Pontifícia Universidade Católica
do Rio de Janeiro

Igor Caetano Diniz

Evaluating the use of Random Forest Regressor to Reservoir Simulation in Multi-region Reservoirs

Dissertação de Mestrado

Thesis presented to the Programa de Pós–graduação em Matemática, do Departamento de Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Matemática.

Advisor : Prof. Sinesio Pesco
Co-advisor: Dr. Thiago de Menezes Duarte e Silva

Rio de Janeiro
April 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Igor Caetano Diniz

Evaluating the use of Random Forest Regressor to Reservoir Simulation in Multi-region Reservoirs

Thesis presented to the Programa de Pós–graduação em Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Matemática. Approved by the Examination Committee:

Prof. Sinesio Pesco
Advisor
Departamento de Matemática – PUC-Rio

Dr. Thiago de Menezes Duarte e Silva
Schlumberger Serviços de Petróleo – Matriz

Dr. Eduardo da Silva Castro
Laboratório Nacional de Computação Científica – LNCC

Dr. Rodrigo Gusmão Cavalcante
Cenpes – PETROBRAS

Prof. Hélio Côrtes Vieira Lopes
Departamento de Informática – PUC-Rio

Prof. Abelardo Borges Barreto Jr.
Departamento de Matemática – PUC-Rio

Dr. Renan Vieira Bela
Transportadora Brasileira Gasoduto Bolívia-Brasil

Rio de Janeiro, April the 26th, 2023

Igor Caetano Diniz

Igor is graduated in Bachelor in Mathematics from the Pontifical Catholic University in 2020.

Bibliographic data

Diniz, Igor Caetano

Evaluating the use of Random Forest Regressor to Reservoir Simulation in Multi-region Reservoirs / Igor Caetano Diniz; advisor: Sinesio Pesco; co-advisor: Thiago de Menezes Duarte e Silva. – 2023.

62 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, 2023.

Inclui bibliografia

1. Matemática – Teses. 2. Ajuste de histórico. 3. Quantificação de incertezas. 4. Ensemble smoother com múltipla assimilação de dados. 5. Caracterização de reservatórios. I. Pesco, Sinésio. II. de Menezes Duarte e Silva, Thiago. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. IV. Título.

CDD: 510

Acknowledgments

I would like to acknowledge my advisors, Sinésio and Thiago, for every support and motivation they gave me during these last year of Master Degree. Also, for assisting me to improve my understanding of petroleum engineering. and for the deconcentration moments that certainly helped my understanding of the topics I was studying. Thank you for everything.

I want to extend my thanks to professors that walk with me during my semesters in Mathematics. I was one of fortunate that was supported for everyone in this awesome Department.

To all the people in the secretariat and technical support like Mariana, Creuza, Katia and Carlos. There are no professionals as well qualified as these people and they are certainly ideal for students to feel welcomed, have all the support and peace of mind to study and have their own space. We all felt a sense of belonging there.

To my friends from PUC-Rio, that stayed by my side from the master's degree until the end of the Ph.D. In particular, Daniel De La Riva, Thiago Lucas and Hugo that helped me many times with all things that neeeded since my first graduation days and suggested valuable comments to improve my work.

A special thanks to all professors of Department of Mathematics, because when I came from another university to study at PUC-Rio, they were the first to provide me with full support in all possible areas and I was always able to have good scientific initiations, good monitoring jobs and have the best courses.

Finally, I want to dedicate a special thanks to my mother, Janice Caetano Maia, for everything they have done for me. I do not know if there exist better mother in the world. I am fortunate to have you as my parents.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Abstract

Diniz, Igor Caetano; Pesco, Sinésio (Advisor); de Menezes Duarte e Silva, Thiago (Co-Advisor). **Evaluating the use of Random Forest Regressor to Reservoir Simulation in Multi-region Reservoirs**. Rio de Janeiro, 2023. 62p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Oil and gas reservoir simulation is a common demand in petroleum engineering, and research, which may have a high computational cost, solving a mathematical numeric problem, or high computational time. Moreover, several reservoir characterization methods require multiple iterations, resulting in many simulations to obtain a reasonable characterization. It is also possible to mention ensemble-based methods, such as the ensemble Kalman filter, EnKF, and the Ensemble Smoother With Multiple Data Assimilation, ES-MDA, which demand lots of simulation runs to provide the output result. As a result, reservoir simulation might be a complex subject to deal with when working with reservoir characterization. The use of machine learning has been increasing in the energy industry. It can improve the accuracy of reservoir predictions, optimize production strategies, and many other applications. The complexity and uncertainty of reservoir models pose significant challenges to traditional modeling approaches, making machine learning an attractive solution. Aiming to reduce reservoir simulation's complexities, this work investigates using a machine-learning model as an alternative to conventional simulators. The Random Forest regressor model is experimented with to reproduce pressure response solutions for multi-region radial composite reservoirs. An analytical approach is employed to create the training dataset in the following procedure: the permeability is sorted using a specific distribution, and the output is generated using the analytical solution. Through experimentation and analysis, this work aims to advance our understanding of using machine learning in reservoir simulation for the energy industry.

Keywords

History matching; Uncertainty quantification; Ensemble smoother with multiple data assimilation; Reservoir characterization.

Resumo

Diniz, Igor Caetano; Pesco, Sinésio; de Menezes Duarte e Silva, Thiago. **Avaliando o uso do Algoritmo Random Forest para Simulação em Reservatórios Multi-regiões**. Rio de Janeiro, 2023. 62p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Simulação de reservatórios de óleo e gás é uma demanda comum em engenharia de petróleo e pesquisas relacionadas, que pode requerer um elevado custo computacional de tempo e processamento ao resolver um problema matemático. Além disso, alguns métodos de caracterização de reservatórios necessitam múltiplas iterações, resultando em muitas simulações para obter um resultado. Também podemos citar os métodos baseados em conjunto, tais como o *ensemble Kalman filter*, o EnKF, e o *Ensemble Smoother With Multiple Data Assimilation*,o ES-MDA, que requerem muitas simulações. Em contrapartida, o uso de aprendizado de máquina cresceu bastante na indústria de energia. Isto pode melhorar a acurácia de predição, otimizar estratégias e outros. Visando reduzir as complexidades de simulação de reservatórios, este trabalho investiga o uso de aprendizado de máquina como uma alternativa a simuladores convencionais. O modelo *Random Forest Regressor* é testado para reproduzir respostas de pressão em um reservatório multi-região radial composto. Uma solução analítica é utilizada para gerar o conjunto de treino e teste para o modelo. A partir de experimentação e análise, este trabalho tem o objetivo de suplementar a utilização de aprendizado de máquina na indústria de energia.

Palavras-chave

Ajuste de histórico; Quantificação de incertezas; Ensemble smoother com múltipla assimilação de dados; Caracterização de reservatórios.

Table of contents

List of Figures

1
Introduction

Oil and gas reservoirs are complex geological systems that play a crucial role in the global research and energy market. Over the past few decades, the development of advanced technologies has enabled the exploration and production of these resources from unconventional and conventional sources. The traditional approach to reservoir characterization involves the use of physical measurements, simulations, and statistical models. However, the increasing volume of data generated from these sources has created time and cost-intensive solutions, particularly in production test simulations. Futhermore, reservoir characterization provides crucial information on the distribution and realization of the reservoir heterogeneity and petrophysical properties. Moreover, it is an integral part of fields, e.g., the formation damage assessment and mitigation, because the magnitude and the extent of the reservoir formation damage are greatly influenced by reservoir formation properties.

Accurate reservoir characterization is a critical step in the development, monitoring, and management of a reservoir, as well as in optimizing production. Geehan and Pearce (1994)[9] mentioned that the first goal of reservoir characterization is to create a geological model that is consistent with the available data and can be used to predict the distribution of reservoir formation and fluid parameters. Many reservoir characterization methods are dynamic, i.e., they incorporate data dynamically in time to achieve better accuracy and ensure that all available information at any given time is incorporated into the reservoir model. To accomplish this, however, one must first create a static reservoir model. This model is updated to account for changes in the reservoir as new petrophysical, seismic, or production data as it become available. The updated model would be a more accurate representation of the reservoir's current state. As more field data becomes available, static reservoir properties such as reservoir formation, or facies type, as well as dynamic reservoir properties such as pressure, fluid saturation, or temperature, can be updated.

Marianito and Worthy (2016)[2] showed a mathematical approach to solve multi-layer diffusion problems using the Laplace transform, which is a

technique used to derive the concentration profile and flux for steady-state diffusion in a multi-layer system. The authors demonstrate the use of this method in solving several examples of multi-layer diffusion problems, showing that it is a powerful tool for understanding and predicting diffusion processes in complex systems.

Enterazi et. al (2022) [17] showed that in the energy industry, machine learning alternatives, however, are becoming increasingly popular, not only for characterizing reservoirs, but as a lower cost way out of commercial simulators. For this reason, a real demand in reservoir engineering and research is the cost-effective simulation of oil reservoir. This type of simulation prevents a high computational cost for several reasons and may requires many iterations or simulations to have a reasonable characterization, such as methods via ensemble, such as the Ensemble Smoother with Multiple Data Assimilation, ES-MDA, (2021) [8]. Additionally, machine learning has been increasingly used in the oil and gas industry to improve the accuracy of reservoir predictions and production strategies. The complexity and uncertainty of reservoir models such as geological structures such as folds, faults, joints, sinkholes, and complexities in the domain characterization process (seismic) and in procedures related to measurement pose significant challenges to traditional modeling approaches, making machine learning an attractive solution.

Gonçalves et al. (2022) [6] discussed the use of Random Forest algorithm to forecast daily oil production in reservoir. The authors proposed to predict a one-time step production using the Volve oil field dataset to conduct experiments. The text also discusses the importance of predicting oil field performance in reservoir engineering and the inherited challenges, such as reservoir simulation and history matching. The authors suggest that machine learning techniques may be efficient in some steps of the history matching problem, and they provide examples of previous studies that have used machine learning algorithms in reservoir engineering.

The characterization of reservoirs is currently costly due to several factors: it takes considerable time to implement a numerical solution, which is mathematically difficult. An escape from this problem might be the acquisition of a commercial simulator. However, it may be expensive for research or more straightforward usage purposes. In addition, the simulation might takes a long time depending on the complexity of the reservoir, the period of time we want to simulate. It implies that applying optimization methods that require many simulations, such as ensemble-based methods, which often need hundreds of simulations to present a proper solution, might be unfeasible. [8].

Aiming to present an alternative to reservoir simulation using

mathematical models or commercial simulators, this work investigates the use of machine learning, particularly the Random Forest model for regression, as an option to conventional simulation processes. To test the proposed methodology, we use a multi-region reservoir model. The pressure response solution to this kind of reservoir with a previously defined liquid rate has been presented by Neto et al. (2021). Therefore, to construct the machine-learning model training dataset, we simulate a set of pressure solutions using a uniform distribution for the reservoir permeability. As a result, the machine learning model can compute the reservoir pressure response given the permeability input for each region. It is important to note that the liquid rate is the same for all simulations. Therefore, if this parameter is also a desired input for the model response, it should be added to the training dataset. However, in this dissertation, we only use permeability as input and pressure response as output, i.e., the liquid rate is considered known and constant for all simulations.

We also aim to demonstrate the potential of machine learning in enhancing the accuracy and efficiency of oil and gas multiregion reservoir simulation, and provide a roadmap for its integration into industry practice. Through extensive experimentation and analysis, this dissertation aims to advance our understanding of the use of machine learning in the simulation of oil and gas multiregion reservoirs. This approach intends to provide a cost-effective alternative to traditional reservoir characterization simulator using ensemble Machine learning techniques.

2
Analytical solutions in multiregion reservoirs

2.1
The Multiregion Reservoir Model

Zhang and Guo (2010)[1] presented a new well-testing model for a two-region linear composite reservoir with varied thicknesses. The model is based on the assumption that the two regions have different thickness along the horizontal direction regions, each one with different homogeneous permeability. The model is derived using the principle of conservation of mass and the Darcy's law, and is applicable to both single and two-phase flow conditions. The validity of the model is verified by comparing the outputs with the results obtained from numerical simulations. The proposed solution is expected to be useful in the analysis of well-testing data and reservoir characterization.

The model and the corresponding type curves are so general that they can be used to predict production performance or to analyze the production data from reservoir systems.

The main idea behind solving a multiregion problem is to reformulate it as a series of single-layer problems with time-dependent equations for each region. The region coupling conditions are then used to determine a new system. Finally, using a numerical approximation of the inverse Laplace transform, these equations are solved and brought to the real field by Marianito and Worthy (2016) [2]. In the Laplace space, the solution for homogeneous multiregion reservoirs was obtained by Lefkovits, et al. [3].

2.2
The one-layer with two regions

The model used in this work assumes that a well fully penetrates a cylindrical reservoir composed of one-layer system with two or three regions. The following hypotheses are considered for the mathematical formulation:

1. Single-phase flow;
2. Reservoir composed of one, two or three regions;
3. Insignificant gravitational forces and pressure gradients;

4. Constant initial pressure (p_i);
5. Constant flow rate throughout the production process;
6. Thickness that remains constant (h);
7. The radius of the last region will always be infinite $r_\infty \to \infty$.

The Figures 2.1 and 2.2 depicts the described reservoir model with two regions. Each region has a unique and homogeneous permeability, but the complete reservoir has a value for the thickness. It is essential to mention that q_1 refers to the reservoir flow rate inner layer, and q_t refers to the total flow rate in the interface, measured on the wellhead. Also, r_1 and r_2 refer to the radius of each reservoir region.

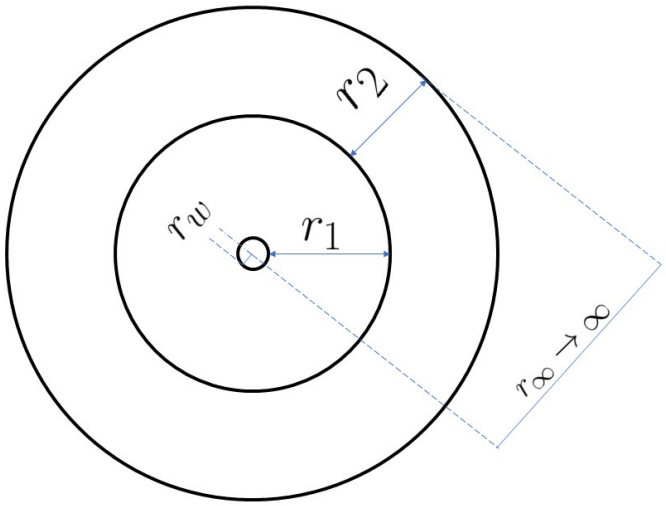

Figure 2.1: Top view of One−Layer two−region reservoir

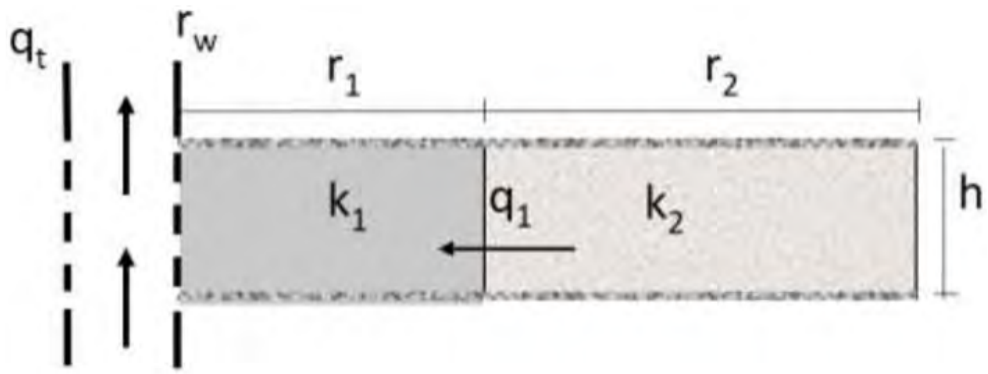

Figure 2.2: One−Layer two−region reservoir model

The following equations are derived for each region:

1. Equations for first region, the partial differential equation, PDE, and the initial boundary conditions, IBC are:

$$\textbf{PDE:}\quad \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p_1}{\partial r}\right) = \frac{1}{\eta_1}\frac{\partial p_1}{\partial t}(r,t), \text{where } r_w < r < r_1 \text{ and } t > 0 \quad (2\text{-}1)$$

$$\textbf{IBC:}\quad p_1(r, t=0) = p_i$$

$$q_{sf} = 2\pi\frac{k_1 h}{\mu}\left(r\frac{\partial p_1}{\partial r}\right)\Bigg|_{r=r_w}$$

