



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Fluid Flow Simulation in Naturally Fractured Porous Media Using Deep Learning

TESIS

Que para obtener el título de
Ingeniero Petrolero

P R E S E N T A

Diego Rodríguez Pérez

DIRECTOR DE TESIS

Dr. Víctor Leonardo Teja Juárez



Ciudad Universitaria, Cd. Mx., 2026



**PROTESTA UNIVERSITARIA DE INTEGRIDAD Y
HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)**



De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado FLUID FLOW SIMULATION IN NATURALLY FRACTURED POROUS MEDIA USING DEEP LEARNING que presenté para obtener el título de INGENIERO PETROLERO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

DIEGO RODRIGUEZ PEREZ
Número de cuenta: 315121269

Dedication

I would like to dedicate this thesis to my parents, whom I owe more than words can express, you have been a constant source of strength, faith, and quiet encouragement through every challenge and triumph. Your belief in me has been both anchor and horizon, guiding my steps even when the path ahead seemed uncertain. Thank you for accompanying me during this important stage of my life.

To those whose presence has brought light, patience, and unwavering trust, and whose kindness has turned effort into purpose — thank you for reminding me that no journey is truly solitary. Every milestone carries the reflection of the hearts that have stood beside it and believed, often more than I did myself.

This work is as much yours as it is mine — a testament to shared dreams, enduring support, and the quiet power of those who believe in us even before we believe in ourselves.

Acknowledgments

To my thesis instructor Victor Leonardo Teja. Thank you so much for your patience and for accompanying me throughout the development of this thesis. Your support, guidance, dedication and time were indispensable in completing this project.

To Rosa Maria Mariscal Romero whose knowledge in artificial intelligence and dedication to teaching have inspired me to engage deeply with this field and encouraged me to pursue this research project.

I also would like to thank DGTIC-UNAM for providing access to the Miztli supercomputer cluster through project number LANCAD-UNAM-DGTIC-065, which was used to generate the synthetic data.

Contents

Abstract	vii
List of Figures	viii
1 Introduction	1
1.1 Background	1
1.2 Problem statement	2
1.3 Objectives	2
1.4 Thesis structure	3
2 Naturally Fractured Reservoirs	4
2.1 Relevance and characteristics	4
2.2 Fluids behavior	7
2.2.1 PSS Model	7
2.2.2 USS Model	8
2.3 Simulation modeling	10
2.3.1 Discrete-fracture	10
2.3.2 MINC	10
2.3.3 ECM	10
2.3.4 Dual-continuum	11
2.4 Challenges and issues	12
3 Deep Learning	13
3.1 Introduction	13
3.2 Neural Networks	14
3.2.1 Deep Feedforward Neural Network	16
3.2.2 Recurrent Neural Network	17
3.2.3 Long Short Term Memory	18
3.3 Optimization	21
3.4 Regularization	22
3.4.1 L1 and L2 Regularization	22
3.4.2 Early Stopping	22

3.5	Hyperparameters	23
3.5.1	Number of Layers and Neurons	23
3.5.2	Learning Rate	23
3.5.3	Batch Size	23
3.5.4	Number of Epochs	23
3.5.5	Activation Functions	24
4	Fractured Reservoirs	26
4.1	Data collection	26
4.2	Exploratory Data Analysis	29
4.3	Data Normalization	31
4.4	Evaluation of possible models	32
4.5	Model selection and training	33
4.6	Model evaluation	34
5	Validation and Application of Predictive Models	36
5.1	Comparative Analysis and Model Application	37
5.2	Predictive models comparison	43
5.3	Case Studies: Model Implementation and Performance	44
5.3.1	Use case 1	45
5.3.2	Use case 2	50
5.3.3	Use case 3	55
6	Conclusion	60

Abstract

NFRs represent more than 50% of reservoirs in the world. This type of reservoirs have been a source of challenging issues such as field development, well stability, drilling, and enhanced oil recovery, as a connected fracture system that can totally dominate flow patterns. Due to the strong heterogeneity in both flow properties and reservoir geomechanics, various mathematical, numerical, and discretization approaches have been developed to model and predict the hydrodynamic behavior of naturally fractured reservoirs.

