \mathbf{r}^{\nu} - \mathcal{P}^{\top}\tilde{\mathbf{r}}$ and $\mathbf{x}^{\nu}_{\sigma\mathbf{x}} = \mathcal{P}^{\top}\mathbf{x}^{\nu}_{\sigma}$;
        Set $\mathbf{x} = \mathbf{x} + \mathbf{x}^{\nu}_{\sigma}$;
        Set $\mathbf{r}^{\nu} = \mathbf{r}^{\nu} - \mathbf{A}\mathbf{x}^{\nu}_{\sigma}$;
        Compute $\bar{\mathbf{A}}$ and set $\mathbf{A} = \bar{\mathbf{A}}$;
    else
        Set $\tilde{\mathbf{r}}^{\nu} = \mathbf{0}$;
    $\nu = \nu + 1$;
Return $\mathbf{x}$;

If we take a close look at this algorithm, we see that $m_{goal} = 1$ is equivalent with the MS method. This is because by construction all positions are to be considered as known, hence no smoothing is applied. In similar fashion we have that $m_{goal} = 0$ corresponds to the iMSFV method, because now all positions are considered to be unknown, hence smoothing is applied across the entire domain.

10

Numerical iterative Multiscale Goal Oriented Experiments

10.1. Introduction

In this section we perform numerous numerical experiments to test the robustness and computational efficiency of the iterative Multiscale Finite Volume Goal Oriented Adjoint-Based criteria methods (iMS-FVGoal, or short iMSGo). The main goal is to show to computational gain when using the iMSGo(m_{goal}) method compared to the original iMS method. The second objective is to show that this method is also capable of producing gradients accurate enough for optimization purposes. Firstly, we will perform experiments which use the fine scale sensitivity region. Finally we will give a short proof of concept of the iMSGo(m_{goal}) method, where the sensitivity matrix is obtained in a multiscale framework. During this section we will employ a misfit objective function without regularization term as our to be minimized goal $\mathcal{J}(\mathbf{h}(\boldsymbol{\theta}))$. Hence,

$$\mathcal{J}(\mathbf{h}(\boldsymbol{\theta})) = \frac{1}{2}(\mathbf{h}(\mathbf{x},\boldsymbol{\theta}) - d_{obs})^{\top} C^{-1}(\mathbf{h}(\mathbf{x},\boldsymbol{\theta}) - d_{obs}), \tag{10.1}$$

where we assume that the observed quantities d_{obs} are observed only on fine scale. In all experiment we have grid-block dimensions of $\Delta x = \Delta y = \Delta z = 1m$. Furthermore we have a fluid viscosity of 1.0×10^{-3} Pa s. At the injection wells there is a constant pressure of $p_{inj} = 1.0$ Pa, and at the production well we have $p_{prod} = 0.0$ Pa.

In this thesis we will perform test cases that correspond to the previous test cases in chapter [8].

10.2. Fine scale sensitivity region numerical tests

In this section we test the iMSGo(m_{goal}) where the sensitivity matrix $\mathcal{S}$ is constructed using a fine-scale computation. This means however, that before we can run the iterative goal method, we must first solve the fine scale model. We don't consider this to be an issue as an optimziation algorithm is an iterative process that requires many forward simulations. Hence, the cost of running one full fine scale forward simulation is compensated by the fact that all other forward simulations will benefit from the more efficient strategy here introduced. Firstly, we perform the three different categories of experiment to show the computational efficiency of the method.

10.2.1. Investigation of the iterative multiscale goal gradient accuracy

In this section we validate the results for the i-MSGo(m_{goal}) method. Since, previous results from chapter [8] have validated the results of the MSFV gradient computation method and the i-MSFV gradient computation method, we are able to use these results to verify the validity of our newly introduces method to also compute gradients. Because theoretically, $m_{goal} = 0$ corresponds to the i-MS method and $m_{goal} = 1$ corresponds to the MSFV method as explained in the previous chapter, the requirement for the method being valid is that the gradients calculated with the different methods should be

comparable. Note that if the gradients are equal, the pressure solutions must also be the equal.

This investigation is carried out using the two-dimensional homogeneous test case as presented in section [8.2]. We have a fine grid of 21×21 grid blocks. We employ a 7×7 coarsening ratio, giving a 3×3 coarse grid. For this experiment we employ a quarter five spot well setting. We check the difference between the gradients in a numerical and analytical setting.

Note that we use a single outer iteration. In the analytical experiments we use $\epsilon = 1e-6, \epsilon = 1e-1$. We use a very high smoothing tolerance of $\epsilon_\sigma = 5 \times 10^{-18}$ to ensure that the numerical gradient method produces accurate enough gradients. The results of this experiment are found in table [10.1]. From this we may conclude that indeed, the MS and iMS method are equivalent to the iMSGo(m_{goal}) method with goal tolerances 1 and 0 respectively, since the differences between the methods are all zero. Also, we conclude that the pressure solutions generated with the iMSGo(m_{goal}) method are equal to the MS and iMS method.

Gradient Comparison		
	$\|\|\nabla_{\boldsymbol{\theta}} O_{MS} - \nabla_{\boldsymbol{\theta}} O_{iMSGo(1)}\|\|_2$	$\|\|\nabla_{\boldsymbol{\theta}} O_{iMS} - \nabla_{\boldsymbol{\theta}} O_{iMSGo(0)}\|\|_2$
Numerical	0	0
Analytical	0	0

Table 10.1: Difference Norm for different methods

10.2.2. Validation of Sensitivity Region

In this section we validate the computation of the Lagrange Multipliers and the sensitivity region. For illustration we consider once more the two-dimensional test case as in section [8.2], but now two twin experiments with different reference fields are used. One is a homogeneous reference field and the second is heterogeneous.