2. Equations for second region, the PDE, Initial condition, IC, and the exact boundary condition, EBC, are:

$$\textbf{PDE:}\quad \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p_2}{\partial r}\right) = \frac{1}{\eta_2}\frac{\partial p_2}{\partial t}(r,t), \text{where } r_1 < r < r_\infty \text{ and } t > 0 \quad (2\text{-}2)$$

$$\textbf{IC:}\quad p_2(r, t=0) = p_i$$

$$\textbf{EBC:}\quad \lim_{r\to\infty} p_2(r=\infty, t) = 0$$

The pressure and flow-rate equalities at the interface between regions 1 and 2 require coupling. As a result, the coupling conditions between the regions (CCR) are defined as follows:

$$\text{CCR}: \begin{cases} p_1(r_1,t) = p_2(r_1,t) \\ q_1(r_1,t) = q_2(r_1,t) \end{cases} \tag{2-3}$$

Using Darcy's Law in each region, rewrite the flow rate equality and conclude that:

$$2\pi\frac{k_1 h}{\mu}\left(r\frac{\partial p_1}{\partial r}\right)\bigg|_{r=r_1} = 2\pi\frac{k_2 h}{\mu}\left(r\frac{\partial p_2}{\partial r}\right)\bigg|_{r=r_1} \Longrightarrow \left(r\frac{\partial p_1}{\partial r}\right)_{r=r_1} = \frac{k_2}{k_1}\left(r\frac{\partial p_2}{\partial r}\right)\bigg|_{r=r_1}$$

Using the Laplace Transform in region 1 equations and taking the derivative on the left side:

$$\textbf{PDE:}\quad \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p_1}{\partial r}\right) = \frac{1}{\eta_1}\Big(u - \bar{p_1}(r,u) - p_1(r,t=0)\Big)$$

Using IBC condition, $p_1(r,t=0) = p_i$. We have,

$$\frac{\partial^2}{\partial r^2}\left(\frac{\partial \bar{p_1}}{\partial r}\right) + \frac{1}{r}\frac{\partial \bar{p_1}}{\partial r} - \frac{u}{\eta_1}\bar{p_1} = -p_i,$$

Using the transformation $r \to r\sqrt{\frac{u}{\eta_1}}$:

$$\frac{\partial^2 \bar{p_1}}{\partial\left(r\sqrt{\frac{u}{\eta_1}}\right)^2} + \frac{1}{r\sqrt{\frac{u}{\eta_1}}}\frac{\partial \bar{p_1}}{\partial\left(r\sqrt{\frac{u}{\eta_1}}\right)} - \bar{p_1} = -p_i\,\frac{\eta_1}{u}$$

$$\text{IBC:}\quad \left(\frac{1}{r}\frac{\partial \bar{p_1}}{\partial r}\right)\bigg|_{r=r_w} = \frac{q_{sf}\mu}{2\pi k_1 h}\frac{1}{u}$$

Using the same idea for equations of region 2, we obatin the following:

$$\frac{\partial^2 \bar{p_2}}{\partial\left(r\sqrt{\frac{u}{\eta_2}}\right)^2} + \frac{1}{r\sqrt{\frac{u}{\eta_2}}}\frac{\partial \bar{p_1}}{\partial\left(r\sqrt{\frac{u}{\eta_1}}\right)} - \bar{p_2} = -p_i\,\frac{\eta_2}{u}$$

$$\text{EBC:}\quad \lim_{r\to\infty} p_2(r,u) = \frac{p_i}{u}$$

Define

$$\Delta\bar{p}(r,t) \coloneqq p_i - p(r,t). \tag{2-4}$$

Applying it in all equations for both regions, the following equations are obtained:

1. For Region 1:

$$\frac{\partial^2 \Delta\bar{p_1}}{\partial\left(r\sqrt{\frac{u}{\eta_1}}\right)^2} + \frac{1}{r\sqrt{\frac{u}{\eta_1}}}\frac{\partial \Delta\bar{p_1}}{\partial\left(r\sqrt{\frac{u}{\eta_1}}\right)} - \Delta\bar{p_1} = 0 \tag{2-5}$$

$$\text{IBC:} \quad \left(\frac{1}{r}\frac{\partial \Delta\bar{p_1}}{\partial r}\right)\bigg|_{r=mw} = \frac{q_{sf}\mu}{2\pi k_1 h}\frac{1}{u} \tag{2-6}$$

2. For Region 2:

$$\frac{\partial^2 \Delta\bar{p_2}}{\partial\left(r\sqrt{\frac{u}{\eta_2}}\right)^2} + \frac{1}{r\sqrt{\frac{u}{\eta_2}}}\frac{\partial \Delta\bar{p_2}}{\partial\left(r\sqrt{\frac{u}{\eta_2}}\right)} - \Delta\bar{p_2} = 0 \tag{2-7}$$

$$\text{EBC:} \quad \lim_{r\to\infty} \Delta p_2(r,u) = 0 \tag{2-8}$$

$$\text{CCR}: \begin{cases} \Delta p_1(r_1,t) = \Delta p_2(r_1,t) \\ \left(r\frac{\partial \Delta p_1}{\partial r}\right)_{r=r_1} = \frac{k_2}{k_1}\left(r\frac{\partial \Delta p_2}{\partial r}\right)\Big|_{r=r_1} \end{cases} \tag{2-9}$$

Solutions

There're two Ordinary Differential Equations with boundary, contour and initial conditions. This system is a well-known Bessel equations which analytical solutions are linear combination of Bessel functions.

1. Region 1:

$$\Delta\bar{p_1}(r,u) = A_1 K_0\left(r\sqrt{\frac{u}{\eta_1}}\right) + B_1 I_0\left(r\sqrt{\frac{u}{\eta_1}}\right) \tag{2-10}$$

2. Region 2:

$$\Delta\bar{p_1}(r,u) = A_2 K_0\left(r\sqrt{\frac{u}{\eta_1}}\right) + B_1 I_0\left(r\sqrt{\frac{u}{\eta_1}}\right) \tag{2-11}$$

Using IBC, EBC, and Bessel functions properties, we found:

$$A := \begin{bmatrix} K_1\left(r_w\sqrt{\frac{u}{\eta_1}}\right) & -I_1\left(r_w\sqrt{\frac{u}{\eta_1}}\right) & 0 \\ K_0\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & I_0\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & -K_0\left(r_1\sqrt{\frac{u}{\eta_2}}\right) \\ K_1\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & -I_1\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & -\frac{k_2}{k_1}\sqrt{\frac{\eta_1}{\eta_2}}K_1\left(r_1\sqrt{\frac{u}{\eta_2}}\right) \end{bmatrix} \tag{2-12}$$

$$\begin{bmatrix} A_1 \\ B_1 \\ A_2 \end{bmatrix} = A^{-1}\begin{bmatrix} M\frac{1}{u} \\ 0 \\ 0 \end{bmatrix} \tag{2-13}$$

where

$$M = \frac{1}{r_w\sqrt{\frac{u}{\eta_1}}}\frac{q_{sf}\mu}{2\pi k_1 h} \tag{2-14}$$

Equation (2-13) shows the solution for the system (2-12) considering the equation (2-1) and (2-2). We can see that the solution behaves, as seen in Figure (2.3) and (2.4). The Figure (2.3) displays the solution for a reservoir with two regions, where the first region has a permeability of $k1 = 500\ mD$ and the second region $k2 = 5000\ mD$ and the figure (2.4) and $k2 = 5000\ mD$ and $k2 = 500\ mD$, respectively:

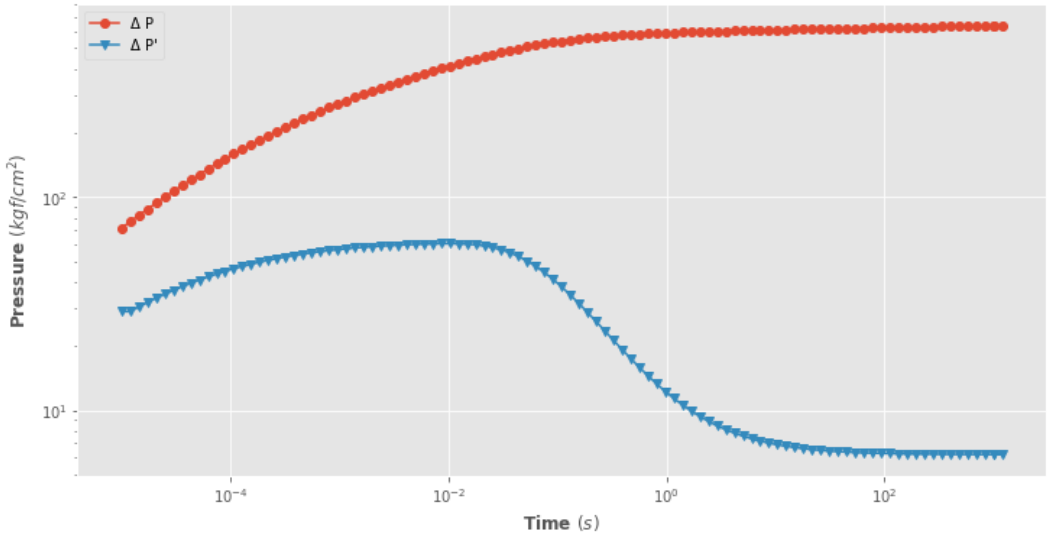

Figure 2.3: The analytical solution of the single phase one-layer with two regions with $k_1 = 500\ mD$ and $k_2 = 5000\ mD$.

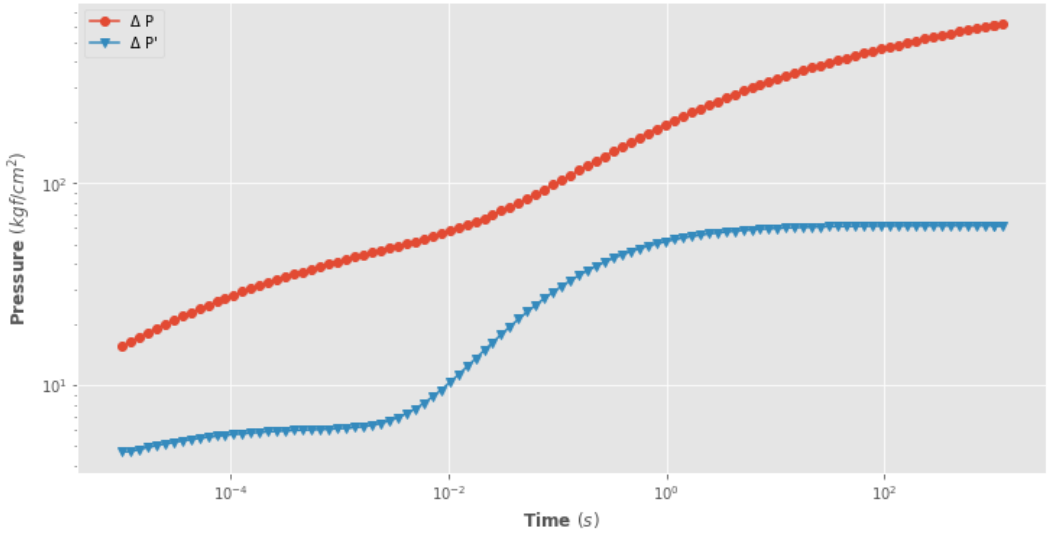

Figure 2.4: Analytical solution of the single phase one-layer with two regions with $k_1 = 5000\ mD$ and $k_2 = 500\ mD$.

Pressure derivative curve indicates when a transition between two regions with different permeability occurs. The graph's decline is produced by the progressive rise in permeability in each layer's following sections. Because the model assumes a constant flow rate, the pressure derivative is similarly likely to be constant.

2.3 The Single-Phase One-Layer with Three Regions

The considered model can be seen above for a three-region single-layer reservoir. The Figure 2.3 depicts reservoir model for the single phase one-layer with three regions problem can be described as follows:

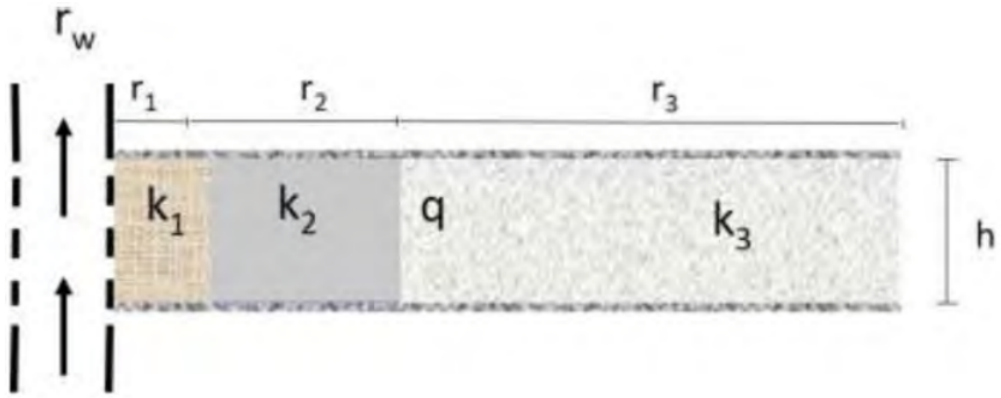

Figure 2.5: One-Layer, three regions reservoir model

The development of the wellbore pressure solution is analogous to that one presented for the two-region reservoir case. The following matrix obtained in this case:

$$\mathrm{A} = \begin{bmatrix} K_1\left(r_w\sqrt{\frac{u}{\eta_1}}\right) & -I_1\left(r_w\sqrt{\frac{u}{\eta_1}}\right) & 0 & 0 & 0 \\ K_0\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & I_0\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & -K_0\left(r_1\sqrt{\frac{u}{\eta_2}}\right) & -I_0\left(r_1\sqrt{\frac{u}{\eta_2}}\right) & 0 \\ K_1\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & -I_1\left(r_1\sqrt{\frac{u}{\eta_1}}\right) & -\frac{k_2}{k_1}\sqrt{\frac{\eta_1}{\eta_2}}K_1\left(r_1\sqrt{\frac{u}{\eta_2}}\right) & \frac{k_2}{k_1}\sqrt{\frac{\eta_1}{\eta_2}}I_1\left(r_1\sqrt{\frac{u}{\eta_2}}\right) & 0 \\ 0 & 0 & K_0\left(r_2\sqrt{\frac{u}{\eta_2}}\right) & I_0\left(r_2\sqrt{\frac{u}{\eta_2}}\right) & -K_0\left(r_2\sqrt{\frac{u}{\eta_3}}\right) \\ 0 & 0 & K_1\left(r_2\sqrt{\frac{u}{\eta_2}}\right) & -I_1\left(r_2\sqrt{\frac{u}{\eta_2}}\right) & -\frac{k_3}{k_2}\sqrt{\frac{\eta_2}{\eta_3}}K_1\left(r_2\sqrt{\frac{u}{\eta_3}}\right) \end{bmatrix} \tag{2-15}$$

Whereas matrix A should meet the following criteria:

$$\begin{bmatrix} A_1 \\ B_1 \\ A_2 \\ B_2 \\ A_3 \end{bmatrix} = A^{-1} \begin{bmatrix} M\frac{1}{u} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2-16}$$

and

$$M = \frac{1}{r_w\sqrt{\frac{u}{\eta_1}}}\frac{q_{sf}\mu}{2\pi k_1 h}$$

The figure 2.6 show the analytical solution of the single phase one-layer with three regions with permeabilities $k_1 = 500\ mD$, $k_2 = 5000\ mD$ and $k_3 = 500\ mD$. The 2.7 show the analytical solution of the single phase one-layer

with three regions with permeabilites $k_1 = 5000\ mD$, $k_2 = 500\ mD$ and $k_3 = 6000\ mD$:

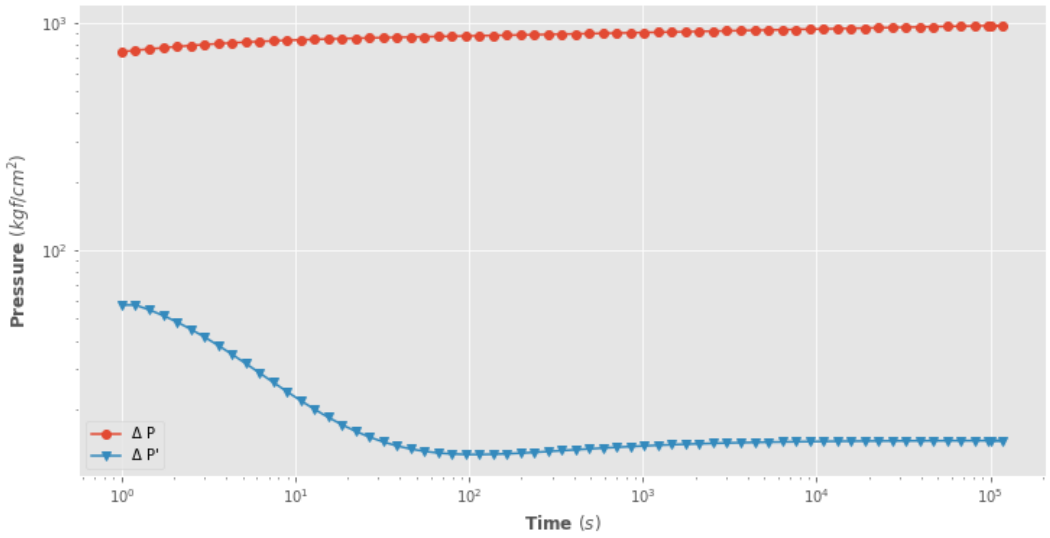

Figure 2.6: The analytical solution of the single phase one-layer with three regions with $k_1 = 500\ mD$, $k_2 = 5000\ mD$ and $k_3 = 500\ mD$.