Reservoir simulation has been used to replicate reservoir conditions and thus, reservoir performance. The downside is that it is computationally demanding, it is time-consuming and it requires certain level of expertise in programming and knowledge in reservoir engineering. Another issue with reservoir simulators is that they tend to be slow or they can be expensive.

To overcome these challenges, this thesis proposes an alternative approach based on the development of an AI model trained using the characteristics of the reservoir of interest. Previous studies have demonstrated the potential of ML models to successfully reproduce fluid flow responses in conventional reservoirs. Therefore, this thesis focuses on the development of a DL model aimed at replicating the results of fluid flow simulations, rather than performing a complete reservoir simulation workflow. The feasibility and potential advantages of employing a complex DL model are evaluated under different scenarios, assessing its accuracy, computational efficiency, and reliability as a surrogate tool for fluid flow simulation.

List of Figures

2.1	Naturally fractured reservoir classification system (Nelson, 2001)	6
2.2	“Schematic representation of the MINC model” (Wu, 2016).	10
2.3	Schematic of conceptual double-porosity model of fractured reservoirs (Warren & Root, 1963)	11
3.1	”Schematic representation of a neuron” (Roos, 2019)	15
3.2	example of a feed-forward neural network with three layers (input, one hidden, and output) and three neurons per layer (Buduma, 2017)	16
3.3	An unrolled recurrent neural network (Olah, 2015)	17
3.4	An unrolled RNN handling short-term dependencies (Olah, 2015)	17
3.5	LSTM state cell (Olah, 2015)	18
3.6	LSTM forget gate (Olah, 2015)	19
3.7	LSTM update gate (Olah, 2015)	19
3.8	Union of information to be updated in the cell state (Olah, 2015)	20
3.9	LSTM output gate (This gate filters the information to be passed to the next layer) (Olah, 2015)	20
3.10	Gradient Descent algorithm (Ghassemzadeh, 2020)	21
3.11	Sigmoid activation function (Buduma, 2017)	24
3.12	ReLU activation function (Buduma, 2017)	25
3.13	Tanh activation function (Buduma, 2017)	25
4.1	Three-dimensional simulation homogeneous model	27
4.2	Reservoir parameters used in the homogeneous simulation model	28
4.3	First rows of the dataset	29
4.4	Columns of the dataset	29
4.5	Checking data types	30
4.6	Checking for missing values	30
4.7	Descriptive statistics of the dataset	30
4.8	Distribution of the dataset	30
4.9	Relation between the time period and Pwf num [Psi]	30
4.10	Grid Search workflow (Scikit-Learn, n.d.)	32
4.11	Cross-validation behavior (Scikit-Learn, n.d.)	32

5.1	Overfitting in Regression models (Sahin, 2025)	36
5.2	DFNN predictions vs real values	37
5.3	DFNN predictions vs real values	38
5.4	DFNN predictions vs real values of target feature	38
5.5	LSTM predictions vs real values	39
5.6	LSTM predictions vs real values	39
5.7	LSTM predictions vs real values of target feature	40
5.8	XGBoost predictions vs real values	41
5.9	XGBoost predictions vs real values	41
5.10	XGBoost predictions vs real values of target feature	42
5.11	Use case 1 simulator graph	46
5.12	DFNN predictions for use case 1	46
5.13	LSTM predictions for use case 1	47
5.14	XGBoost predictions for use case 1	48
5.15	Use case 2 Simulator graph	51
5.16	DFNN predictions for use case 2	51
5.17	LSTM predictions for use case 2	52
5.18	XGBoost predictions for use case 2	53
5.19	Use case 3 simulator graph	56
5.20	DFNN predictions for use case 3	56
5.21	LSTM predictions for use case 3	57
5.22	XGBoost predictions for use case 3	58