We note that because we are using an goal function as in equation (10.1), the algebraic formulation of the Lagrange Multipliers can be written as

$$\boldsymbol{\lambda}^{\mathsf{T}} = -\mathbf{W} \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \mathbf{A}^{-1}, \tag{10.2}$$

where $\mathbf{W}$ indicates the misfit at the observation wells. We can see that the Lagrange Multipliers are directly dependent on the misfit , i.e. if the misfit is large, then so will be the Lagrange Multipliers. For the two-dimensional homogeneous test case, with the homogeneous twin experiment, the pressure solution will be symmetric and the misfit will therefore be the same for all the observation wells. This also means, that from equation (10.2) we have that the Lagrange multipliers are symmetric along the $x-$ and $y-$axis. Then, also the sensitivity region must show a symmetric behaviour. Figure [10.1] shows, that this is indeed the case. To build the sensitivity matrix as in equation (9.11) we use $m_{goal} = 1e-2$ purely for illustrative reasons. For the twin experiment with the heterogeneous reference field, we can see that the Lagrange Multipliers indeed are related to the misfit at these positions. Furthermore, we clearly see that the sensitivity region is not symmetric and that the sensitivity region selects more positions in the neighbourhood of the largest misfit. This can be explained easily, as the largest misfit leads to high values of our posed goal, we want to accurately calculate the solution near this location so that we can minimize this effect. Therefore, more positions in this neighbourhood are marked as sensitive. These results confirm the correctness of the computation of the Lagrange Multipliers and the sensitivity regions.

10.2.3. Investigation of the i-MSFVGoal convergence behaviour

In this section we will we test if the iMSGo(m_{goal}) is computationally more efficient, by investigating the total number of smoothing steps required for the convergence to the outer tolerance ϵ. Next to this, the correctness of the method can be illustrated with aid of the difference in pressure solution. We define the pressure error norm to be

$$\gamma = ||\mathbf{x}_F - \mathbf{x}_M||_2, \tag{10.3}$$

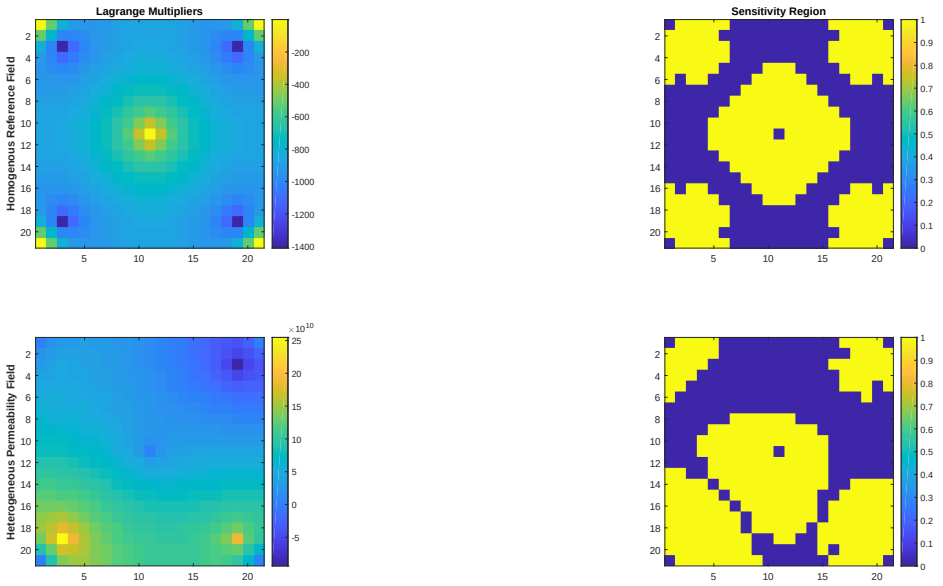


Figure 10.1: The Lagrange Multipliers for the two-dimensional test case with two different twin experiments. Also the Sensitivity regions for $m_{goal} = 1e-2$ are given

where F is the subscript for the fine scale pressure solution and M denotes the method used to approximate the solution. In this thesis we approximate the solution using our newly introduce iMSGo(m_{goal}) method. We investigate the pressure error norm as a function of the goal tolerance. We note however, that the pressure error norm is not evaluated as an criteria for the quality of the method. This is because in the algorithm, we accept that some positions in our domain are not important to our goal and therefore the solution is already good enough. This means that by assumption, the error pressure norm for the iMSGo(m_{goal}) method is not comparable to the iMS error pressure norm. Still, you would like the pressure error norm to decrease and we can use the error pressure norm to validate the correctness of the algorithm as the error pressure should decrease when the goal tolerance m_{goal} decreases as well.

Homogeneous and Heterogeneous distribution tests

First of all we test the behaviour in a two-dimensional homogeneous setting. The full details of the settings of the method have already been described, and we refer the reader to section [8.2] for the full specifications. We investigate the behaviour by adapting the input parameter m_{goal}. These different goal tolerances lead to different sensitivity regions as indicated in Figure [10.2].

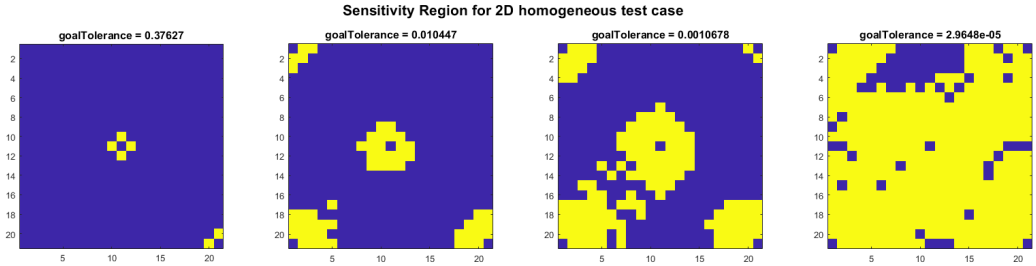


Figure 10.2: Figure indicating four different sensitivity regions

We remark that instead of supplying the algorithm with a goal tolerance m_{goal}, sometimes it is beneficial to supply the algorithm with the total number of positions to be smoothed, say P_s. From this number P_s, one can determine the corresponding goal tolerance by sorting the total sensitivity $\mathcal{S}$ in ascending order. Then, if we select entry $\mathcal{S}(P_s)$, this gives the corresponding goal tolerance. This feature can be important as for each model the sensitivity matrix $\mathcal{S}$ is different, and hence imposing the same goal tolerance m_{goal} may lead to different results. By fixing P_s, we force the algorithm to smooth over the same number of positions and therefore give more reliable results.