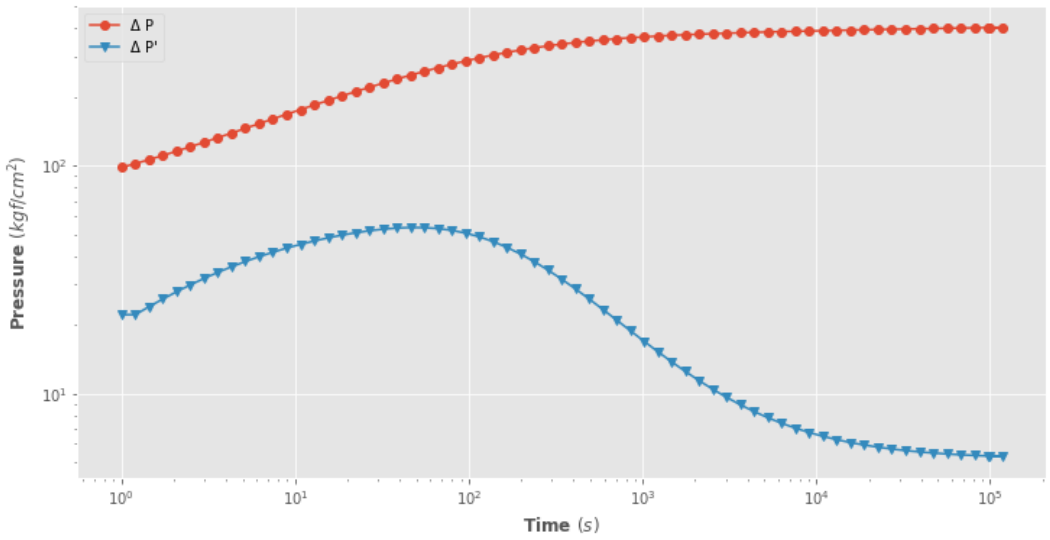

Figure 2.7: The analytical solution of the single phase one-layer with three regions with $k_1 = 5000\ mD$, $k_2 = 500\ mD$ and $k_2 = 6000\ mD$.

3
The Random Forest Algorithm

In our problem, we identify both the pressure and its derivative as a function of time. Therefore, we can sort permeability values from a specific probability distribution. Bearing this in mind, we can use such data to train a statistical learning model to then simulate the distribution of line source solutions based on permeability distribution.

We believe it is important to introduce a chapter and expose some important concepts to understand the reason we will use the Random Forest regressor for reproducing reservoir simulations.

3.1
Machine Learning and Learning paradigms

Arthur Samuel, an IBM employee and pioneer in the fields of artificial intelligence and computer gaming, coined the term "machine learning" in 1959. Machine learning is a subfield of artificial intelligence that uses concepts of mathematics, statistics and computation and involves training computer systems to make predictions or decisions based on patterns in data, rather than explicit instructions.

Machine learning approaches are traditionally divided into three broad categories, which correspond to learning paradigms, depending on the nature of the "signal" or "feedback" available to the learning system: supervised learning, unsupervised learning and reinforcement learning:

Unsupervised Learning

In this work we did not adopt unsupervised methodologies. Therefore, we will give an overview of the topic. Unsupervised learning is a sort of algorithm that uses unlabel data to discover patterns. The idea is for the computer to be driven to develop a compact picture of its surroundings and then generate innovative material through imitation, which is an essential form of learning in humans.

Unsupervised approaches display self-organization that captures patterns as probability densities or a mixture of neural feature preferences stored in

the machine's weights and activations, as opposed to supervised learning where data is tagged by an expert, such as "ball" or "fish." Further degrees of supervision include reinforcement learning, in which the machine is just provided a numerical performance score as guidance, and semi-supervised learning, in which only a small fraction of the data is labeled.

Supervised Learning

Our entire methodology is based on a supervised algorithm, specifically a regression via a Random Forest regression algorithm. Supervised learning algorithms construct a mathematical model of a set of data that includes both the inputs and the desired outputs. Supervised learning is a machine learning paradigm for circumstances in which the available data consists of labeled instances, i.e., each data point comprises characteristics and an associated label. Russell et al.(2010) [12] said that the goal of supervised learning algorithms is to create a function that transforms feature vectors (inputs) to labels, also known as outputs, based on example input-output pairs. It derives a function from labeled training data, which consists of a set of training examples. According to Mahri et al. (2012)[13], each example in supervised learning is a pair consisting of an input item (typically a vector) and a desired output value (commonly called supervisory signal).

Classification, where the aim is to predict the class label of a new input data point, and regression, where the goal is to predict a continuous output value, are two examples of supervised learning. Natural language processing, computer vision, and speech recognition are all applications of supervised learning.

Consider the training examples of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$ such that x_i is the feature vector of the i-th example and y_i is its label (i.e., class or value), a learning algorithm searches for a function $g : X \to Y$, where X is the input space and Y is the output space. The function g is an element of some space of possible functions $G = G(X, Y)$, usually called the hypothesis space. It is sometimes convenient to represent g using a scoring function $f : X \times Y \to \mathbb{R}$ such that g is defined as returning the y value that gives the highest score: $g(x) = \arg\max_y f(x, y)$.

Let $F = F(X, Y)$ denote the space of scoring functions. Although G and F can be any space of functions, many learning algorithms are probabilistic models where g takes the form of a conditional probability model $g(x) = P(y|x)$, or f takes the form of a joint probability model $f(x, y) = P(x, y)$. For example, Naive Bayes classifier and Naive Bayes linear discriminant analysis

are joint probability models, but logistic regression is a conditional probability model.

There are two techniques to deciding whether to use f or g: empirical risk minimization and structural risk minimizing. The function that best matches the training data is sought after via empirical risk reduction. A penalty function manages the bias/variance tradeoff in structural risk minimization.

In both cases, it is assumed that the training set consists of a sample of independent and identically-distributed random variables, independent and identically distributed pairs, $(x_i,\ y_i)$. In order to measure how well a function fits the training data, a loss function $L : Y \times Y \to \mathbb{R}^{\geq 0}$ is defined. For training example $(x_i,\ y_i)$, the loss of predicting the value $\hat{y}$ is $L(y_i, \hat{y})$.

The risk $R(g)$ of function g is defined as the expected loss of g. This can be estimated from the training data as

$$R_{emp}(g) = \frac{1}{N} \sum_i L(y_i, g(x_i))$$

Regression Algorithms

Regression algorithms are a type of machine learning algorithm used to predict continuous numerical values based on input data. The purpose of regression analysis is to construct a mathematical model that can predict the output variable accuratly based on one or more input factors.

There are several types of regression algorithms and the choice of which algorithm to use depends on the nature of the problem and the characteristics of the dataset.

1. Linear regression: Linear regression use a linear equation to represent the connection between the input variables and the output variables.

2. Polynomial regression: Polynomial regression is an extension of linear regression that models the relationship between the input variables and the output variable using a polynomial equation.

3. Ridge regression: it is a regularized version of linear regression that introduces a penalty term to prevent overfitting.

4. Lasso regression: it is another regularized version of linear regression that introduces a penalty term and can be used for feature selection.

5. Support Vector Regression (SVR): it is a regression algorithm that utilizes support vector machines to establish a connection between

input and output variables. Its objective is to discover a function that approximates the relationship between a set of input variables and a continuous target variable. Unlike conventional regression methods that minimize prediction errors, SVR concentrates on identifying a hyperplane that maximizes the margin within which the majority of training data points reside.

6. Decision tree regression: a non-parametric algorithm that models the relationship between the input variables and the output variable using a decision tree.

7. Random forest regression: Random forest regression is an ensemble algorithm that combines multiple decision trees to improve the accuracy of the predictions.

Ensemble Learning

Ensemble learning is the process of averaging the results of multiple models that have been trained on the same data to find a more powerful predictive regression/classification result. Our hope, and requirement, for ensemble learning is that the errors of each model (in Random Forest case, decision tree) are independent and distinct from tree to tree, according to Cutler (2012) et. al. [5] and Breiman (1996) [10].

Ensemble machine learning regressors are used to solve non-linear problems by combining multiple models. In this approach, a set of weak learners are combined to create a strong learner. The weak learners can be decision trees, neural networks, support vector machines or other models. The idea is to train each weak learner on a subset of the training data or with different feature sets to create diversity in the model.

Once the weak learners are trained, the ensemble model is created by combining the output of all the weak learners. This combination can be done in various ways such as taking the mean, median, or mode of the outputs. Alternatively, a weighted combination of the weak learners can be used, where the weights are determined by the performance of each weak learner on the validation set.

By combining multiple models, the ensemble machine learning regressor can overcome the limitations of a single model and can capture complex non-linear relationships between the input variables and the target variable. This approach can improve the accuracy and generalization of the model, making it more robust to noise and outliers in the data.

We can cite three major groups: Bagging, Boosting and Stacking:

1. Bagging, which stands for Bootstrap Aggregation, is a type of ensemble method in machine learning that involves training multiple instances of the same model using different subsets of the training data. In bagging, each instance of the model is trained on a randomly selected subset of the training data with replacement. This means that some examples may be repeated in different subsets and others may be left out. The output of the ensemble model is obtained by taking the average (in regression problems) or the majority vote (in classification problems) of the predictions made by each model instance. Bagging is particularly useful for reducing overfitting in complex models such as decision trees or neural networks.

2. Boosting is an ensemble method in machine learning where multiple weak models are combined to create a strong model. It works by iteratively training a weak model on the data, and then giving more weight to the misclassified data points for the next iteration. This process is repeated multiple times until the weak models are combined to form a strong model that performs better than any individual weak model. Some popular boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost. Boosting is particularly effective in improving the performance of decision trees, and has been successfully applied to a variety of machine learning tasks such as classification, regression, and ranking.

3. Stacking is an ensemble learning method that combines multiple models to improve the performance of a prediction task. It is a meta-algorithm that uses the predictions of base models to train a meta-model. The base models are trained on the same data and their predictions are used as input features for the meta-model. Stacking can handle complex non-linear relationships and can improve the accuracy over individual models or simpler ensemble methods. However, stacking can also lead to overfitting and is computationally expensive.

Bootstrapping

Bootstrapping is the process of randomly sampling subsets of a dataset over a specified number of iterations and variables. These results are then averaged to produce a more powerful result. Bootstrapping is particularly useful for ensemble learning, where the objective is to construct a combination of weak learners dealing with data variability.

3.2 The Random Forest Algorithm

The random forest algorithm applies the general technique of bootstrap aggregation, or bagging, to tree learners. As a result, the bootstrapping Random Forest algorithm combines ensemble learning methods with the decision tree framework to generate multiple randomly drawn decision trees from the data, then averaging the results to produce a new result that frequently leads to strong predictions as regressions/classifications.

The Random Forest algorithm is an ensemble learning method for classification, regression and other tasks. It works by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees [5], [10]. Random Forest algorithm can be used in predicting production in an oil field through the Random Regression algorithm which takes into account various factors affecting production such as well parameters, geological conditions, and production history to create a predictive model for future production. This approach has been shown to be effective in predicting oil and gas production in real-world applications.

The random forest training algorithm applies the general technique of bootstrap aggregation, or bagging, to tree learners. As a result, the bootstrapping Random Forest algorithm combines ensemble learning methods with the decision tree framework to generate multiple randomly drawn decision trees from the data, then averaging the results to produce a new result that frequently leads to strong predictions as regressions/classifications. The figure (3.1) illustrates a brief description of how the random forest regression algorithm works:

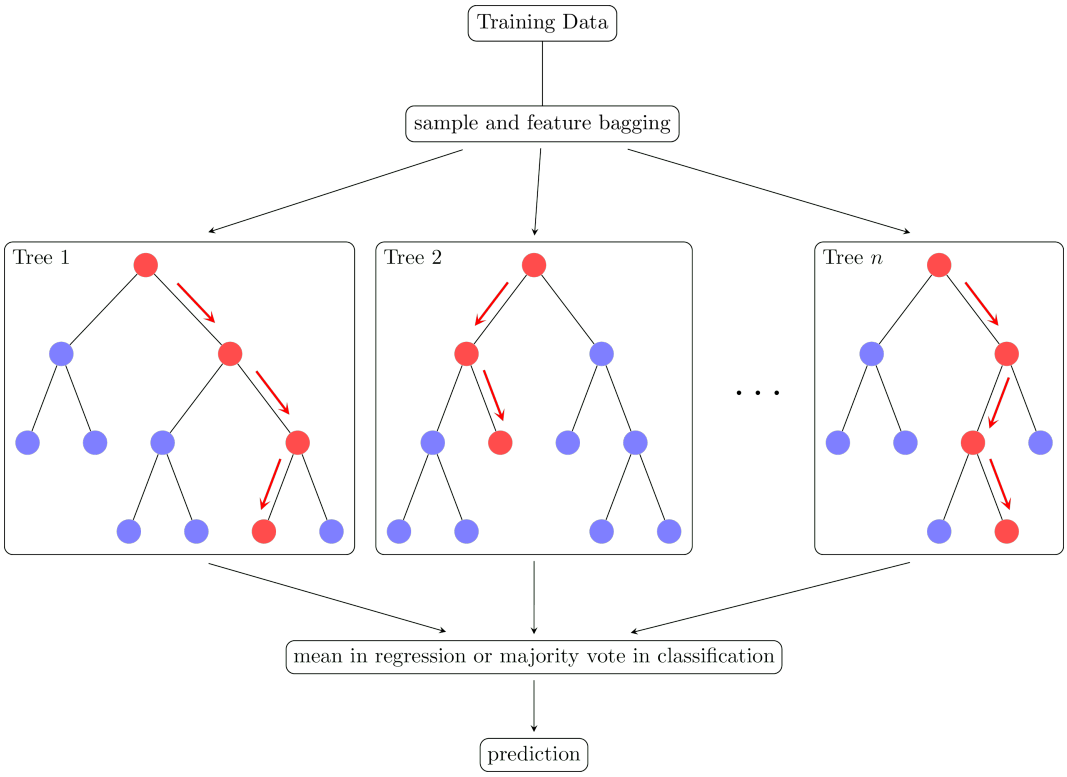

Figure 3.1: A brief description of how the random forest algorithm works

Given a training set $X = \{x_1, \ldots, x_n\}$, with responses $Y = \{y_1, \ldots, y_n\}$ bagging B repeatedly times and selects a random sample with replacement of the training set and fits trees to these samples. So,

For $b = 1, \ldots, B$:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.

2. Train a regressor tree f_b over X_b, Y_b.

3. Predictions for unseen samples x_{val} can be made by averaging the predictions from all the individual regression trees on x_{val}:

$$\hat{f}(x_{val}) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_{val})$$

The Random Forest Regression algorithm is less prone to over-fitting and produces predictions and delivers more reliable predictions than a single decision trees, according to Breiman (2001) [10]. Furthermore, they are relatively simple to apply, considering known frameworks such as scikit-learn for python, do not require data transformation, are less susceptible to irrelevant

or highly correlated input features, and easy to evaluate the factors that control the prediction, the decision structure and the relative importance of each input variable.

In addition, given that the problem in this work involves simulate solutions of partial differential equations perturbing the permeability parameter in each region, the use of Random Forest Regressor could offers some advantages and motives, including:

1. Nonlinearity and Complex Relationships: PDE simulations often involve complex and nonlinear relationships between input variables and the desired output. Random Forest Regressor excels in capturing these intricate relationships, as it can handle nonlinearities and interactions between variables effectively.