Acronyms

Adam	Adaptive Moment Estimation.	22
AI	Artificial Intelligence.	vii, 1, 13
ANNs	Artificial Neural Networks.	1
DFNs	Deep Feedforward Networks.	16
DL	Deep Learning.	vii, 1, 13
ECM	Effective-continuum method.	v, 10
EDA	Exploratory Data Analysis.	29
GD	Gradient Descent.	21
GUI	Graphical User Interface.	60
HPC	High-Performance-Computing.	1
LSTM	Long Short-Term Memory.	18
MAE	Mean Absolute Error.	34
MAPD	Mean Absolute Percentage Deviation.	34
MAPE	Mean Absolute Percentage Error.	34
MINC	Multiple INteraction Continua.	v, 10
ML	Machine Learning.	vii, 1, 13
MLPs	multilayer perceptrons.	16
MSE	Mean Square Error.	34

NFRs Naturally Fractured Reservoirs. vii, 1, 4

PSS Pseudo Steady-State. v, 7

Pwf Flowing Bottomhole Pressure. 1

ReLU Rectified Linear Unit. 24

REV Representative Elementary Volume. 11

RMsp Root Mean Square Propagation. 22

RNN Recurrent Neural Network. 16

RNNs Recurrent Neural Networks. 17

SGD Stochastic Gradient Descent. 21

Tanh Hyperbolic Tangent. 25

USS Unsteady State. v, 8

Introduction

1.1 Background

The main goal of the oil and gas industry is to discover and exploit oil and gas reserves to satisfy global demand, although to plan a recovery strategy there are many challenges and risks that can make it such a complicated task; one part of these risks can be treated by reservoir engineers who rely mainly in reservoir simulation, which plays a vital role due to its use from the planning stages to the understanding of the reservoir behavior, facilitating detailed analysis, and creating efficient recovery processes.

Reservoir simulation integrates principles from physics, mathematics, reservoir engineering, geoscience, and computer programming to model the hydrocarbon reservoir performance under various operating conditions. It has been studied through the years, and with the development of more powerful hardware and more advanced techniques such as finite differences and finite elements methods, now a days it has become a standard in oil and gas engineering problems as a result of the better precision of the simulations.

A growing area of interest in the reservoir simulation is Naturally Fractured Reservoirs (NFRs). These types of reservoirs have been studied since the 1960s and also represent a more difficult challenge as a consequence of the existence of dual porosity systems.

Despite the modernization of simulators with High-Performance-Computing (HPC), the integration of new techniques into the oil and gas industry such as Artificial Intelligence (AI) has brought more possibilities for the optimization of possible processes. The incorporation of Machine Learning (ML) and Deep Learning (DL) algorithms especially, Artificial Neural Networks (ANNs) have permitted to obtain accurate simulation models. These techniques represent new opportunities to improve oil and gas production and thus, improve decision making in the industry.

The main focus of this thesis is to examine the feasibility of using AI models as surrogate models for predicting the Flowing Bottomhole Pressure (P_{wf}) obtained from numerical reservoir simulations .

1.2 Problem statement

Naturally fractured reservoirs present a significant challenge in reservoir simulation mainly because of the complexity of the interaction between fractures, the rock matrix and the effects of fractures are poorly understood.

Traditional numerical simulation models, like dual-porosity methods, although widely used, have limitations when it comes to accurately capturing fluid flow through highly heterogeneous fracture systems. These limitations can lead to inaccurate production predictions which affects decision making in field development projects.

Moreover, the variability of geological properties, such as fracture permeability and porosity, introduces additional uncertainties in modeling. To address these challenges, more advanced simulation techniques have been explored.

This thesis proposes the use of DL models as a complementary tool to traditional numerical simulation models. The implementation of ANNs will allow learning complex patterns from historical production data and reservoir properties, thus improving the accuracy of predictions in naturally fractured reservoirs. The main hypothesis is that ANNs can reduce simulation time in a fractured system obtaining very good accuracy, providing a competitive advantage in production optimization and reservoir management.