The pressure error norm and total number of smoothing iterations for different tolerances is visualized in Figure [10.3]. If the goal tolerance equals 1, the method is equivalent to the MSFV method and has the largest pressure error norm. Gradually increasing the goal tolerance means decreasing the number of positions that we smooth, hence gradually decreases the error norm. The error norm is the smallest when a goal tolerance of 0 is used, because then the entire model is smoothed. From this figure, we can see that the total number of smoothing steps decreases as the goal tolerance increase.

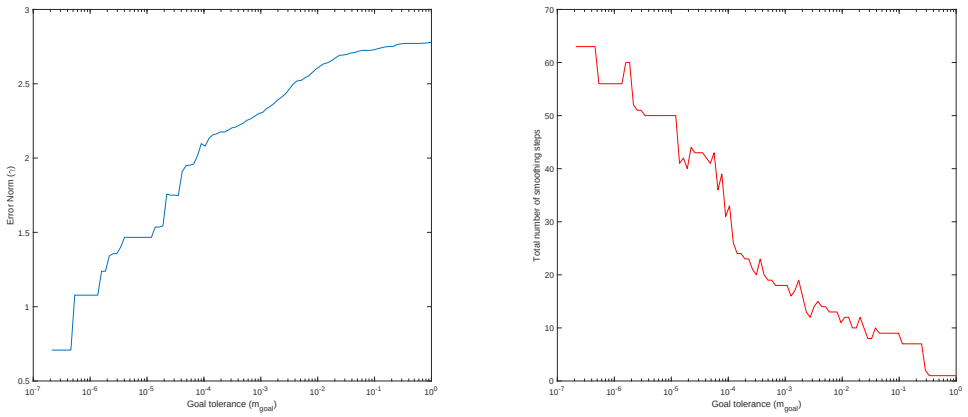


Figure 10.3: Figure indicating the pressure error norm (left) and the total number of smoothing steps (right) as a function of the goal tolerance.

However, we notice that even though that the pressure error norm decreases as the goal tolerance decreases, it decreases at a slower rate and the error remains relatively large. To show that these results are correct, we show the effect of selecting 95% of the domain to be smoothed. Then, we expect the pressure error norm to be very close to the iMS method, because almost the entire domain is smoothed over. Figure [10.4] shows the final pressure solution, the final residual, and the sensitivity region for the iMSGo(m_{goal}) method, where in total 433 positions are selected in the sensitivity matrix $\mathcal{S}$.

Results from Figure [10.4] shows us that the method behaves correctly. Exactly at the positions where we do not smooth the solution, we see the biggest value of the pressure error norm. Furthermore, the

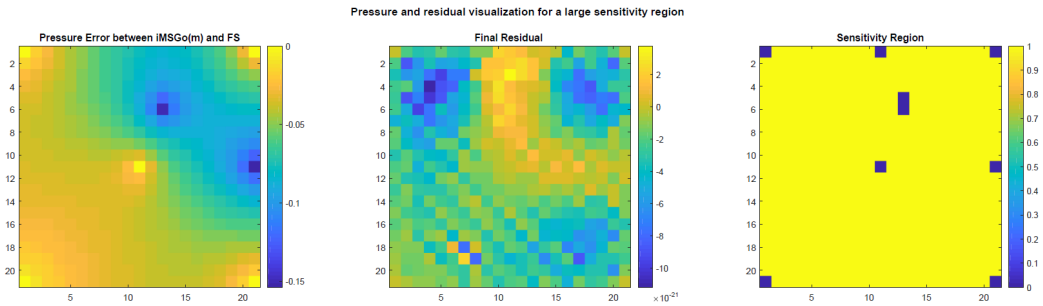


Figure 10.4: Final Pressure solution, Final Residual and Sensitivity Region

residual at these positions is high, which is to be expected as the convergence criteria only looks if the norm of the smoothed positions is below a certain tolerance.

Next, we discuss the heterogeneous realizations to slightly increase the complexity of the problem. The specifications of the model can be found in section [8.3.1]. Figure [10.5] shows us the behaviour with respect to the iterations and the pressure error norm. The results are comparable to the homogeneous test case. The result from the pressure error norm indicate the correctness of the method as gradually the error norm decreases when the goal tolerance is increased. However, it is observed that the decrease of this pressure error norm is slow. For smoothing 92.5% of the complete domain, the error norm decreases on average with only 1% compared to the errorneous MS method. However, increasing the goal tolerances over the 1000 test cases, eventually leads to a rigorous decrease in the amount of smoothing steps. Also we can see that the variance in the convergence behaviour decreases when the goal tolerance larger.