2. Robustness to Outliers: PDE simulations may encounter outliers or noisy data points, which can adversely affect the accuracy of the simulation. Random Forest Regressor is robust to outliers, as it constructs an ensemble of decision trees and makes predictions based on the collective knowledge of multiple trees, reducing the impact of individual outliers.

3. Feature Importance Assessment: Random Forest Regressor provides a measure of feature importance, indicating which input variables contribute the most to the predicted output. This information can help in understanding the underlying physics or dynamics of the PDE system and guide feature selection or refinement of the simulation model.

4. Handling High-Dimensional Data: PDE simulations often involve a large number of input variables, resulting in high-dimensional data. Random Forest Regressor can effectively handle high-dimensional data without the need for dimensionality reduction techniques, such as feature selection or dimensionality reduction algorithms.

5. Model Flexibility and Robustness: Random Forest Regressor is known for its flexibility and robustness, making it suitable for various PDE simulation scenarios. It can handle different types of input variables (continuous, categorical) and accommodate missing data or imbalanced datasets.

6. Scalability: Random Forest Regressor can be efficiently parallelized and distributed, allowing it to handle large datasets and perform computations in a scalable manner. This scalability is advantageous when

dealing with PDE simulations that involve a significant amount of data or complex systems.

7. Model Interpretability: While Random Forest Regressor is an ensemble method composed of multiple decision trees, it still provides a level of interpretability. The feature importance measures and decision paths of individual trees can offer insights into the relationships between input variables and the PDE system.

3.3 Loss Function

A loss function in machine learning is a function that converts the difference between the anticipated and actual values of the target variable to a real number. The loss function measures how well the model performs on training data. The objective of machine learning is to minimize the loss function by determining the best set of model parameters to reduce the difference between predicted and actual values. Depending on the goal, multiple types of loss functions are utilized, such as regression, classification, and clustering. The loss function to be utilized is determined by the task at hand and the type of model being employed.

The loss function has a substantial influence on the model's performance and parameter optimization and some loss functions are more sensitive to changes in input data than others, according to Hastie et. al (2017) [18]. In general, there are two types of loss functions: regression and classification. The discrepancy between the predicted and actual values of a continuous variable is measured by regression loss functions, whereas the difference between the predicted and real class labels of a categorical variable is measured by classification loss functions.

One commonly used regression loss function is the mean squared error (MSE), which penalizes large errors than small errors. However, the MSE is highly sensitive to outliers, which can significantly affect the optimization of the model parameters.

1. The Mean Absolute Error (MAE) is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3-1}$$

where y_i is the actual value of the i-th observation and $\hat{y}_i$ is the predicted value [18].

The equation presented here illustrates the calculation of the area between the observed values' curve (y_i) and the predicted values' curve

($\hat{y}_i$). The modulus in this equation plays a crucial role in eliminating the negative contributions to the area. Consequently, as the value of this area decreases, the overall proximity between the curves improves.

2. The mean squared error (MSE) and can be expressed as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3-2}$$

where y_i is the actual value of the i-th observation and $\hat{y}_i$ is the predicted value [18].

This measure has been designed in a manner that assigns greater weight to observed data (y_i) that deviate more significantly from the predicted data ($\hat{y}_i$) in the error measure. This weighting effect is achieved through the use of a power function.

3. Finally, specially on forecast analysis, according to [18], the root mean squared error, $RMSE$, is another Loss Function defined as:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3-3}$$

In general, RMSE is preferred over MAE and MSE as it provides the same units as the predicted variable. However, the choice of loss function depends on the problem at hand and the goals of the model [18]. For example, if minimizing the absolute difference between predicted and actual values is more important than minimizing the squared differences, then MAE may be a more appropriate loss function. A brief explanation of the differences between them:

1. MAE measures the average magnitude of the errors in a set of predictions. It is less sensitive to outliers compared to MSE and RMSE because it takes the absolute value of the differences between the predicted and actual values. This makes it a good choice when outliers are present in the data or when we want to minimize the impact of large errors.

2. MSE measures the average of the squared differences between the predicted and actual values. Because it takes the squared value, it is more sensitive to outliers compared to MAE and RMSE. MSE is useful when we want to penalize large errors more heavily than small errors.

3. RMSE is the square root of the average of the squared differences between the predicted and actual values. It has the same units as the predicted variable, making it easier to interpret than MSE. Like MSE, RMSE is sensitive to outliers. However, it is less sensitive than MSE because of

the square root operation, which helps to reduce the influence of large errors.

Machine learning has become an integral part of many industries and fields, from finance and healthcare to manufacturing and entertainment. At the core of many machine learning models is the ability to make predictions based on patterns in data. These patterns can range from simple relationships between variables to complex interactions between multiple factors.

To make accurate predictions, machine learning models need to be trained using a loss function that measures the difference between predicted and actual outcomes. This loss function helps to guide the model towards making more accurate predictions by adjusting its parameters based on the amount of error between predicted and actual outcomes.

The choice of loss function is a crucial aspect of developing a successful machine learning model. It depends on the specific problem being addressed and the goals of the model. For example, if we are building a model to predict stock prices, we may want to minimize the average magnitude of errors. In this case, MAE would be a good choice of loss function. MAE takes the absolute value of the differences between predicted and actual outcomes, which means it is less sensitive to outliers compared to other loss functions. This is useful when we want to minimize the impact of large errors in our predictions, which can be caused by outliers or noise in the data.

Another important consideration when choosing a loss function is the type of data being used. For example, if we are working with continuous data, such as temperature readings or financial data, MSE (Mean Squared Error) or RMSE (Root Mean Squared Error) may be a better choice. These loss functions are sensitive to differences in magnitude between predicted and actual outcomes, which can be important when working with continuous data.

Overall, the choice of loss function is a critical step in developing a successful machine learning model. It requires careful consideration of the specific problem being addressed, the goals of the model, and the type of data being used. By choosing an appropriate loss function, we can help to ensure that our machine learning model is accurate, reliable, and useful in real-world applications.

Outlier and Robustness of Loss function

An outlier is described as an uncommon action, incursion, or suspicious data point that is located at an irregular distance from a community. Nevertheless, the definition of an outlier event is subjective and relies on the

application and domain (energy, health, wireless network, etc.). To eliminate potential mistakes or biases in data analysis and modeling, it is critical to recognize outlier occurrences as precisely as possible. Several techniques, including visual inspection, z-score analysis, clustering, and density-based algorithms, can be used to find outliers. Outliers are handled differently depending on the context and the objective of the study, and they can be removed, transformed, or treated as a separate class [11].

Robustness is described as a system's or a model's ability to remain stable and exhibit only minor changes (or none at all) when subjected to noise or exaggerated inputs. Outliers must have less of an impact on a robust system or metric. In this case, it is straightforward to conclude that MSE is less resilient than MAE since the squaring of the errors places a greater emphasis on outliers.

3.4 Hyperparameter optimization

Hyperparameter optimization refers to the process of finding the best combination of hyperparameters for a machine learning algorithm to achieve the highest performance on a given task. Hyperparameters are configuration settings that are set prior to training a machine learning model. As stated in by Hutter et. al [16], "the goal of hyperparameter optimization is to select the values of the hyperparameters that lead to the best performance on a given task, as measured by a chosen evaluation criterion." This process is often performed using search algorithms such as grid search, random search, or Bayesian optimization.

Grid search involves establishing a grid of hyperparameter values to explore, then training and assessing the model for each grid combination of hyperparameters. For each hyperparameter combination, the evaluation measure used to assess the model's performance (such as accuracy or mean squared error) is recorded. The optimal set of hyperparameters for the model is then chosen as the combination of hyperparameters that results in the best performance. With a random forest regressor, for example, the number of trees in the forest, the maximum depth of the trees, and the minimum number of samples necessary to split a node are all hyperparameters that may be tweaked.

While grid search is a straightforward and popular approach for adjusting hyperparameters, it can be computationally costly if the hyperparameter space is huge. Alternative approaches, such as random search and Bayesian optimization, can be employed to circumvent this constraint and more effectively locate ideal hyperparameters.

Grid search then iteratively searches through all possible combinations of the provided hyperparameters to find the optimum combination that results in the greatest model performance as assessed by a chosen metric such as mean squared error or R-squared.

4
Simulating single-layered multiregion reservoir

4.1
Proposed methodology

The complexity of reservoir models and simulations of well-testing and their analysis of observations brought us costly market solutions, which is why machine learning might be an alternative. The goal of this research is to show how machine learning may improve the time consumed with acceptable precision and effectiveness of oil and gas reservoir simulations and to lay out a plan for its adoption in commercial practice. This dissertation seeks to expand our knowledge of the application of machine learning in the simulation of oil and gas reservoirs through comprehensive experimentation and analysis.

The input data is generated only considering permeability perturbations. Although, the equations in chapter 2 use some other important features like porosity and permeability. Permeability indicates the ease of hydrocarbons flow and production from a reservoir. The porosity and permeability of reservoir formation is important in determining the reservoir exploration project.

A reservoir formation that is both porous and permeable might need less complexity in the production management as it allows oil and gas to move up through the pores into the wells with primary recovery method, from the reservoir to the surface where it can be extracted.

The proposed solution aims to reduce the complexity of using mathematical or commercial simulators, but it is necessary to build a training dataset using one of these simulators. However, when we seek time optimization, simulator reproduction becomes feasible with the use of machine learning. The present work seeks to open the discussion of the application of machine learning for reservoir simulation. Therefore, we will build the training dataset using an analytical model available in the literature. Such a model has already been proven to faithfully reproduce solutions also available by traditional commercial simulators. With this choice, we can focus our work on developing strategies to improve the use of machine learning in reservoir simulation. In this approach, we only use permeability as an input for the model, although it is possible to use many parameter as an input, which leads

to a greater degree of freedom for the simulation. However, it is expected that the solution becomes less accurate as input parameters are added, due to the phenomenon known as Curse of Dimentionality, by Bellman (1957) [14], that discusses the difficulties of interpreting data effectively in high-dimensional spaces. It happens when the number of dimensions, characteristics, or variables in a dataset expands exponentially faster than the amount of data available to fill those spaces.

Due to the complexity and non-linearity of the equation, historically, tree-based algorithms end up being an alternative of interest. Random Forest Regressor is not directly applicable for simulating differential equations. As known, differential equations describe the relationship between a function and its derivatives, and their solution requires numerical methods and mathematical modeling rather than machine learning algorithms.

However, one possible way to use a Random Forest Regressor for simulating differential equations is to treat it as a surrogate model or an emulator of the differential equation solver. In this approach, one would train a Random Forest Regressor on a set of input-output pairs generated by a reference differential equation solver. The input features could represent the initial conditions, parameters, or other relevant variables of the differential equation, while the output targets would correspond to the solution of the differential equation at a given time or a set of times.

Once the Random Forest Regressor is trained, it can be used to predict the solution of the differential equation for new input conditions. This approach can be useful for reducing the computational cost of solving complex differential equations, especially in cases where the differential equation solver is computationally expensive or time-consuming. However, it is important to note that the accuracy of the Random Forest Regressor-based emulator will depend on the quality and representativeness of the training data and the complexity of the differential equation being solved.

Our approach is basically to unite numerical solutions as data and generate new solutions with acceptable errors to be considered a lower cost and higher speed solution. Hereafter we present the methodology applied to construct the machine learning model:

1. Data Preparation: the first step is to prepare the data for training the Random Forest algorithm. This includes obtaining data on the properties of the two or three regions in each region, such as permeability, region radius, thickness, and other relevant properties. In this work, what we use the results from Chapter 2 to generate training targets for the training data from permeability values taken randomly from uniform

distributions. We will consider the most immediate case of reservoir simulation, where permeability is the only perturbation of the system.

2. Training the Machine Learning model: the Random Forest algorithm is trained using the prepared data and features. This involves setting the parameters of the algorithm, such as the number of trees and the maximum depth of each tree.

3. Hyperparameter tuning by grid search: A grid search is performed over a predefined set of hyperparameters to determine the optimal combination of hyperparameters that leads to the best performance of the model. After the experiments, we verified that hyperparameters tuning was only necessary for the case of three regions, since the difference in results for one and two regions was considered small.

4. Prediction: Once the model is trained, it can be used to make predictions on new data. In this case, the model can be used to predict the pressure response in well-testing based on the properties of the two or three regions.

5. Validation: The final step is to validate the model by comparing the predictions made by the Random Forest algorithm to the actual pressure data obtained from well-testing. The accuracy of the model can be assessed using metrics such as mean absolute error, or root mean squared error.

Throughout our work we used the Numpy, Scipy and Scikit-Learn python libraries. The model was implemented by the native framework of scikit-learn, considering its standard configuration for the well-testing simulations for the cases of 1 and 2 regions. For the third case, we verified that grid search returned a configuration of number of estimators equal to 1000 trees in the forest, the maximum depth of the trees equal to 10, the minimum number of samples split equals to 8, minimum number of leaf samples equal to 3, the maximum features equal to 2 and bootstrap parameter equals to 'True', which means that bootstrap samples are used when building trees.

4.2 Simulating reservoir with one region

The analytical solution for pressure response in an well-testing given a random normal distribution of permeabilities can be obtained using the mathematical concept of superposition. The basic idea is to represent the

pressure variation at any point within the reservoir as the sum of the pressure contributions from each individual region with permeability K_i.

The analytical solution based on permeability was introduced in last chapter for reservoir model for single phase one-layer with two and three regions. But, considering the simplest case of this problem: the laplace solution for one layer with only one region.

The process here is use the distribution of analytical solutions of pressure response in an oil well obtained by generating multiple realizations of the permeabilities from the given random normal distribution $K_1, ..., K_n$ and then solving for the pressure response at any point within the reservoir for each realization, store the pressure response for each realization in an array, considering the reservoir's properties constant such as wellbore radius, flow rate, thickness, compressibility, porosity, etc. The problem then is: given a distribution of permeabilities $K = (k_1, ..., k_n)$ where n is a reasonably sufficient value to train the model, through the analytical solution, we generate the Laplace solutions in time $(f_1, ..., f_n)$, considering all above. With this, we will only have the set K as a feature and the generated functions as a target. Considering this as tabular data, we split it into training and test data to train the selected model, the Random Forest. So, our solution occurred as follows: given k, there is a function that maps k to a vector in $\mathbb{R}^m$,

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

$$k \mapsto f(k),$$

where such vector is composed by m points of its analytic solution, spaced apart in time-steps $t_1, ..., t_m$. An important insight is that the goal of the machine learning model is to get as close as possible to such a function f, given K.

We create two and three uniform distributions to simulate each region in the one-layer reservoir model with two and three regions, respectively. However, the procedure is the same; the only difference is that we now have two or three columns, respectively, as a feature, as opposed to just one. Furthermore, in these cases, we have an analogous function, but $k = (k_1, k_2)$, and $n = 2$, and $k = (k_1, k_2, k_3)$ and $n = 3$, respectively. The figures (4.1) and (4.2) describe a bundle for single-phase reservoir model with 2 and 3 regions, respectivaly, generated by uniform distributions.

We decided to choose distributions K_1, K_2, K_3 such that all were generated by a uniform distribution of non-disjoint and disjoint intervals, so that we can simulate cases where the regions have permeabilities with several

different combinations, such as, for example

$$K_1 \cap K_2 \neq \emptyset$$

$$\exists q, w \in \{1, 2, 3\}\ , j \in \{1, \ldots, n\}, K_q(j) \geq K_w(j)$$

It will be shown throughout the tests that, given a solution, regardless of the differences between the permeabilities in each of the regions, the methodology has reasonable results, therefore only a simulation with a reasonable amount of input values for each case - one region, two regions and three regions - would be enough to evaluate it.