1.3 Objectives

To develop a simple computational tool applying AI with the objective of predicting pressure and saturation distributions during fluid flow in fractured porous media within homogeneous unconventional gas reservoirs.

Specific Objectives

1. Review of the literature on Deep Learning applications for bottom hole pressure prediction applied to unconventional or naturally fractured reservoirs.
2. Realize several fluid flow simulations in fractured porous media using commercial simulators or of own development.
3. To analyze the performance of the computational tool and theoretically assess the feasibility of its application for fluid flow characterization in a reservoir.
4. Conduct a computational performance comparison between a deterministic simulator and trained models.

1.4 Thesis structure

In Chapter 1 a brief description about reservoir simulation and the needs for new solutions in the simulation of naturally fractured reservoirs that can be proposed as a complement during oil and gas projects is presented.

Chapter 2 highlights the basic concepts for understanding the characteristics of naturally fractured reservoirs, fluid flow in NFR, the models used to model this type of reservoirs, and the challenges in general during the simulation and modeling of NFR.

In chapter 3 we introduce the core concepts of how modern DL models work so that the user can understand the capabilities of each model, what options are available to tune the models, and what their main advantages and disadvantages are.

Chapter 4 indicates a typical process during a Data Science project, ranging from data collection, understanding our data, cleaning it, best way for training and selecting a model, to the evaluation of making predictions with unseen data.

Chapter 5 evaluates the models in the train and test set and shows the comparative results between a numerical NFR simulator and the trained algorithm.

Finally, in Chapter 5 conclusions and limitations of the thesis are brought.

Naturally Fractured Reservoirs

2.1 Relevance and characteristics

According to (Aguilera, 1995) "A naturally fractured reservoir is a reservoir which contains fractures created by mother nature. These natural fractures can have a positive or a negative effect on fluid flow."

(Nelson, 2001) defines naturally fractured reservoirs as "subsurface rock formations that contain fractures formed by natural geological processes, where the fractures act as pathways for fluid flow, and the rock matrix acts as the primary storage of fluids."

(Bourbiaux, 2010) approaches NFRs from a simulation perspective emphasizing that fractures significantly contribute to both the storage and flow properties. These fractures can substantially enhance the permeability of the reservoir and influence recovery methods, making their understanding between matrix and fracture systems interaction essential for designing efficient recovery strategies.

Based on the definitions above we can conclude that a natural fractured reservoir is a reservoir which contains fractures created by geological processes, the fractures can affect or contribute to fluid flow. These reservoirs are characterized by their dual-porosity nature where fractures act as conduit for fluid flow and the rock matrix serves as the main storage space.

Fracture Classification

A fracture is a break or discontinuity in a rock formation caused by stress that exceeds the rock's strength. These breaks can occur due to various geological processes like tectonic activity, thermal stress, or compaction (Nelson, 2001).

Main elements:

- 1) Fracture aperture
- 2) Fracture trace
- 3) Fracture orientation

The classification of fractures in naturally fractured reservoirs has been widely discussed in the literature. Several authors have proposed different criteria based on origin or stress conditions. The following fracture classification is primary based on the classifications presented by (Stearns & Friedman, 1972) and (Nelson, 2001).

Shear Fractures: They are the result when rocks are subjected to stresses that produce a displacement between the rock blocks and the fracture plane. Shear fractures form when the three principal stresses acting on the rock are compressive.

Extension Fractures: In this type of fractures, the rock mass undergoes deformation due to stretching or dilation processes that generate fractures roughly perpendicular to the minimum principal stress

Tension Fractures: Similar to extension fractures, although this type of fractures include at least one tensile stress component .

Tectonic Fractures: Produced by large-scale forces acting on the Earth's crust. These fractures commonly occur as networks associated with geological structures such as faults and folds. In a fault fracture system, shear stress is the most common type. Fractures associated with faults exhibit complex patterns but they often form geometries associated with the deformation history of the rock.

Regional Fractures: Fracture systems characterized by long fractures exhibiting little change in orientation over their length. These fractures have a simpler and more consistent geometry.