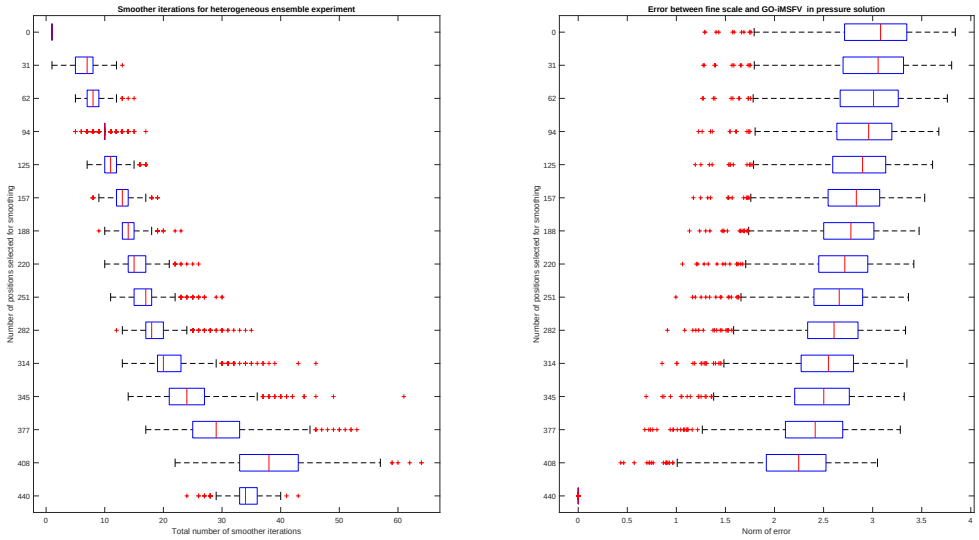


Figure 10.5: Left: Total number of smoothing steps required for convergence as a function of the amongst full ensemble size for different goal tolerances. Right: Pressure Error Norm for different goal tolerances amongst the full ensemble size.

So even though the number of smoothing positions decrease and that we know that the pressure error norm is not comparable to the iMS method, the pressure error norm decreases at much slower rate. This may not necessarily be an issue, as the goal is not affected by the unsmoothed degrees of freedom. However, the degrees of freedom that are used to evaluate our goal also converge at a slow rate. One of the reasons for slow the decrease of the error pressure norm, can be explained by the pressure corrections made at solving the coarse scale system. For illustration we use Figure [10.4]. Remember that in this case, a coarsening ratio of 7×7 is used. Then, it seems as though the basis functions are unable to accurately capture the effect of the non-smoothed positions whilst still improving the other positions. So, to increase the accuracy an decrease of the coarsening ratio may provide basis functions that are able to more accurately make corrections. Therefore we redo the experiments described above, but now we employ a coarsening ratio of 3×3. Figure [10.6] shows the mean number of total smoothing positions and the pressure error norm. In comparison to the 7×7 case, we see that pressure error norm is much smaller.

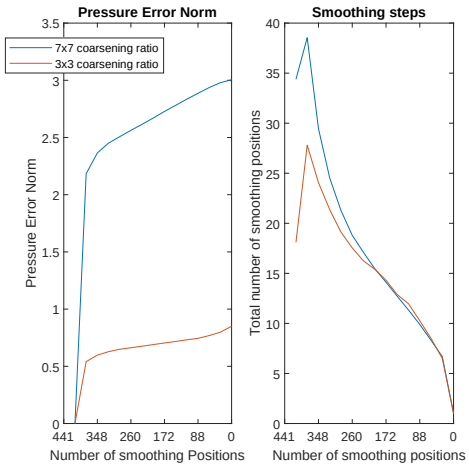


Figure 10.6: Comparison of the pressure error norm and total number of smoothing steps for two different coarsening ratio's.

Investigation of Highly heterogeneous permeability distributions
The next experiments are based upon the highly heterogeneous permeability distributions as in section [8.3.2]. This section also contains the full model specifications. The most important note for the reader, is that the permeability distributions are generated in four sets of 20 realizations each, with patches of high- and low permeability differences in streaks of different angles. We consider angle sets of $0°, 15°, 45°$ and a patchy behaviour set. We use a coarsening ratio of 25×25.

Before we illustrate the solution behaviour of the iMSGo(m_{goal}) method, we calculate the fine scale sensitivity matrix $\mathcal{S}$ for the different sets. This is illustrated in Figure [10.7]. We notice that the sensitivity region aligns with the direction of the flow governed by the heterogeneity.

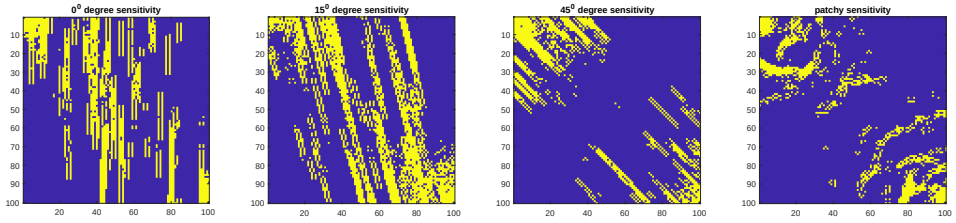


Figure 10.7: Visualization of examples of sensitivity regions for the different angle sets

Figure [10.8] shows the total number of smoothing steps for the four different permeability sets. For computational reasons and readability we only discuss experiments where more then 50% of the domain is smoothed. We notice that for all sets increasing the goal tolerance leads to a rigorous decrease in the number of total smoothing steps. Only for the $45°$ angle set, there is an increase first.

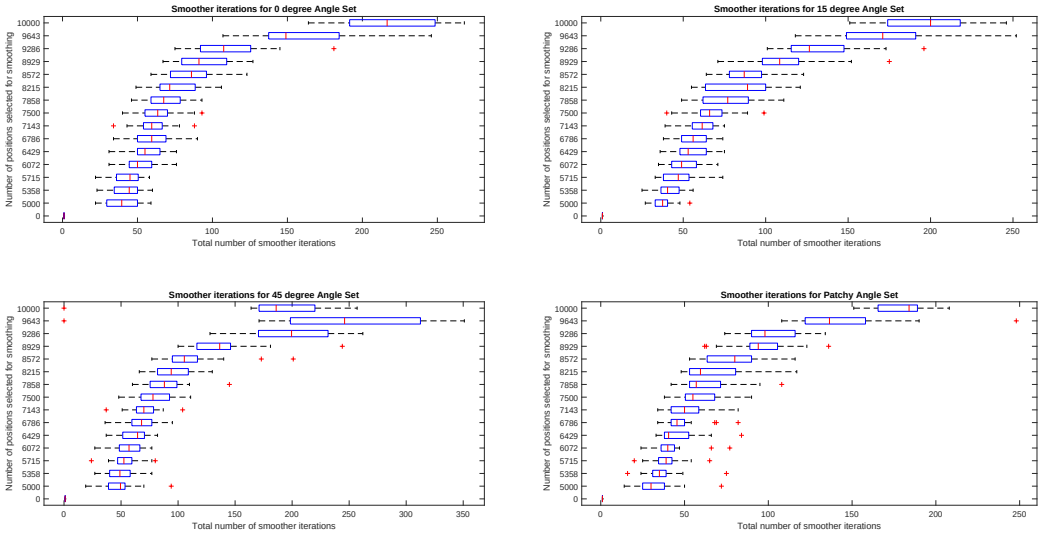


Figure 10.8: Results for the four different angle set realization for smoothing of 50% of the system, or more.