An example of test data distribution would be a simulation as described in (4.1) and (4.2):

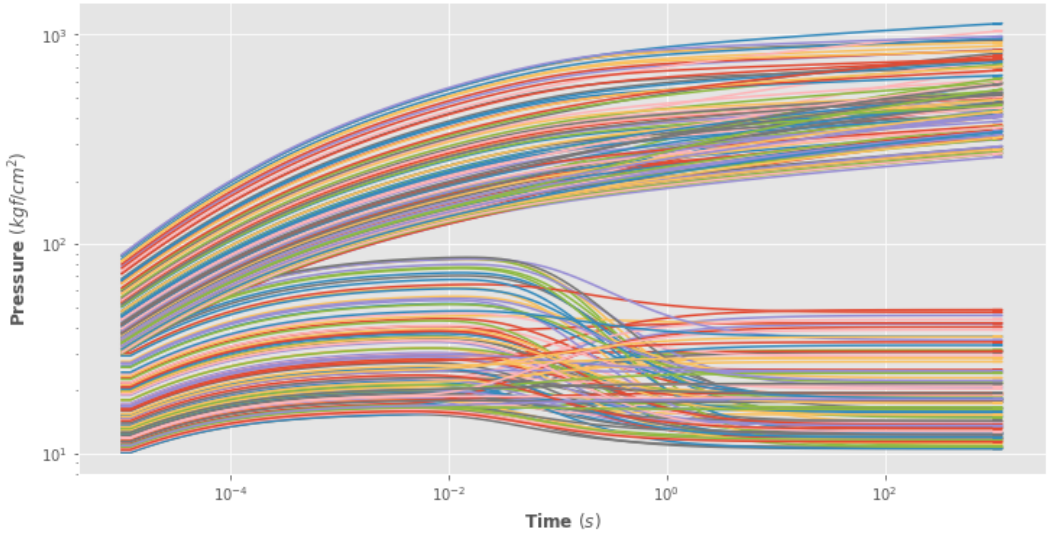

Figure 4.1: Δp and $\Delta p'$ bundle for single-phase reservoir model with 2 regions generated by uniform distribution.

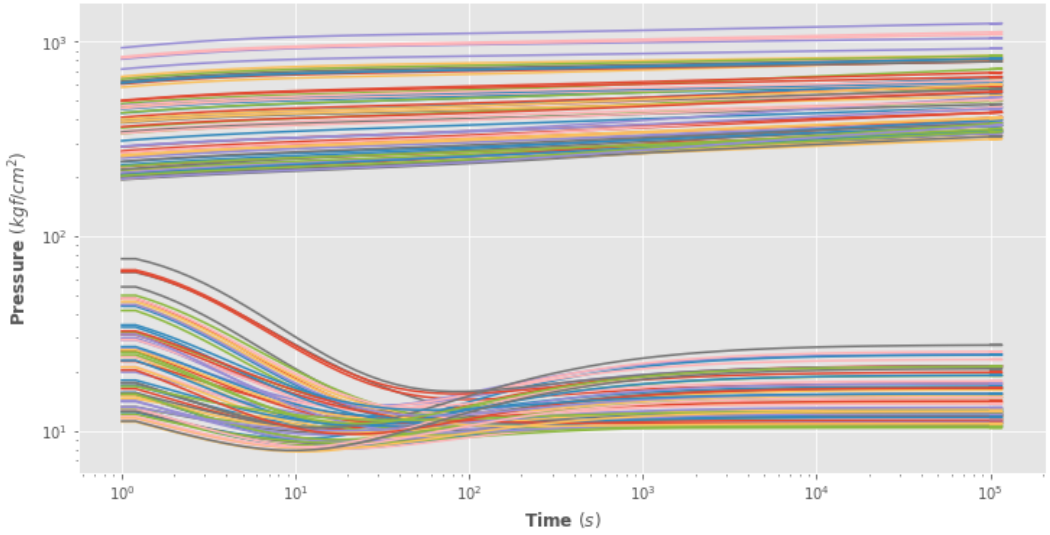

Figure 4.2: Δp and $\Delta p'$ bundle for single-phase reservoir model with 3 regions generated by uniform distribution.

Defining The reservoir properties and generating the permeabilities

To model the well-testing it's important to set up the simulation parameters, including the production rates, boundary conditions, and time step size, the following steps are necessary:

1. Define the reservoir geometry and properties, including the dimensions, shape, the rock and fluid properties. For this modelling, we will consider Porosity as 0.15, Total compressibility as 12×10^{-6} $\frac{cm^2}{kgf}$, Permeability as 600 md, Radius of wellbore as 0.33 m, Reservoir thickness $h = 32$ m, Oil viscosity $2\frac{\mathbf{kgf\ s}}{cm^2}$, Distance from centre of wellbore to outer reservoir boundary 3000 m, initial pressure 250 kgf/cm^2.

2. Define the permeability range and distribution for the training dataset. For a uniform distribution, the minimum and maximum permeability values need to be specified. For our proposal, we generate randomly 1000 values in uniform distribution $U([350, 3000])$ for the reservoir permeability. With 1000 different and ramdomly selected permeabilities, we generated 1000 different possible wellbore. The fig. 4.3 shows this random distribution.

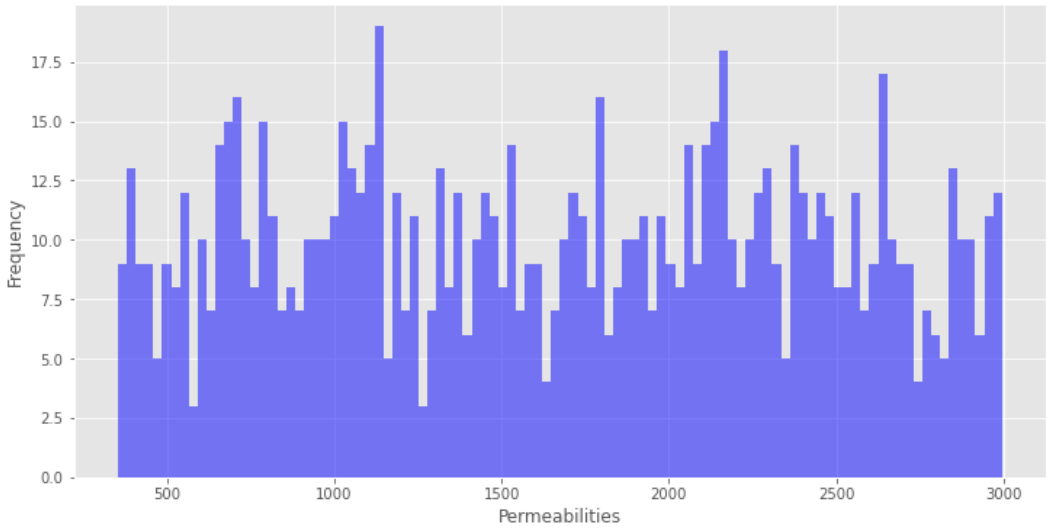

Figure 4.3: Uniform distribution of permeability values between 350 and 3000.

Let m the number of time-steps, n the number of permeabilities randomly generated, $K = \{k_1, \ldots, k_n\}$ be the set of permeabilities generated above randomly by a uniform distribution.

For each $k \in K$, define the pressure equation:

$$f_k(t) = p_{wf,k}(t) = p_i - \frac{Z}{k} \ln Wkt$$

where

$$Z = \frac{\alpha_p q B \mu}{h}$$

and

$$W = \frac{4\alpha_t}{e^{\gamma} \phi \mu c_t r_w^2}.$$

For the purpose of this work, we can consider such values constant. As we want to simulate a well-testing, we consider a time interval between 1 and 100 seconds. One can take a number m of points equally spaced between 1 and 100 to generate a vector of instants to obtain a numerical solution of the pressure. So we can define a vector

$$t_{vec} = [t_1, t_2, \ldots, t_m] = [1, \frac{100-1}{m-1} + 1, \frac{100-1}{m-1} + t_2, \ldots, 100]$$

One can consider $m = 50$, for example. With that, we have three facts:

1. if $k_i \neq k_j$ then $f_{k_i} \neq f_{k_j}$.

2. for every $k \in K$, there exists f_k such that

$$f_k(t) = p_i - \frac{Z}{k} \ln Wkt \tag{4-1}$$

and, we can define the vector F_k such that

$$F_{k_i} = f_{k_i}[t_{vec}] = [f_{k_i}(t_1), f_{k_i}(t_2), \ldots, f_{k_i}(t_m)]$$

3. Therefore, we can generate the matrix of vectors of pressure points indexed by their permeability:

$$Y = \begin{bmatrix} f_{k_1}(t_1) & f_{k_1}(t_2) & \ldots & f_{k_1}(t_m) \\ f_{k_2}(t_1) & f_{k_2}(t_2) & \ldots & f_{k_2}(t_m) \\ \vdots & \vdots & \ddots & \vdots \\ f_{k_n}(t_1) & f_{k_n}(t_2) & \ldots & f_{k_n}(t_m) \end{bmatrix} = \begin{bmatrix} F_{k_1} & F_{k_2} & \ldots & F_{k_n} \end{bmatrix}^T \tag{4-2}$$

Therefore, there're n functions indexed by each k_i, where the value of each of each function is defined in m points. Plotting this on a graph, we have a we have a bundle of functions, for $m = 50$ and $n = 1000$, considering $p_i = 300\ kgf/cm^2$. The (4-1) gives the pressure bundle for one-layer with one region generated with permeability as the only perturbation of the system showed in fig. 4.4.

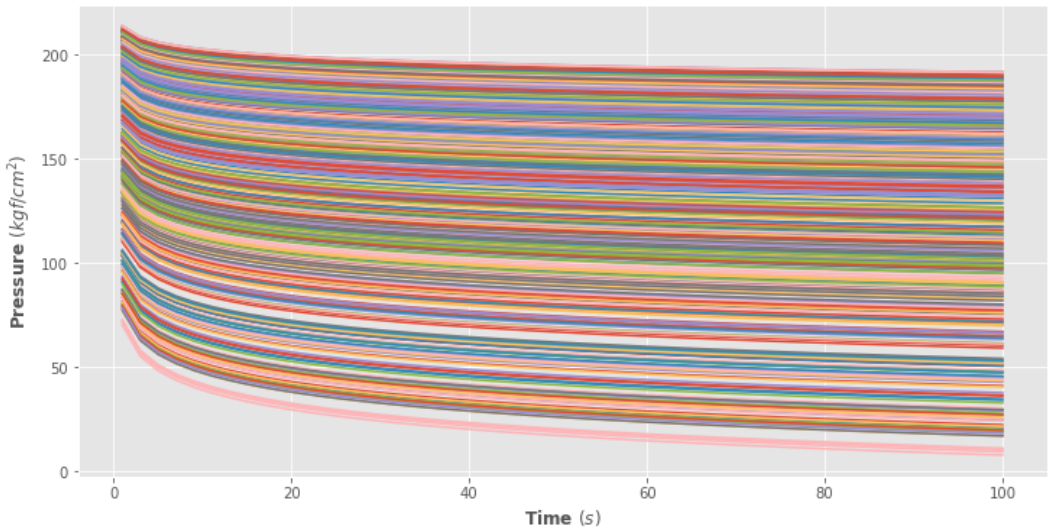

Figure 4.4: Pressure bundle for one-layer with one region with permeability as perturbation of the system.

The Bourdet's derivative

The production test uses pressure derivative function as a powerful mechanism for interpreting well test behavior. The $\Delta p_d(t)$ function as defined by Bourdet et al. [i.e., $\Delta p_d(t) = d\Delta p/dln(t)$] provides a constant value for the case of a well producing at a constant rate in an infinite-acting homogeneous reservoir. That is, $\Delta p_d(t) =$ constant during infinite-acting radial flow behavior. The Bourdet's derivative is one of the most used diagnostic in pressure transient analysis.

Therefore, it is extremely important that the Bourdet derivative of the presented model is consistent with the analytical case.

The definition of the algorithm for implementing the derivative follows below:

The Bourdet's derivative is an alternative method for calculating and smoothing a log-cycle fraction derivative. One point before and one point after a given point are used to calculate the Bourdet's derivative. Considering $x_1 < x_c < x_2$, the Bourdet's algorithm for computing the Bourdet's derivative at the point x_c is defined as:

Function `BourdetDerivative`$(x_1, y_1, x_c, y_c, x_2, y_2)$**:**

$d_1 \leftarrow \frac{(y_2 - y_c)/\log(x_2/x_c) * \log(x_c/x_1)}{\log(x_2/x_1)}$;

$d_2 \leftarrow \frac{(y_c - y_1)/\log(x_c/x_1) * \log(x_2/x_c)}{\log(x_2/x_1)}$;

return $d_1 + d_2$;

For the case of only one region, for example, there're n numerical pressure solutions for the wellbore and we obtain m points sufficiently spaced so that we can determine its Bourdet derivativ in $m - 2$ points(except extremities).

Structuring the dataframe to apply the regression model

The Equation (4-2) gives us a target structure for the model. Considering as input a column vector:

$$K = \begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix}^T = \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{bmatrix} \tag{4-3}$$

Define the learning set $\mathcal{L}$ as the set of n pairs of input vectors and output vectors $(k_1, f_1), ..., (k_n, f_n)$, where

$$k_i \in K$$

and

$$f_i = [f_{k_i}(t_1), f_{k_i}(t_2), \ldots, f_{k_i}(t_m)] \in F.$$

So, the supervised learning task can be stated as learning a function

$$\varphi : K \to F$$

from this learning set $\mathcal{L} = (K, F)$ and the goal is to find a model such that its predictions $\varphi(f)$, also denoted by the variable $\tilde{F}$, fit the analytical solution as good as possible, that is, with less error than for the Random Forest Regressor case, based on the the loss function defined as follows:

1. apply the algorithm over the pair (K, F) and let ϕ the model obtained by stated the learning function using the random forest multi-regression algorithm.

2. Generate the validation set as a set of new permeability values K_{val}, apply the model ϕ. Define

$$F_{val} = \phi(K_{val})$$

3. Verify this result with the known analytical pressure model: For each $k_i \in K_{val}$ the error for each point k_i can be calculated considering (4-1) and Euclidean metric (norm L_2)[15]: Considering

$$\delta(k_i) = \sum_{j=1}^{m} |\phi(k_i)(t_j) - f_{x_i}(t_j)|$$

Define

$$Err(x_i) = \frac{1}{m}\delta^2(x_i) \tag{4-4}$$

4. Using (4-4) and considering $K_{val} = \{x_1, \ldots, x_N\}$ one can define the loss function Δ as:

$$\Delta(K_{val}) = \frac{1}{N}\sum_{i=1}^{N} Err(x_i)$$

For each $i \in \{1, \ldots, N\}$, define the function

$$e_i : \{t_1, \ldots, t_m\} \to \mathbb{R}$$

that maps $t \mapsto \phi(x_i)(t) - f_{x_i}(t)$. Then e_i shows us how the predicted solution differs from the analytical solution over time and its modulus is the distance at each point in the solution. Applying the definitions previously presented, we have the $\{e_i\}_{i=1}^{N}$ the fig. 4.5 illustrate the difference between the analytic solutions and predicted solutions.

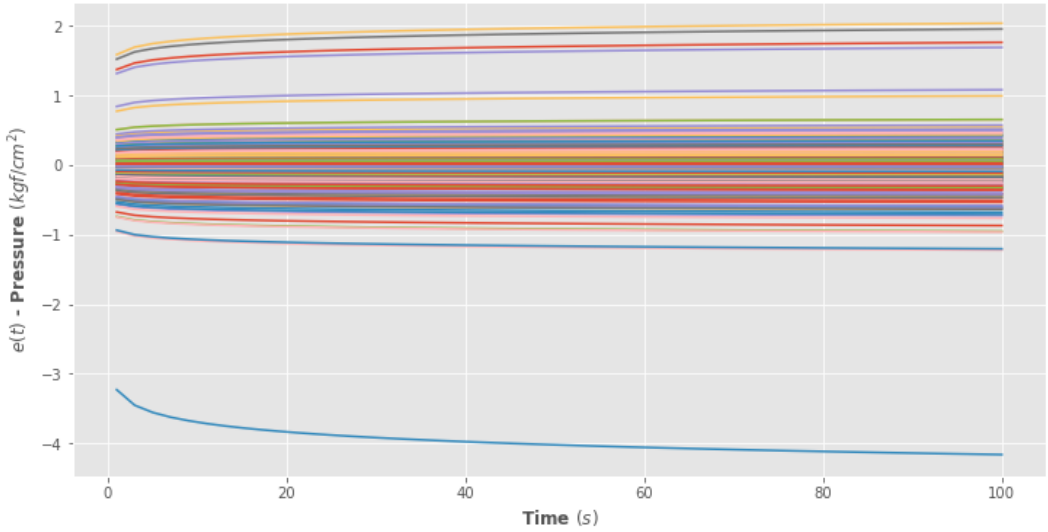

Figure 4.5: The difference $e_i(t) = \phi(x_i)(t) - f_{x_i}(t)$ between the analytic solutions and predicted solutions.

And $\{Err(x_i)\}_{i=1}^{N}$ plotted in fig. 4.6 shows us a local quadratic error, that is, it shows the local behavior of the error in the function, which complements the global analysis, the MSE.