Contractional Fractures: Set of fractures associated with a general bulk volume reduction throughout the rock. These fractures are the result from desiccation, syneresis, thermal gradients and mineral phase changes. The most common fractures of this type are mud cracks.

Natural Fracture Reservoirs Classification

(Nelson, 2001) presents an expansion that of proposed in (Hubbert & Willis, 1955). The classification consists into four categories based on the concept that the fracture system provides a positive impact in fluid flow, he assumes that this classification has to occur once the reservoir properties of the fracture system are determined.

Type I: In this type, fractures are crucial as they provide essential permeability and storage capacity of the rock.

Type II: Here, fractures provide essential permeability.

Type III: The fractures assist to the reservoir permeability.

Type IV: These type of fractures create significant reservoir anisotropy and they don't provide additional porosity or permeability.

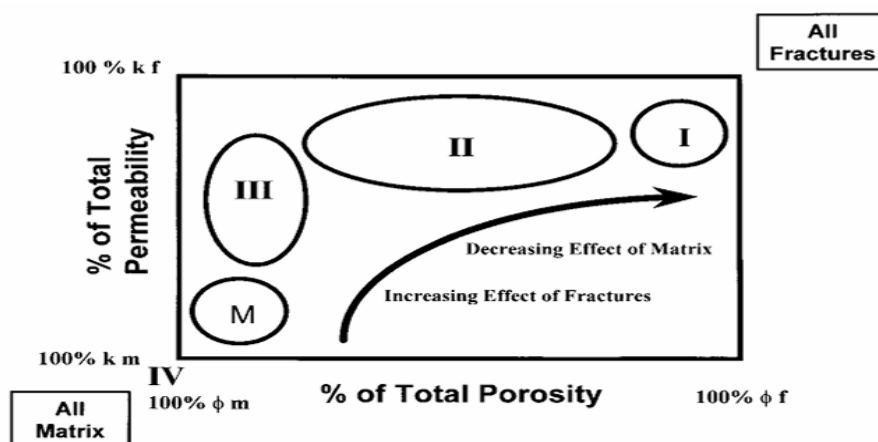


Figure 2.1: Naturally fractured reservoir classification system (Nelson, 2001)

2.2 Fluids behavior

2.2.1 Pseudo Steady-State (PSS) Model

Barenblatt Model

In 1960 flow models for naturally fractured reservoirs began to be formulated with the work of Barenblatt & Cols. This flow model considers the flow between the matrix and the fracture as pseudo stationary, physical concepts of fluid transfer in two media was founded under the assumptions of fractured rock basic fluid transfer equation.

The model presents a fractured system with two porous media in which the first are the fractures and the second are the matrix blocks, as reservoir production progresses, the pressure difference between the matrix and fractures decreases until a state is reached where fluid flow becomes constant.

The Barenblatt model captures how pressure differences tend to equilibrate in the pseudo-stationary flow phase, providing a simplified and useful approach for analyzing production at this late stage. It is especially useful for simulating reservoirs where production has reached a constant level and matrix-to-fracture flow is not expected to vary significantly. (Barenblatt et al., 1960)

Warren and Root Model

Published in 1963, the Warren and Root model represents a fundamental advance in the simulation of naturally fractured reservoirs, this work is based on the model of Barenblatt et al. The authors define two types of porosities in the reservoir: primary and secondary porosities. (Warren & Root, 1963)

1-) The medium with primary porosity (matrix) is homogeneous and isotropic and is located within a rectangular parallelepiped-shaped array.

2-) The secondary porosity (fracture) is continuous, uniform and orthogonal with orientation parallel to the main permeability axes.

This model was designed to capture how fluid flow behaves distinctly in a medium containing a network of connected fractures and a matrix that acts as a storage reservoir.

λ represents the interporous matrix-fracture transfer flux and the heterogeneity of the system.