Figure [10.9] shows the relative decrease of the mean number of smoothing steps. It shows us that if we use 80% of the total positions in the smoothing step, the number of total smoothing steps is approximately 30% of the computational effort required in the iMS method. This already proves to be a significant improvement on the computational efficiency.

We also observe that, the pressure error norm performs in a similar way in the previous experiments.

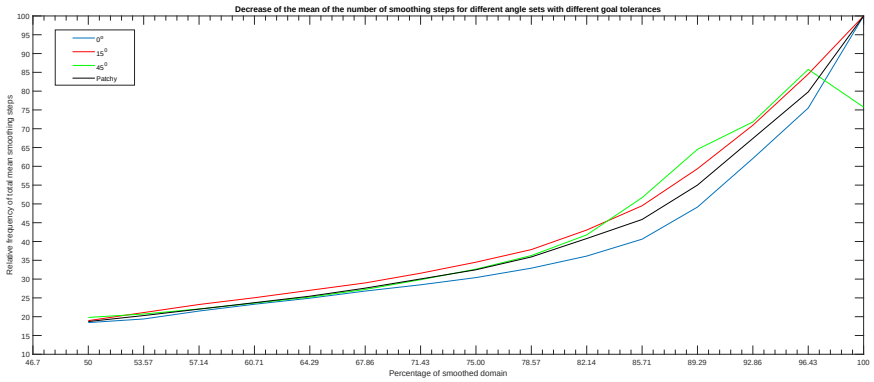
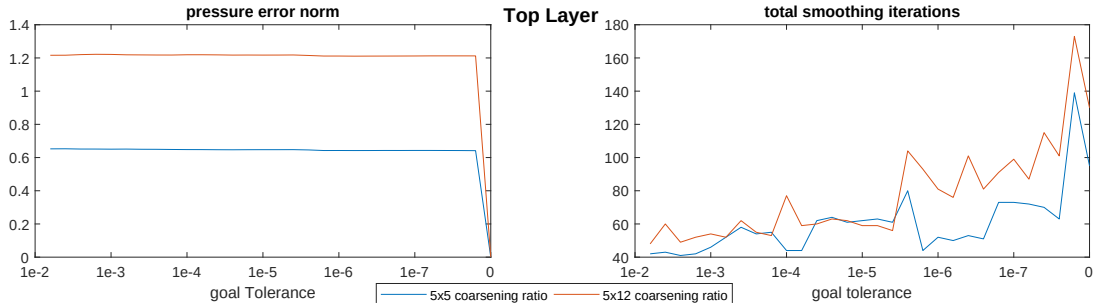


Figure 10.9: The decrease of number of smoothing steps percentage wise as function of the percentage domain used for smoothing in the smoothing step.

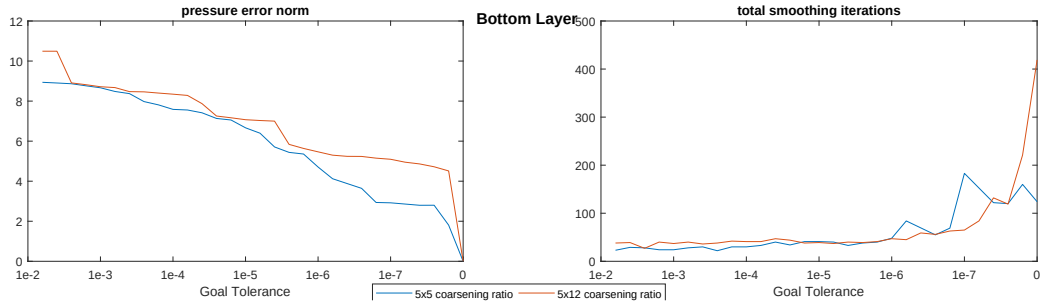
So even though the decrease in computational effort is promising, the pressure error norm decreases at a slow rate. To show that this conclusion holds, we confirm this conclusion also holds for the final, most challenging test case.

SPE-10 Comparative test case

For our final test, we investigate the convergence behaviour of the SPE-10 comparative test case. The full model specifications are already discussed and can be found in section [8.3.3]. Also mentioned in this section is the great complexity of the associated with the very high heterogeneity in the permeability distributions. We consider experiments on two different permeability fields and we use two different coarsening ratio's. One of 5×11, which corresponds to the original setting, and one of 5×5, increasing the coarse grid size by a factor 2.5. We investigate the total number of smoothing steps taken in the algorithm, and the pressure error norm. Figure [10.10] visualizes the results.



(a) Top Layer Results



(b) Bottom Layer Results

Figure 10.10: Results for the SPE-10 comparative test case

These results show us that once again, that the total number of smoothing steps decrease quite rigor-

ously. Therefore we may conclude that using the iterative goal method may be beneficial for computational effort. From the pressure error norm however, we indicate that changing the coarsening ratio does not necessarily improve the quality of the basis functions. This method proves to be promising, but because of the behaviour of the pressure error norm, further investigation needs to be performed of how to increase the accuracy of the solution to prove it's full potential.

10.3. Multi Scale Sensitivity Region Numerical Tests

The previous results where based upon building a Sensitivity Matrix $\mathcal{S}$, for which computations on a fine scale model where required. This heavy computation can be simplified by computing the Sensitivity Matrix in a multiscale framework by means of equation (9.19). We use a multiscale pressure solution to build the finescale sensitivity region. As this is not fully supported by the multiscale formulations as presented, the results presented here can be thought of as a proof of concept. The results are not waterproof but they give a good indication to whether the method proves to be efficient or not.