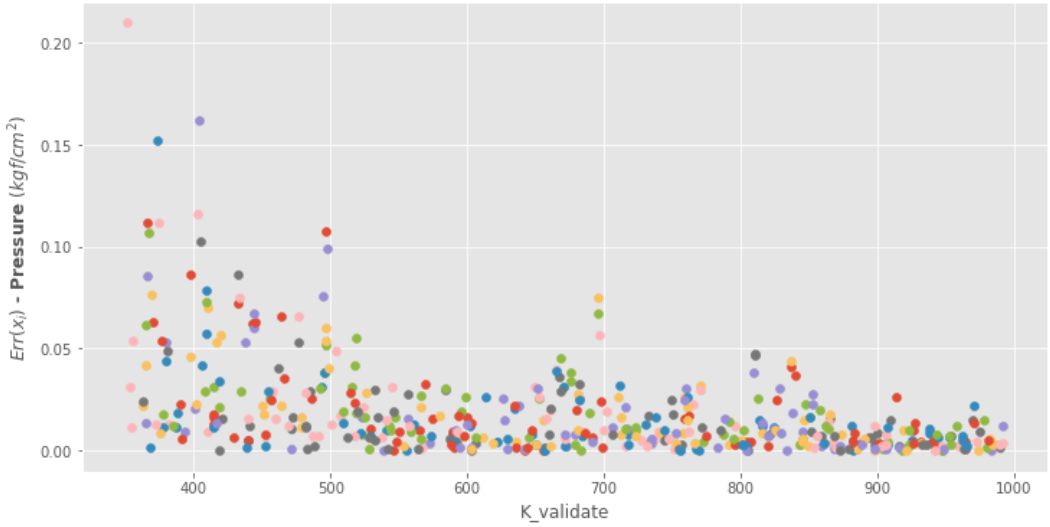

Figure 4.6: For each $x_i \in K_{val}$, this graph describes $Err(x_i)$ as $\frac{1}{m}\Big(\sum_{j=1}^{m} |\phi(x_i)(t_j) - f_{x_i}(t_j)|\Big)^2$.

If $f_k(t) = p_i - \frac{Z}{k} \ln Wkt$ then, the bourdet derivative is $df_k = A/k$, where A is a constant. Therefore, an approximately constant value is expected, unless variations due to the methods numbers and number of discrete points. The figures (4.7) and (4.8) are consistent with the expected result and show the loglog plotting graph of Bourdet derivative of Analytical Solutions Generated by K_{train} and the loglog plotting graph of Bourdet derivative of Solutions Predicted using the model ϕ at K_{val}:

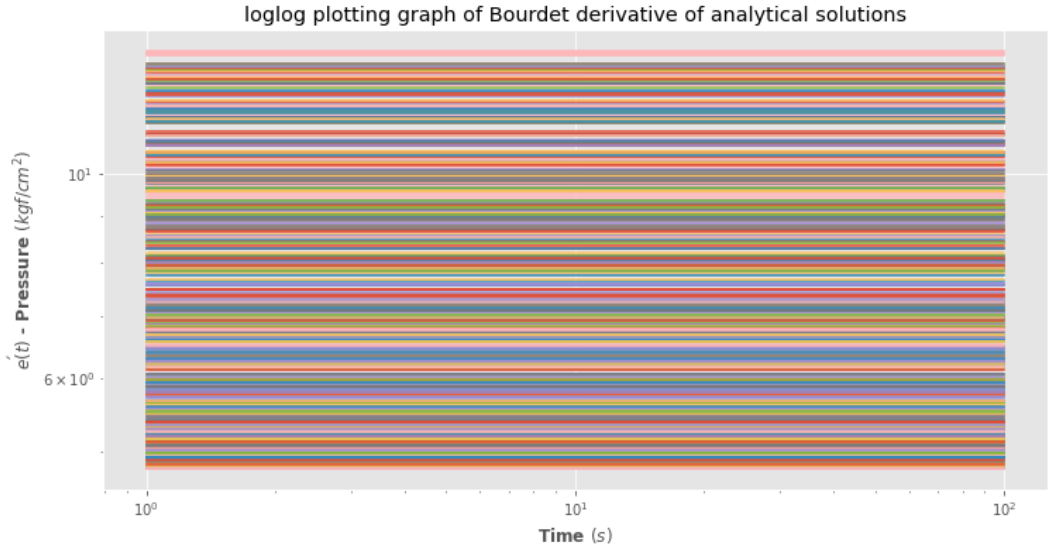

Figure 4.7: loglog plotting graph of Bourdet derivative of Analytical Solutions Generated by K_{train}.

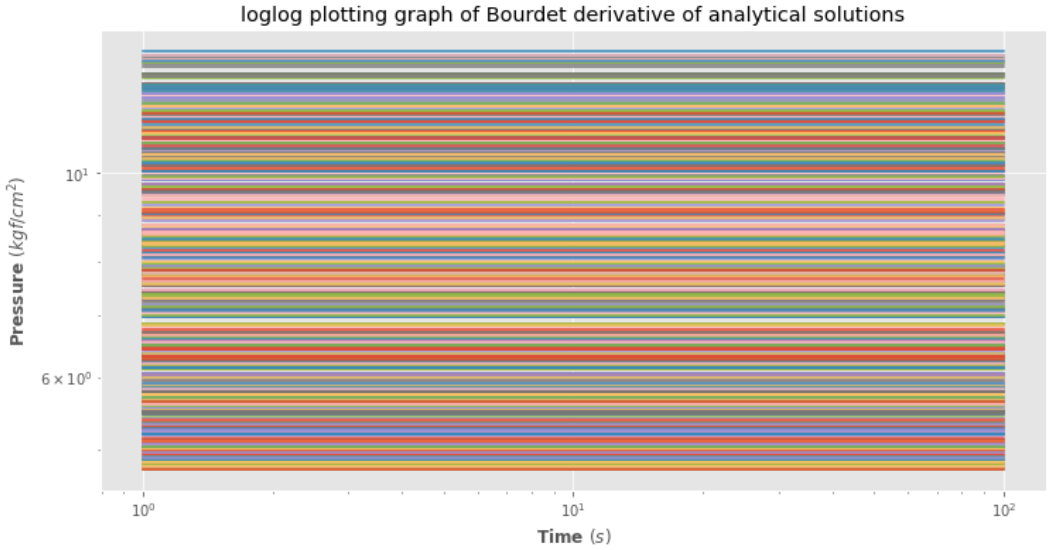

Figure 4.8: loglog plotting graph of Bourdet derivative of Analytical Solutions Predicted using the model ϕ at K_{val}.

The fig. (4.9) shows that, for the validation set, the ratio of the bourdet derivative of f_{k_i} and $\phi(k_i)$is close to 0, consistent with expectations.

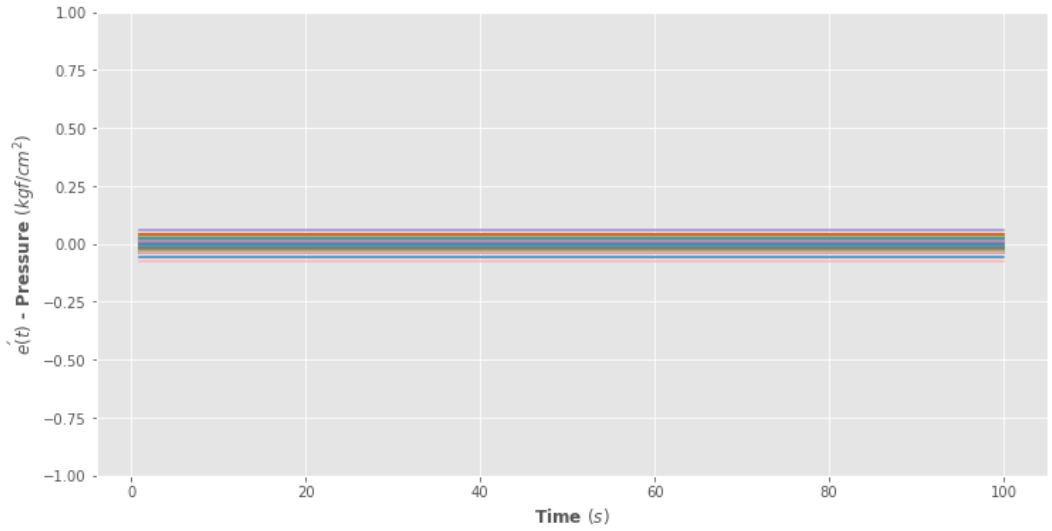

Figure 4.9: The difference between the analytical and predicted solutions as a function of k over time on validate set.

The error function applied to the derivative of both functions also shows to be close to zero at all points in the validation set, as described in fig. (4.10).

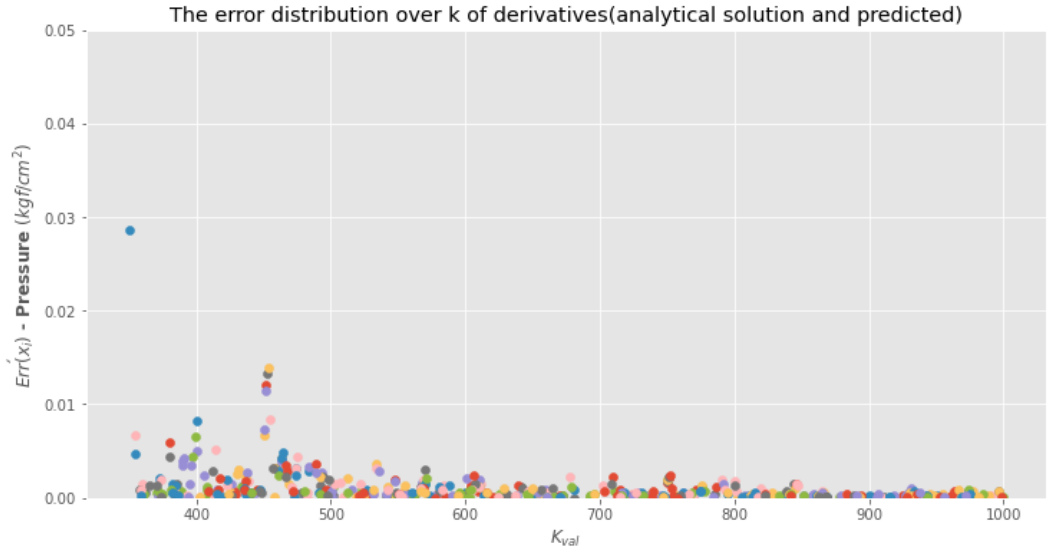

Figure 4.10: The error $Err'(x_i)$ as $\frac{1}{m}\Big(\sum_{j=1}^{m} |\phi'(x_i)(t_j) - f'_{x_i}(t_j)|\Big)^2$.

In our experiments, the results were satisfactory within the range adopted, but it is noticed that the smallest errors occur for permeabilities between 500 and 1000 in our permeability space. Regarding the set of values of the errors above, it is noticed that there are few points whose error would be greater then almost points but still within reasonable values. Finally, it is noticed that for the simple case the model has results with small errors and consistent values both in function and its derivative. As the Random Forest regressor calculates the value of the points through means and weights defined by the trees, the more training data points there are close to the validation point, the smaller the error.

4.3 Simulating reservoir with two regions

All the theory and methodology behind the problem has already been exposed above, however, we fixed the main characteristics of the reservoir with two regions, except the permeabilities in each region, to focus on its distribution. Then, applying the proposed methodology, consider two uniform distributions for permeabilities in region 1 and 2 respectively

$$K_1 = \begin{bmatrix} k_{1,1} \\ k_{2,1} \\ \vdots \\ k_{n,1} \end{bmatrix}$$

and

$$K_2 = \begin{bmatrix} k_{1,2} \\ k_{2,2} \\ \vdots \\ k_{n,2} \end{bmatrix}$$

The features is defined as

$$X = \begin{bmatrix} K_1 & | & K_2 \end{bmatrix}$$

For each pair $x_i = (k_{i,1}, k_{i,2})$, $i \in \{1, \ldots, n\}$, let f_i the analytical solution proposed in 2-13. Define

$$F = \begin{bmatrix} f_{x_1}(t_1) & \cdots & f_{x_1}(t_m) \\ f_{x_2}(t_1) & \cdots & f_{x_2}(t_m) \\ \vdots & \cdots & \vdots \\ f_{x_n}(t_1) & \cdots & f_{x_n}(t_m) \end{bmatrix}$$

So, one can define the learning set as

$$L = \begin{bmatrix} X & | & Y \end{bmatrix} = \begin{bmatrix} x_1 & | & f_{x_1}(t_1) & \cdots & f_{x_1}(t_m) \\ x_2 & | & f_{x_2}(t_1) & \cdots & f_{x_2}(t_m)) \\ \vdots & | & \vdots & \cdots & \vdots \\ x_n & | & f_{x_n}(t_1) & \cdots & f_{x_n}(t_m) \end{bmatrix}$$

For visualization purposes, consider the fig. 4.11 with permeability distribution as $(K_1, K_2) = \Big(U([350, 1000]), U([600, 2000])\Big)$:

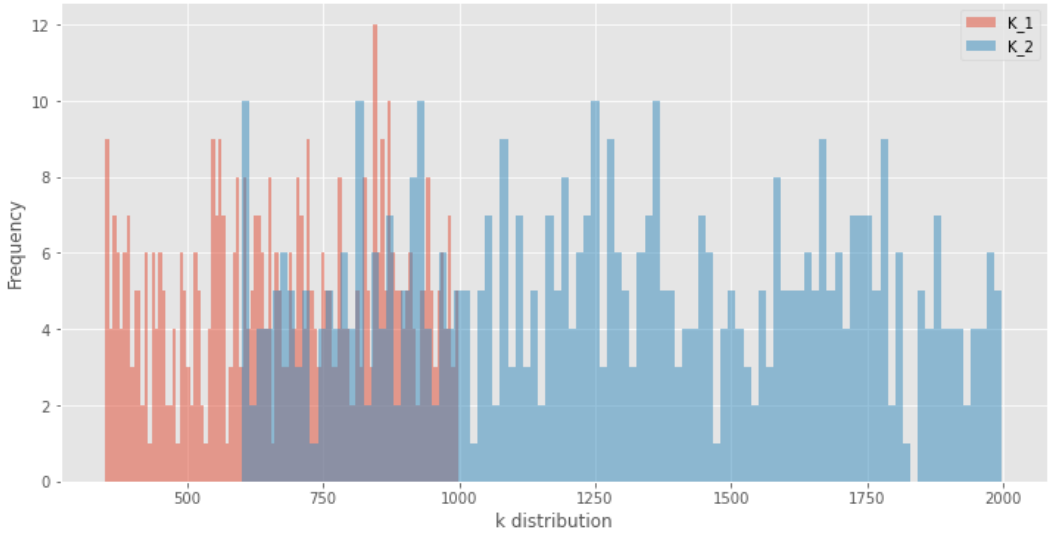

Figure 4.11: Uniform graph of distributions $K_1 = U([350, 1000])$, $K_2 = U([600, 2000])$.

This (K_1, K_2) distribution generates the bundle of solutions described in fig. 4.12

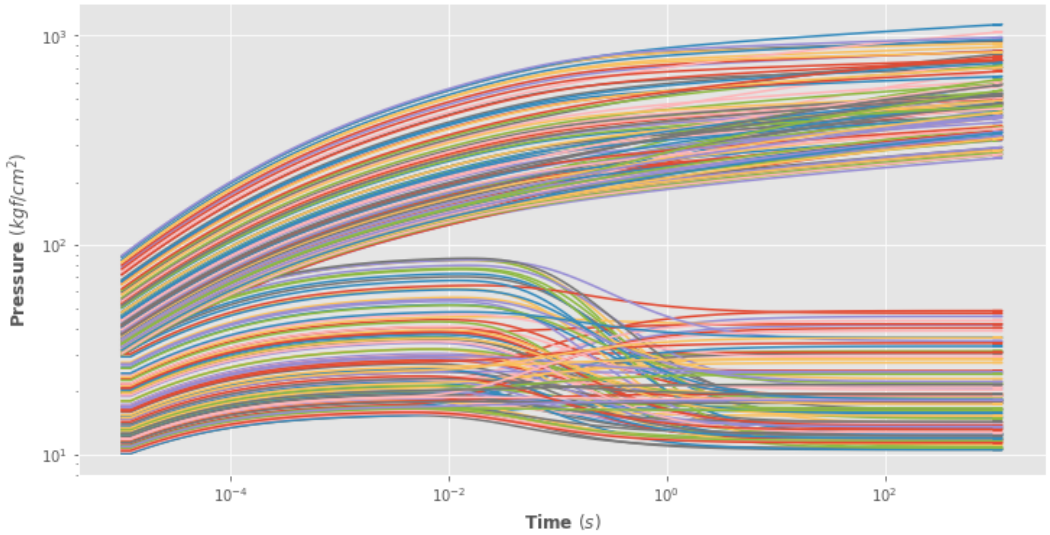

Figure 4.12: Δp bundle for single-phase reservoir model with 2 regions generated by uniform distribution $\Big(U([350, 1000]), U([600, 2000])\Big)$.