ω represents fracture fluid storage, fracture expansion and fracture flow capacity

2.2.2 Unsteady State (USS) Model

Kazemi Model

Kazemi introduces a new flow behavior considering a finite circular reservoir with a centrally located well and two distinct porous regions: matrix and fracture, the matrix which has high storage but low flow capacity produces towards fractures which has low storage but high flow capacity and fractures produce to the wellbore, the flow throughout the reservoir is considered unsteady-state and single-phase.

This model is a special case from the Warren and Root model, the model considers two fracture distributions in the reservoir. 1) The fractures are horizontal in the first case. 2) In the second case the fractures are replaced by an equivalent set of horizontal fractures, horizontal fractures were chosen so the equivalent flow in fractures would become radial and converged to the wellbore.(Kazemi, 1969)

General assumptions of this model:

- * Flow occurs in both radial and vertical directions
- * Flow is unsteady state
- * The reservoir is horizontal and the matrix and the fracture are each homogeneous and isotropic
- * There is single-phase flow
- * The well is centrally located in a finite circular reservoir

Cinco-Ley and Samaniego Model

This model is based on the transient matrix flow formulated by de Swaan. This model uses three dimensionless parameters to correlate the behavior of a reservoir. It also shows that another semilog straight line of one half of the correct line could exist during the transition period in addition to the two classical parallel semilog straight lines.(Samaniego & Cinco Ley, 1982)

The model considers:

- * Unsteady-state radial flow of a slightly compressible fluid in a double porosity system
- * Assumes that the medium possesses a bulk fracture permeability and a bulk fracture porosity
- * Flow towards the well occurs only through fractures
- * Flow from matrix to fractures is under unsteady-state conditions

Streletsova Model

The author presented a model to model the fluid flow behavior between the matrix and the natural fracture considering a similar behavior to the model presented by (DeSwaan, 1976). To reach this the author discusses two assumptions on response of matrix blocks to pressure changes at the fracture-matrix interface.

The first assumption, used in the Warren and Root model considers matrix to fracture flow governed by a linear-diffusion process. The second assumption, employed in the gradient-flow model, considers matrix -to-fracture flow governed by a linear-diffusion process.(Streletsova,1983)

The distributed-parameter or "gradient" flow model predicts transitional pressure which, when plotted vs. logarithm of time The average gradient is considered a flow distribution parameter because the matrix pressure and its flow are not distributed linearly along the normal direction of the fracture plane. The matrix flow is proportional to the pressure gradients and storage capacity, which are not constant but depend on spatial position, thus influencing the response time.

2.3 Simulation modeling

2.3.1 Discrete-fracture

The explicit or discrete-fracture modeling approach represents each individual fracture within the model domain and directly simulates flow through each fracture along with fracture–matrix interactions. This approach is highly detailed and rigorous but is computationally demanding and requires extensive data on fracture and matrix geometry and spatial distribution—information often lacking for natural fractures at field sites.

Discrete-fracture modeling is particularly challenging for multiphase flow and thermal processes due to additional multiphase flow properties needed for both the fracture and matrix systems across the reservoir (Wu, 2016).

2.3.2 Multiple INteraction Continua (MINC)

Developed by Pruess and Narasimhan (1985), MINC is an extension of the dual-continuum model. It rigorously handles transient flow between fractures and the matrix in fractured reservoirs by accounting for gradients in pressure, temperature, and concentration. By subdividing matrix blocks near fractures, MINC allows for a more accurate simulation of fluid and heat flow in porous fractured media.

MINC is especially useful for scenarios with extended transient flow periods, such as in cases with low matrix permeability or large matrix blocks, where the dual-porosity model falls short. This method supports multiphase flows with variable compressibility and phase transitions, using an integrated finite difference approach (Wu, 2016).

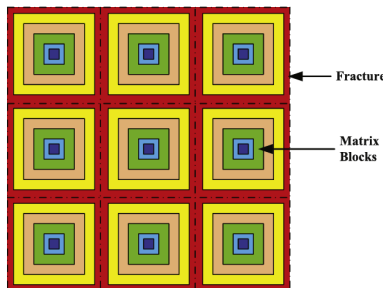


Figure 2.2: “Schematic representation of the MINC model” (Wu, 2016).