Firstly, we investigate the sensitivity region for the two-dimensional homogeneous test case. Here we investigate, if the same regions are identified for both methods. We investigate the overlap of the sensitivity regions as a function of number of smoothing positions. Figure [10.11] visualizes the results. To check if these regions align, we look at the mismatch between the regions. Next we determine the percentage match between both regions. These results are found in Figure [10.12].

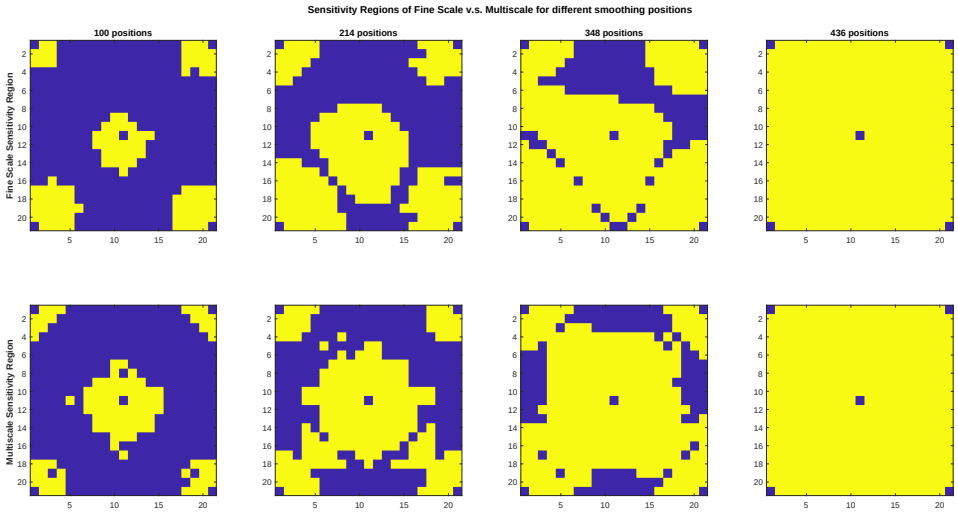


Figure 10.11: Figure of the different sensitivity regions when computed by the fine scale and multiscale model.

From these figures we see that the resemblance in both sensitivity regions is quite well. At minimum there is a 69% overlap between the regions. One can note that for a low number of smoothing positions, up to approximately 50% of the region the overlap is more then 80%. This means, that using a computationally more efficient solving method leads to comparable sensitivity regions. What is also noticeable, and probably causes the lowest percentages for smoothing more then 50%, is that the iMS sensitivity regions are not symmetrical in the sense that the regions expand faster to the wells where the mismatch is the largest. For the iMSGo(m_{goal}), the sensitivity regions show a more symmetric behaviour. Probably, this is because the approximations in the multiscale solution also effect the Lagrange Multipliers that are used.

To complete this proof of concept we investigate the effect when heterogeneity is introduces. We investigate the percentage of overlap for the 2-dimensional heterogeneous test case, where 1000 realisations are used. The investigation is performed for different number of smoothing positions. The results are found in Figure [10.13] and give similar results. For the entire ensemble the average overlap between the sensitivity regions is not lower then 73%. There is, however some variability amongst

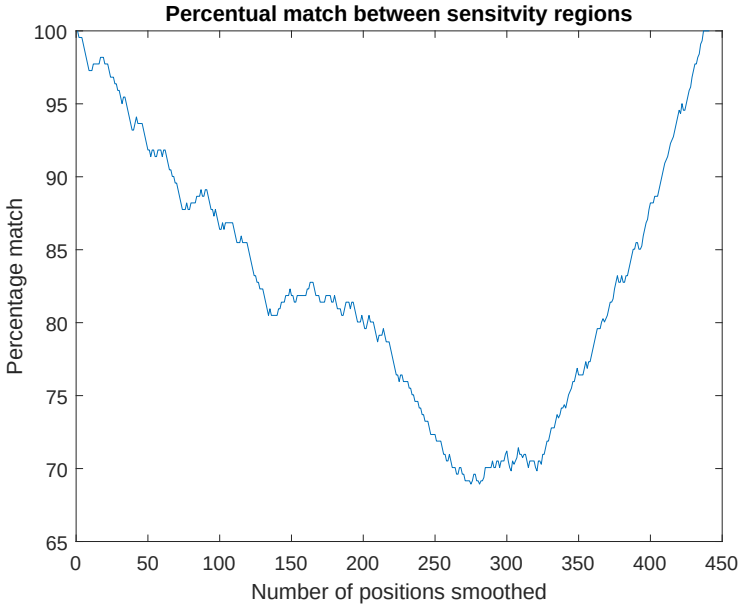


Figure 10.12: Percentage of overlap of the sensitivity regions for different numbers of smoothing positions

the ensemble if around half of the positions are smoothed in the domain. However, again, in the low and high numbers of the smoothing positions the multiscale sensitivity region resembles the fine scale sensitivity region quite well. What remains to be seen is that if using the multiscale sensitivity regions instead of the fine scale sensitivity regions lead to similar results in the iMSGo(m_{goal}) method. For future research one can investigate whether the multiscale regions are accurate enough, or if the positions that do not overlap lead to inaccurate results.

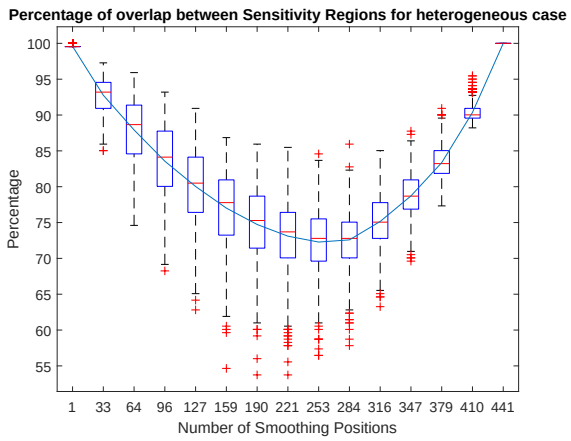


Figure 10.13: Percentage of overlap of the sensitivity regions for different numbers of smoothing positions in the heterogeneous case

11

Conclusion and recommendations

In this chapter we will summarize the results of this thesis and we will draw conclusions based on these results. Finally we will make some recommendations for future work.