It is clear that the problem is quite similar. Applying the Random Forest Regressor algorithm, we obtain a model Φ.

To test the model, new pairs of permeabilities are generated and the validation set X_{val} is defined as the first case. Defining

$$Y_{val} = \phi(X_{val})$$

Following the same steps as first case, but considering the L_1 norm we obtain the results:

1. The validation cases were very close to the analytical one, where the errors in each input proved to be well behaved, as presented in (4.13):

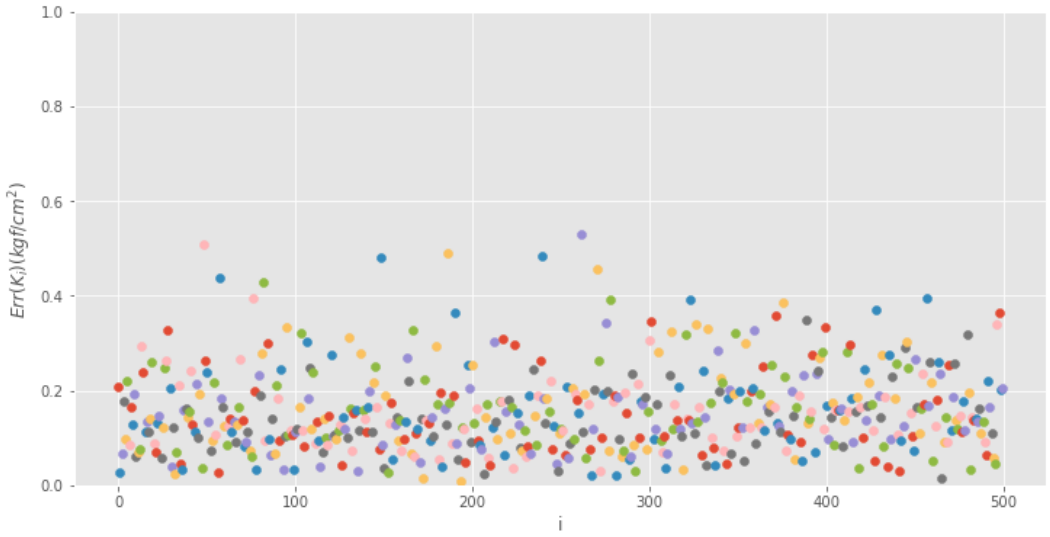

Figure 4.13: For each $x_i = (k_{i,1}, k_{i,2}) \in K_{val}$, this graph describes $Err(x_i)$ as $\frac{1}{m}\Big(\sum_{j=1}^{m} |\phi(x_i)(t_j) - f_{x_i}(t_j)|\Big)^2$.

2. Given any pair x_i in X_{val}, Applying Bourdet's derivative both in the analytical solution and in the generated model, the error Err' showed, in fig. (4.21), to be small and the functions showed to be well behaved.

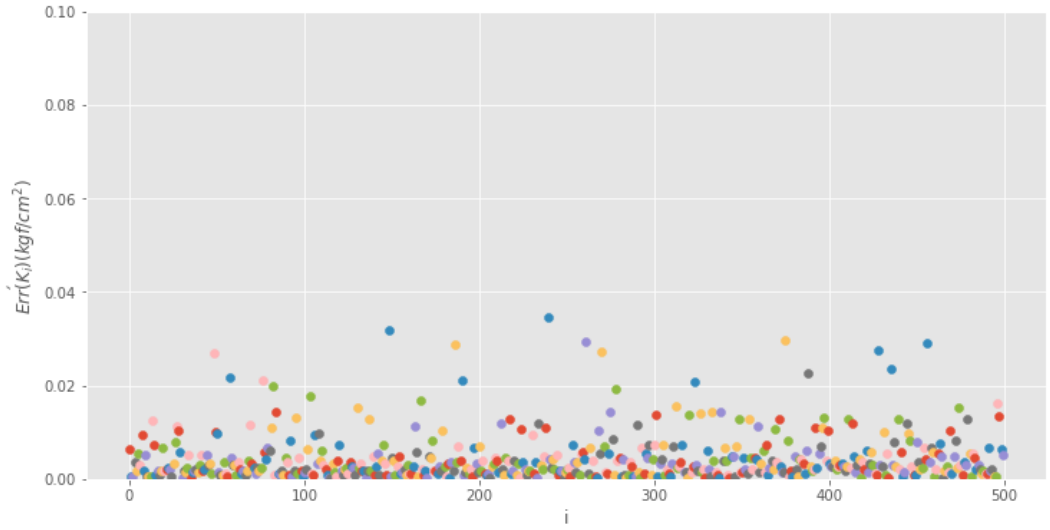

Figure 4.14: For each $x_i = (k_{i,1}, k_{i,2})$ in K_{val}, the error $Err'(x_i)$ as $\frac{1}{m}\Big(\sum_{j=1}^{m} |\phi'(x_i)(t_j) - f'_{x_i}(t_j)|\Big)^2$.

3. Furthermore, it is possible to see, for all these points, little difference when analyzing the loglog graph comparing both solution and its derivative in analytical and predicted cases. The fog (4.15) below follows from one of the points, generated randomly, however, all points have a similar behavior for the sample space that was adopted:

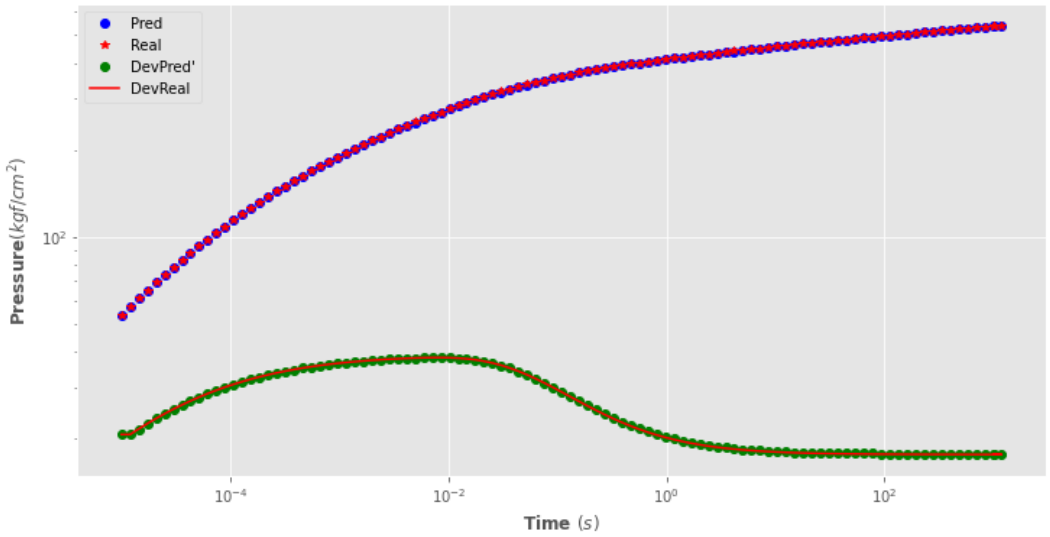

Figure 4.15: Comparing the Analytical and predicted solution, considering $x_i = (k_{i,1}, k_{i,2}) = (430, 1797)$.

4. For each $x \in X_{val}$, At all points $t \in \{t_1, \ldots, t_m\}$, on the loglog graph,

$\Delta P_x(t)$ and $\phi(x)(t)$ were close and the fig. (4.16) describes the ratio at each point.

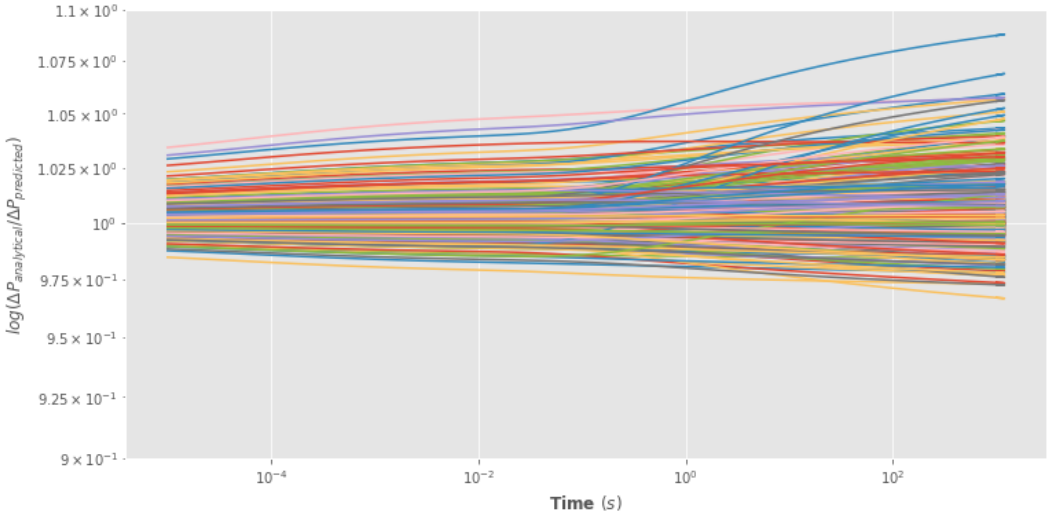

Figure 4.16: The loglog plotting graph of Analytical Solutions over Predicted solutions $log\left(\frac{\Delta p(x_i)}{\Delta \phi(x_i)}\right)$.

4.4 Simulating the reservoir with three regions

Except for permeability, the constant and known physical quantities are considered for each region in a multiregion reservoir with three regions. As was did before, for $i \in \{1, 2, 3\}$ define three normal distribution:

$$K_i = \begin{bmatrix} k_{1,i} \\ k_{2,i} \\ \vdots \\ k_{n,i} \end{bmatrix}$$

The new feature is defined as

$$X = \begin{bmatrix} K_1 & | & K_2 & | & K_3 \end{bmatrix}$$

and, for each triple $x_i = (k_{i,1}, k_{i,2}, k_{i,3})$, $i \in \{1, \ldots, n\}$, let f_{x_i} the analytical solution proposed in 2-16. Define

$$Y = \begin{bmatrix} f_{x_1}(t_1) & \cdots & f_{x_1}(t_m) \\ f_{x_2}(t_1) & \cdots & f_{x_2}(t_m) \\ \vdots & \cdots & \vdots \\ f_{x_n}(t_1) & \cdots & f_{x_n}(t_m) \end{bmatrix}$$

and the training set as

$$D_{train} = \begin{bmatrix} X & | & Y \end{bmatrix} = \begin{bmatrix} k_1 & | & f_{x_1}(t_1) & \cdots & f_{x_1}(t_m) \\ k_2 & | & f_{x_2}(t_1) & \cdots & f_{x_2}(t_m) \\ \vdots & | & \vdots & \cdots & \vdots \\ k_n & | & f_{x_n}(t_1) & \cdots & f_{x_n}(t_m) \end{bmatrix}$$

For visualization purposes, consider

$$(K_1, K_2, K_3) = \Big(U([350, 2000]), U([3000, 5000]), U([1000, 3000])\Big)$$

The fig. 4.17 show the graph of uniform distributions (K_1, K_2, K_3).

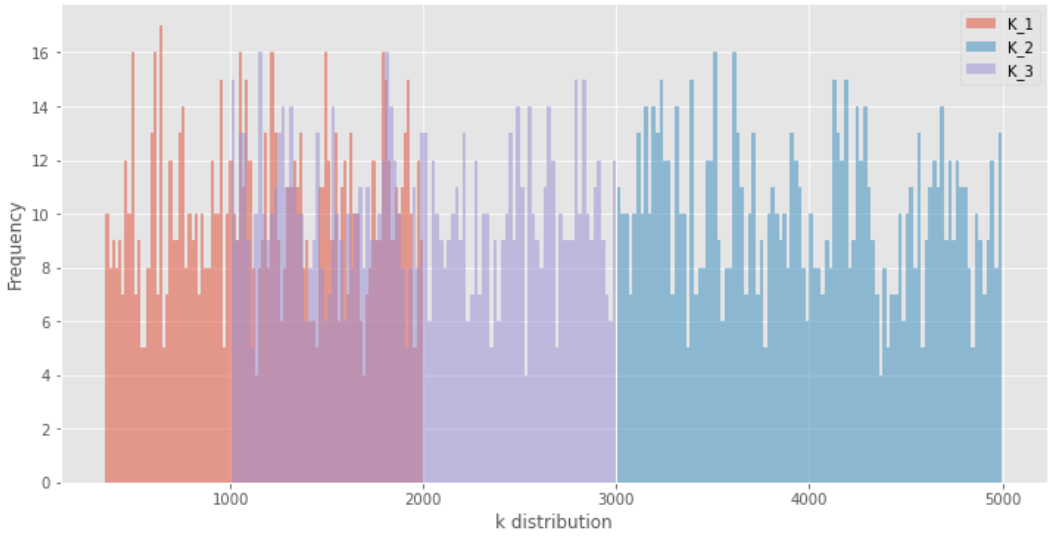

Figure 4.17: Uniform distributions of $K_1 = U(350, 2000)$, $K_2 = U(3000, 5000)$, $K_3 = U(1000, 3000)$.

Using this distribution, the bundle of analytical solutions is showed in fig. (4.18).

1. The training set is described below:

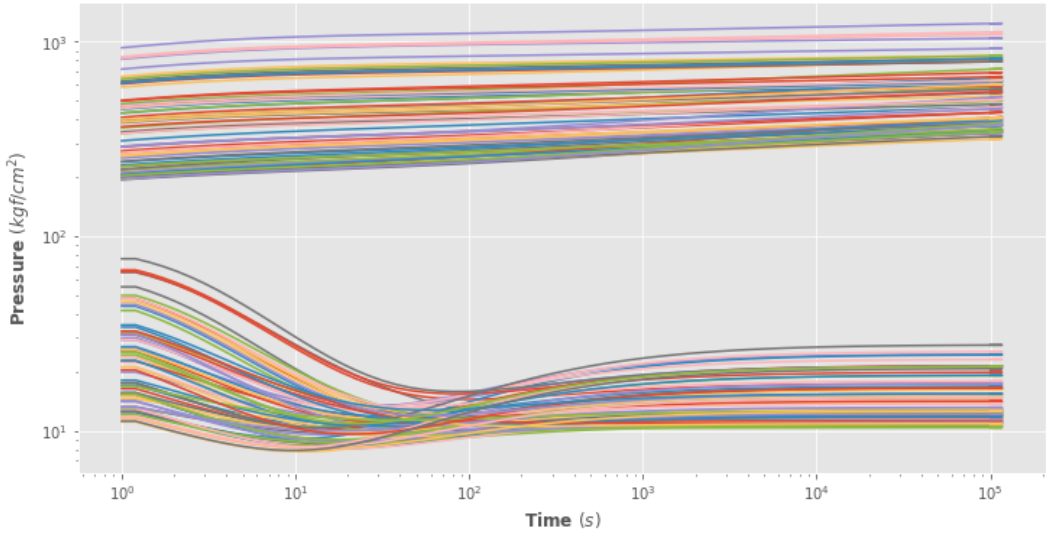

Figure 4.18: Loglog plotting of Pressure bundle for one-layer with three regions.

2. The ratio of the analytical solution and the solution predicted by the trained model is showed in fig. (4.19) as the loglog plotting graph of analytical solutions over predicted solutions $log\left(\frac{\Delta p(x_i)}{\Delta \phi(x_i)}\right)$.

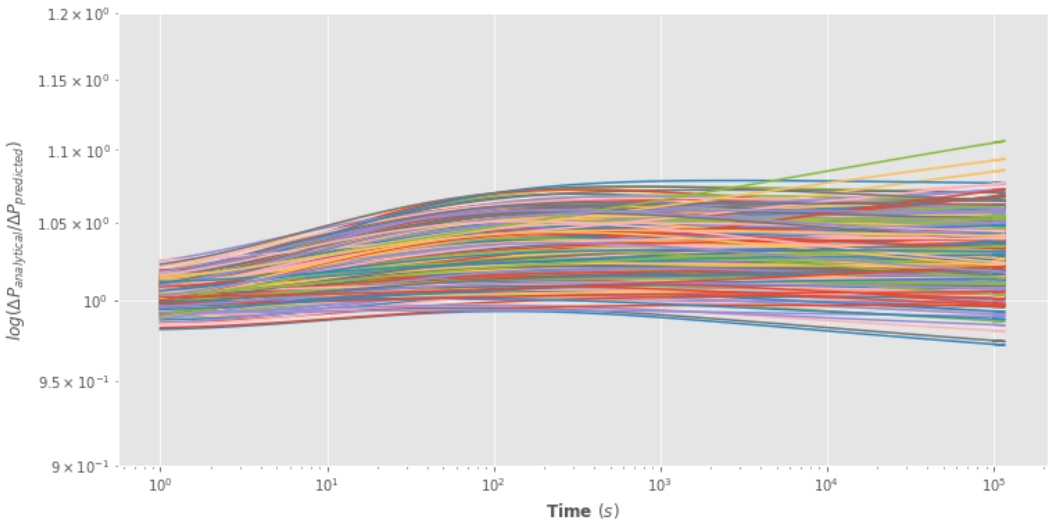

Figure 4.19: The loglog plotting graph of analytical solutions over predicted solutions $log\left(\frac{\Delta p(x_i)}{\Delta \phi(x_i)}\right)$.

3. For each $x \in X_{val}$, At all points $t \in \{t_1, \ldots, t_m\}$, on the fig. 4.20, $\Delta P'_x(t)$ and $\phi'(x)(t)$ were close and the graph below describes the ratio at each point on the fig. (4.20).

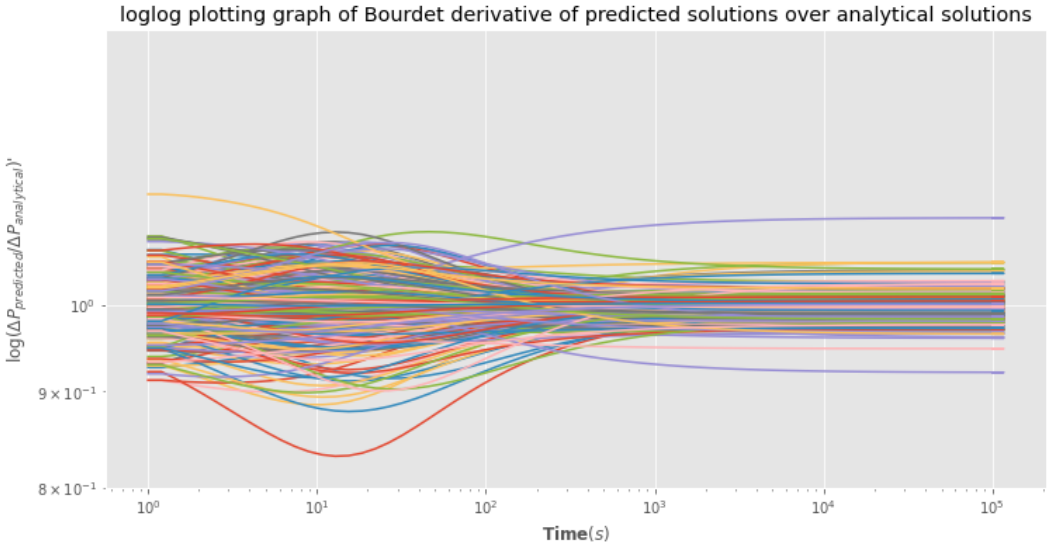

Figure 4.20: The loglog plotting graph of Analytical Solutions over Predicted solutions $log\left(\frac{dp'(x_i)}{d\phi'(x_i)}\right)$.

4. The validation set were remarkably similar to the analytical situations, with the faults in each input showing to be well behaved.

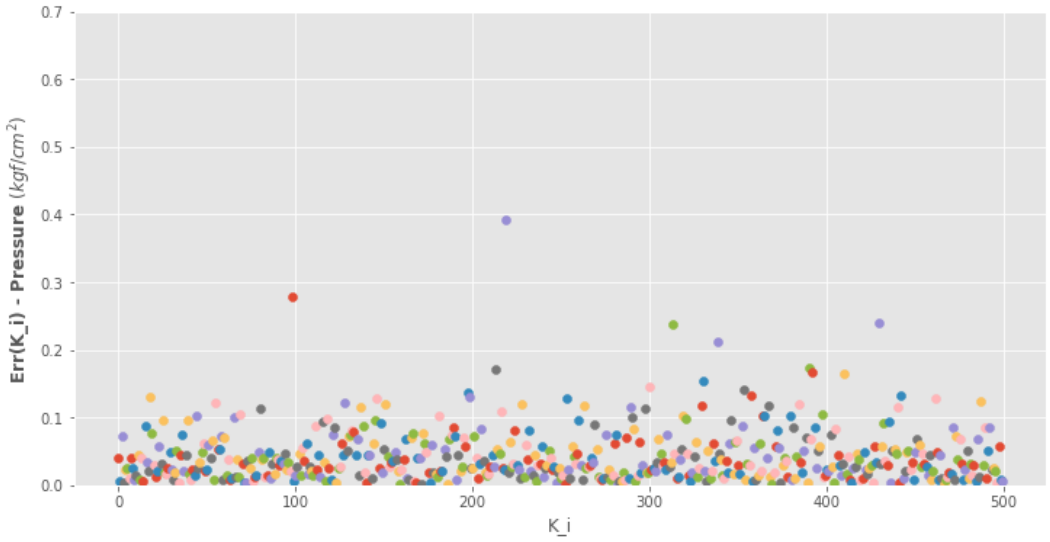

Figure 4.21: For each $x_i = (k_{i,1}, k_{i,2}, k_{i,3}) \in K_{val}$, this graph describes $Err(x_i)$ as $\frac{1}{m}\left(\sum_{j=1}^{m} |\phi(x_i)(t_j) - f_{x_i}(t_j)|\right)^2$.

5. For every pair x_i in X_{val}, using Bourdet's derivative in both the analytical solution and the produced model, the error proved to be modest, and the functions demonstrated to be well behaved in fig 4.22, where For each

$x_i = (k_{i,1}, k_{i,2}, k_{i,3})$ in K_{val}, the error is defined as

$$Err'(x_i)\frac{1}{m}\Big(\sum_{j=1}^{m} |\phi'(x_i)(t_j) - f'_{x_i}(t_j)|\Big)^2.$$

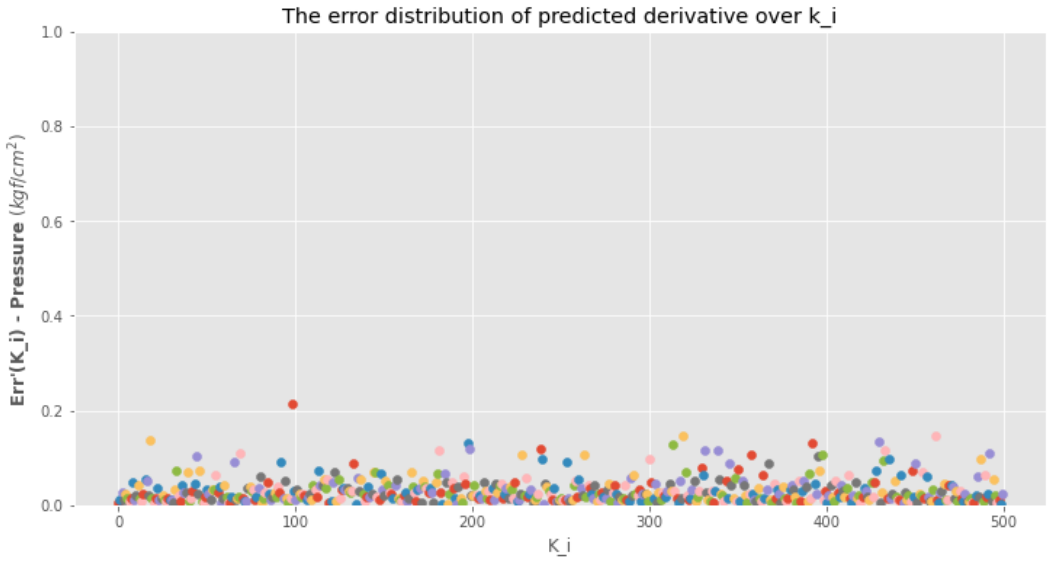

Figure 4.22: For each $x_i = (k_{i,1}, k_{i,2}, k_{i,3})$ in K_{val}, the error $Err'(x_i)$ as $\frac{1}{m}\Big(\sum_{j=1}^{m} |\phi'(x_i)(t_j) - f'_{x_i}(t_j)|\Big)^2$.

6. While evaluating the loglog graph and comparing the solution and its derivative in analytical and anticipated scenarios, it is feasible to observe that there is minimal change for all of these points. The fig. (4.23) shows a result from one of the randomly produced points. Nonetheless, all points show a similar behavior for the sample space that was used:

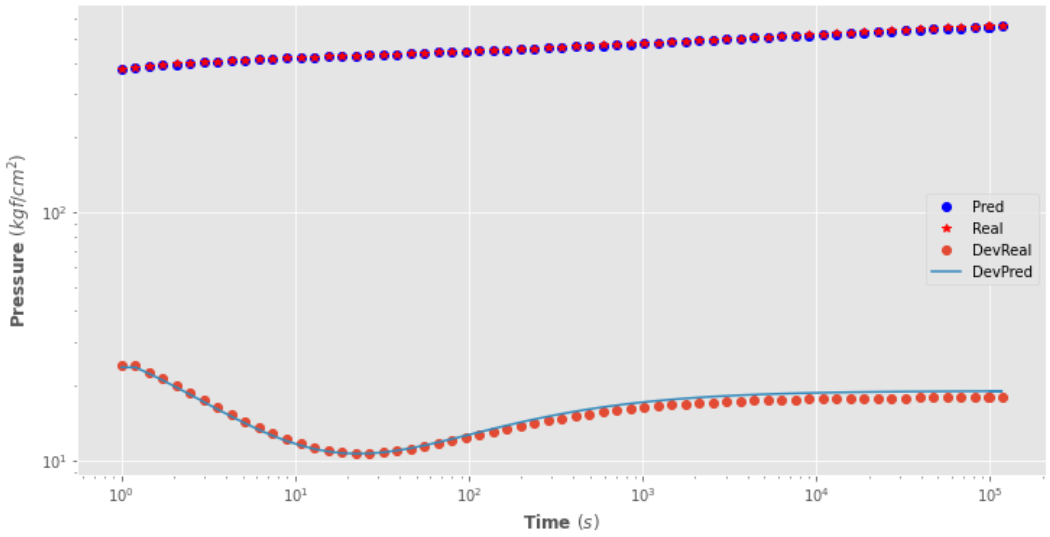

Figure 4.23: Comparing the Analytical and predicted solution, considering $x_i = (k_{i,1}, k_{i,2}, k_{i,3}) = (1471, 4261, 1837)$.

5
Conclusion

Oil and gas reservoir models are complex due to their strong heterogeneities attributes. As a result, an accurate reservoir characterization is crucial for reservoir development, monitoring, and management. Nevertheless, traditional approaches to obtaining a good characterization are based on reservoir simulation. Several solutions for simulating reservoir data use numerical analysis to compute a plausible solution for the differential equation that models the problem. These solutions can be academic, where the solution is programmed by a researcher, or commercial, with software that computes the data as a black box. This process might be slow and require robust computation capacity. Therefore, this work discusses the use of machine learning as a support for reservoir simulation processes.

The literature also proposes analytical solutions to compute data from reservoir models. This approach is also profitable and delivers reasonable approximate, and accurate solutions. As a result, to verify the challenges involved in using machine learning as an escape from traditional reservoir simulation tools, we propose using the Randon Forest regressor algorithm to compute the data for a heterogeneous reservoir model. To train the machine learning model, we create a training dataset by simulating the data using an analytical solution available in the literature for composite reservoirs. One can also use commercial simulators to create such a dataset. However, aiming to evaluate the new approach, we decided to use the analytical approach.

The experiments performed suggest that the proposed methodology may be used to reproduce a well-testing simulation. We observe that using several measures across numerous experiments helped us understand model correctness. This methodology shows that the efficiency of the simulation occurs in a large sample space and that, as long as the model is well trained and has a sufficient number of points, the goal seems to be achieved: it is possible to simulate wellbores reservoir with reasonable error for observations involving analytical solitons. The model's validation data showed tolerable errors that were consistent with the analytical model in permeabilities in the intervals of interest. This approach works reasonably for a single zone with uniform permeability as well as a medium with uniform permeability for two and three

regions.

Additionally, the suggested methodology appears to have the potential to be an alternative aiming to decrease the time and expense associated with standard well-testing procedures, which need considerable fieldwork, difficult mathematical and statistical methods, as well as time and cost. The suggested methodology may deliver accurate and efficient well-testing simulations using machine learning, which can be utilized to improve production plans, simulation for reservoir characterisation, and decision-making processes in the oil and gas sector. Furthermore, the suggested methodology may be simply incorporated into existing reservoir simulation software with historical actual data, giving reservoir engineers, students, and researchers with an extra tool for analyzing and optimizing well performance. Overall, the findings of this study show that machine learning approaches have the ability to integrate the way well testing is done and to increase the efficiency and accuracy of reservoir characterization in a short period of time.

Future Works

Future works could include testing other machine learning models, such as deep learning or reinforcement learning, to compare their performance with existing models. Another approach would be to incorporate data from a commercial simulator into machine learning algorithms, allowing for more accurate predictions of reservoir behavior. Furthermore, conducting a comparison between different machine learning methods and traditional modeling approaches would help identify which method best fits the proposed problem. If access to real data is possible, training machine learning algorithms with this data can improve the accuracy of reservoir predictions. Additionally, a hybrid simulator model could be created by mixing real and commercial simulator data to improve the accuracy of predictions while reducing computational costs. To generate more realistic models, data from a simulator could be generated and errors added to all samples. Finally, a machine learning sensitivity analysis of errors commonly found in the oil and gas industry could be conducted to better understand how these errors affect reservoir predictions and identify potential ways to mitigate them.

Bibliography

[1] ZHANG, LH., G. J. . L. Q.. **A new well test model for a two-zone linear composite reservoir with varied thicknesses.** J Hydrodyn, p. 804–809, 2010.

[2] RODRIGO, M. R.; WORTHY., A. L.. **Solution of multilayer diffusion problems via the laplace transform.** Journal of Mathematical Analysis and Applications., 444:475–502, 2016.

[3] H. C. LEFKOVITS, P. HAZERBROEK, E. E. A. C. S. M.. **A study of the behavior of bounded reservoirs composed of stratified layers.** SPE J., 1:43–58, 1961.

[5] CUTLER, A., C. D.; STEVENS, J.. **Random forests**. Ensemble Machine Learning, Springer, New York, p. 157–175, 2012.

[6] GONÇALVES, ISABEL F. A.; SILVA, T. M. D. B. A. B. P. S.. **Predicting oil field production using the random forest algorithm.** WORKSHOP DE APLICAÇÕES INDUSTRIAIS - CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES (SIBGRAPI), 35:134–139, 2022.

[8] SILVA, T. M. D.; PESCO, S. ; BARRETO JR., A. B.. **ES-MDA applied to estimate skin zone properties from injectivity tests data in multilayer reservoirs**. Computers & Geosciences, 146, 2021.

[9] GEEHAN, G. W., . P. R. E.. **Integration of geologic and engineering data in reservoir characterization.** AAPG bulletin, 78(1):22–42, 1994.

[10] BREIMAN, L.. **Bagging predictors**. Machine Learning, 24:123–140, 1996.

[11] AGGARWAL, C.. **Outlier analysis.** Springer, 2001.

[12] B. WANG, J. SHARMA, J. C. P. P.. **Artificial intelligence: A modern approach**. Prentice Hall, 2010.

[13] M. MOHRI, A. ROSTAMIZADEH, A. T.. **Foundations of machine learning**. The MIT Press, 2012.

[14] BELLMAN, R. E.. **Dynamic programming**. Princeton University Press, 1957.

[15] BREZIS, H.. **Functional analysis, sobolev spaces and partial differential equations**. Springer Science Business Media, p. 89–130, 2010.

[16] HUTTER, F., H. H. . L.-B. K.. **Automated machine learning: Methods, systems, challenges**. Springer, 2019.

[17] ENTEZARI, A., A. A. Z. R.-. N. Y.. **Artificial intelligence and machine learning in energy systems: A bibliographic perspective**. Energy Strategy Reviews, 45:101017, 2023.

[18] HASTIE, T., T. R. F. J.. **The elements of statistical learning**. Springer New York, 219 - 222:3–15, 2017.