Firstly, we investigated how to reformulate the multiscale gradient computation using the Lagrange Multiscale method. The benefit of this method is that we have a clear formulation for the Lagrange Multipliers, which are used in a later stage of this thesis for improving the computational efficiency of the iterative multiscale method. We successfully implemented algorithms that not only compute the gradient, but also save the Lagrange Multipliers for different usages.

Secondly, we have seen that by adapting our multiscale method to a residual based iterative multiscale method we are able to align our gradient estimation to the fine scale gradient with a very high accuracy. This result is validated for both the Adjoint and Direct iterative method. This has been observed for different test cases of increasing complexity. Because of the nature of our data assimilation problem, where the number of parameters is greater than the number of output functionals, we used the Adjoint method in the test cases to show the efficiency and accuracy of the iterative MS gradient computation strategy.

After this, we have shown that the Lagrange Multipliers used in the algorithm for gradient computation are useful to create a sensitivity criteria for the iterative multiscale smoothing step. Results for the homogeneous test case have indicated the validity of the method . Furthermore results show decreasing numbers of smoothing steps in the iterative loop for smaller sensitivity regions, indicating that the convergence behaviour of the newly proposed method is faster then the original method. This is shown by all test cases considered. It is also observed that the method seems to be prone to the quality of the corrections made at the MS stage, and hence the pressure solution may not be as accurate as the original iterative method. A strategy to solve this problem is to use a more refined coarse grid, however results show that this still may not necessarily lead to good enough results for the more challenging test cases when compared to the fine scale solution. We conclude that the method therefore leads to promising results, but that more investigation is required to prove it's full potential. Next to this, we have shown that even though the gradient estimation using this method shows much variability, the results are promising for estimating gradients as well. However, more investigation regarding the gradient computation using the newly introduced strategy is still necessary. Finally, we have shown in a proof of concept that the multiscale sensitivity region resemble the fine scale sensitivity region.

For future work many next steps could be taken. Firstly, further investigation should be performed to find out how the solution of the iMSGo(m_{goal}) method behaves, and how we could increase the accuracy of the solution using the iterative multiscale goal method. We note that the solution errors are associated with the degrees of freedom associated with the regions that have less impact on the objective function. For problems with a smaller sensitivity region, this number is higher and hence, less accurate solutions could be accepted as they do not impact our goal. Thus we pose the question if we can indeed afford to accept less accurate solutions in the non sensitive areas, and how we can

increase the accuracy in the sensitive areas. This investigation could take several directions. One of the solutions we would suggest to investigate in, is to use the Lagrange Multipliers, or the sensitivity region, to define a non-uniform coarsening ratio for the MS-stage as we have seen that corrections made at the multiscale solving stage, are important to the overall accuracy of the solution. This investigation could be beneficial, since decreasing the coarsening ratio over the full domain is not only costly, it is also not necessary if you consider the sensitivity regions. A second solution is to decouple the positions associated to the sensitive region completely from the rest of the positions. However, this requires a bit more work as the original settings posed in multiscale framework need to be redefined to fit the new decoupled system dimensions. It would, however remove the dependency of the solution of smoothing position on the non-smoothed positions.

Next, the adaptation in the code for the computation of the iterative gradient using the newly introduced method should be made, so that we can verify the correctness of the method for estimating gradients. Even though we have validated the gradient computation for the outlier cases, the gradient computation using the iMSGo(m_{goal}) method needs more investigation. In iteratively solving for the pressure, we adapt the the system matrix $\mathbf{A}$ to $\bar{\mathbf{A}}$, where the new matrix deals with the fact that for the non-smoothed positions we accept the solution as it is. Then, when we perform the computations required in the iterative Multiscale Backward Gradient Computation algorithm however, we use the non-adapted system matrix. Obviously, since the solution is build from using the adapted system matrix, one must systematically use this adapted matrix in this algorithm too.

Also, the implementation for computing the multiscale sensitivity region should be finished in order to verify the proof of concept made, leading to another decrease in computational effort.

Even though the goal method is promising, the question remains; what goal tolerance should we use, so that we find the required accuracy, without doing too much work? In order to answer this question, one must know beforehand how accurate you want the solution or gradient to be. Therefore, one must find a a-priori formulation for the accuracy of the gradient. If we know this, we can adapt our system parameters in a way that allows us to control the optimization process fully.

When one is happy with the convergence control of the gradient estimation, model extensions can be considered. Currently, we are considering incompressible flow leading to a time independent problem. Firstly, we would suggest to re-derive the iterative multiscale gradient estimation method for compressible flow, as the time dependecy could change the system behaviour. This would also lead to sensitivity regions in the goal method that are time dependent and it would be interesting to see this effect. One could also consider extending the model to multi-phase flow.

Bibliography

[1] Multiphysics software solutions. URL https://www.comsol.com/.

[2] J.E. Aarnes, V. Kippe, and K.A. Lie. Mixed multiscale finite elements and streamline methods for reservoir simulation of large geomodels. *Advanced Water Resources*, 28, 2005.

[3] Andrew M. Bradley. Pde-constrained optimization and the adjoint method, june 2013. goo.gl/KzaQo2.

[4] M. Augustine Cauchy. *Méthodes générales pour la résolution des systèmes d'équations simultanees*. 1947.

[5] G. Chavent, N. Dupuy, and P. Lemmonier. History matching by use of optimal theory. *SPE*, 15, 1975.

[6] M.A. Christie and M. Blunt. Tenth spe comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Simulation Symposium, Society of Petroleum Engineers*, 2001.

[7] B. N. Davis and R.J. LeVeque. Adjoint methods for guiding adaptive mesh refinement in tsunami modeling. *Pure Applied Geophysics*, 173, 2016.

[8] E. de Klerk, C. de Roos, and T. Terlaky. Nonlinear optimization, february 2006. Course notes Continuous Optimization, Master Math.

[9] Syed Fahad. Basics of mesh generation. URL https://tinyurl.com/y83e74jk.

[10] R. M. Fonseca, S. S. Kahrobaei, L. J. T. Van Gastel, O. Leeuwenburgh, and J. D. Jansen. uanti cation of the impact of ensemble size on the quality of an ensemble gradient using principles of hypothesis testing. *SPE Reservoir Simulation Symposium, Society of Petroleum Engineers*, 2015.

[11] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[12] H. Hajibeygi, G. Bonfigli, and M. Hesse. Iterative multiscale finite-volume method. *Journal Of Computational Physics*, 227(19), 2008.

[13] Hadi Hajibeygi. *Iterative Multiscale Finite Volume Method for Multiphase Flow in Porous Media with Complex Physics*. PhD thesis, Sharif University of Technology, 2011.

[14] infinity Educate. Linked-in slide share, september 2015. URL https://tinyurl.com/yahsylx6.

[15] Jan Dirk Jansen. A simple algorithm to generate small geostatistical ensebles for subsurface flow simulation. uuid:6000459e-a0cb-40d1-843b-81650053e093?collection=research.

[16] Jan Dirk Jansen. *A system Description of Flow Through Porous Media*. Springer, february 2013.

[17] Jan Dirk Jansen. Gradient-based optimization of flow through porous media, september 2015. Draft version published as course notes of Advanced Reservor Simulation (AES1490).

[18] Jan Dirk Jansen. College sheets advance reservoir simulation, porous media and system models, 2017. Course ID: AES1490.

[19] J.F.B.M. Kraaijenvanger. Optimal waterflood design using the adjoint method. *SPE Reservoir Simulation Symposium, Society of Petroleum Engineers*, 2007.

[20] Mats G. Larson and Fredrik Bengzon. *The Finite Element Method: Theory, Implementation, and Applications*. Springer, januari 2013.

[21] John M. Lewis, S. Lakshmivarahan, and Sudarshan Dhall. *Dynamic Data Assimilation, a least square approach*. Cambridge University Press, New York, 2006.

[22] Knut Andreas Lie. Reservoir simulation: From upscaling to multiscale methods, 2007. Seminar on Multiscale Computational Science and Engineering in Trondheim, Norway.

[23] Knut-Andreas Lie. Multiscale methods for reservois simulation, June 2017. Slides for Summer School into Multiscale Methods.

[24] Knut-Andreas Lie, Jostein Roals Natvig, Olav Moyer, and Kyrre Bratvedt. Succesful application of multiscale methods in a real reservor simulator environment, march 2017.

[25] R. Moraes, J. Rodrigues, and H. Hajibeygi. Multiscale gradient computation for flow in heterogeneoud porous media. *Journal of Computational Physics*, 336, 2017.

[26] R.J. Moraes, W. de Zeeuw, H. Hajibeygi, and J.D. Jansen. Iterative multiscale gradient computation with errorcontrol for heterogeneous subsurface flow. *Advances in Water Resources*, 2018. To be submitted.

[27] Board of Royal Dutch Shell. 2016 annual report, 2016. The Annual Report is made publicly by the board members of Shell.

[28] Department of Computer Science Ohio State University and Engineering. Visualize the gradient descent method for high dimension functions. URL `https://tinyurl.com/y8ys3mdx`.

[29] D.S Oliver, A.C Reynolds, and N. Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambdrige University Press, 2008.

[30] Frank Pennekamp. Iterative multiscale adjoint gradient computation for incompressible flow optimization in porous media, 2017.

[31] Inc. Rigzone. Rigzoze, an online resource for news, jobs, data and events for the oil and gas industry. URL `https://www.rigzone.com/`.

[32] J.R.P. Rodrigues. Calculating derivatives for automatic history matching. *Computational Geosciences*, 10, 2006.

[33] Yoshikazu Sawaragi, Hirotaka Nakayama, and Tetsuzo Tanino. Theory of multiobjective optimization. *Mathematics in Science and Engineering*, 176, 1985.

[34] Schlumberger. URL `https://preview.tinyurl.com/y87fvp8d`.

[35] U.S. Energy Information Administration Independent Statistics and Analysis. Oil crude and petroleum product explained, september 2017. goo.gl/Tb2H5u.

[36] Jan Synman. *Practical Mathematical Optimization, An introduction to basic optimization theory and classical and new gradient-based algorithms*. Springer, 2005.

[37] Kambiz Vafai. *Handbook of Porous Media*. CRC Press, 3rd edition.

[38] J. van Kan, A. Segal, and F. Vermolen. *Numerical Methods in Scientific Computation*. Delft Academic Press/VSSD, 2nd edition, 2014.

[39] David A. Vendetti and David L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flow. *Journal of Computational Physics*, 176, 2002.

[40] Ariel Vidal and Rosalind Archer. Geostatistical simulations of geothermal reservoirs: Two-and multiple-point statistic models. *Proceedings World Geothermal Congress*, 2015.

[41] C. Vuik and D.P. Lahaye. Scientific computing, 2017. Course notes of Scientific Computing (WI4201).

[42] J. Wallis and H.A. Tchelepi. Apparatus, method and system for improved reservoir simulation using an algebraic cascading class linear solver, 2010.

[43] Yixuan Wang, Hadi Hajibeygi, and Hamdi A. Tchelepi. Algebraic multiscale solver for porous media. *Journal of Computation Physics*, 259, 2014.

[44] Yixuan Wang, Hadi Hajibeygi, and Hamdi A. Tchelepi. Monotone mutliscale finite volume method. *Computational Geosciences*, 20(3), june 2016.

[45] Roger J.M. De Wiest. *Flow Through Porous Media*. Academic Press Inc. London, 1969.