2.3.3 Effective-continuum method (ECM)

ECM represents fractures and rock matrix as a single effective continuum unlike dual-continuum or multiple-continuum models, which treat fractures and the rock matrix as separate interacting systems. It is often used for modeling fracture–matrix flow due to it is computationally efficient, and it has low data requirements. (Wu, 2016)

2.3.4 Dual-continuum

The dual-continuum model was introduced by Barenblatt et al. (1960) and Warren and Root (1963), offers a simpler, computationally efficient alternative to the discrete-fracture approach for modeling fluid flow in fractured reservoirs. This model uses a Representative Elementary Volume (REV) concept, which captures many fractures or fracture networks within a local continuum. It has been widely used for simulating fluid flow, heat transfer, and chemical transport in petroleum, geothermal engineering, and groundwater studies.

Key variations of this model include the double-porosity and dual-permeability concepts. In the double-porosity model, the fracture system allows global flow, while matrix blocks act as localized sources or sinks without direct matrix-to-matrix flow. The dual-permeability model extends this by allowing flow between both fractures and matrix blocks, representing fractures and the matrix as separate grid blocks connected locally. Both models use a pseudosteady-state assumption to simplify fracture-matrix interactions, and these approaches are integrated into most commercial reservoir simulators (Wu, 2016).

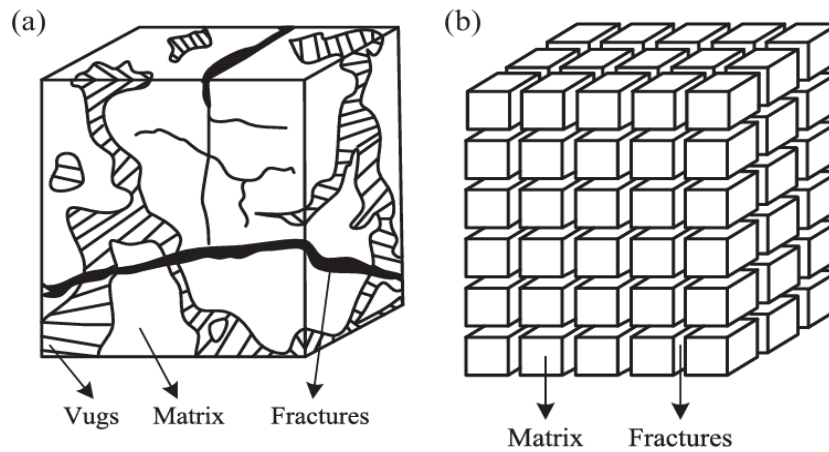


Figure 2.3: Schematic of conceptual double-porosity model of fractured reservoirs (Warren & Root, 1963)

2.4 Challenges and issues

Naturally fractured reservoirs present significant challenges in reservoir simulation due to the complex nature of the fractures and their interactions with the surrounding rock matrix. The following points outline the main complications encountered in such type of reservoirs:

- * **Heterogeneity and Uncertainty:** The inherent variability in fracture properties, such as type, size, orientation, and connectivity, complicates the characterization of the fracture-matrix system, leading to uncertainties in flow predictions.
- * **Flow and Transport Complexity:** Understanding and conceptualizing fluid flow and transport processes in fractured formations is difficult due to the dual flow paths through fractures and the matrix, which can behave differently under varying conditions.
- * **Fracture-Matrix Interaction:** Accurately modeling the interaction between fractures and the matrix is crucial, especially in multiphase flow scenarios (oil, gas, water), where pressure gradients, permeability, and capillary forces play significant roles.
- * **Data Requirements:** Detailed knowledge of fracture and matrix properties is often lacking, making it challenging to obtain the necessary data for effective modeling at field scales.
- * **Computational Intensity:** Incorporating realistic three-dimensional fracture geometries into mathematical models requires substantial computational resources, which can limit the feasibility of comprehensive simulations.

These factors collectively pose significant obstacles to successful simulate flow behavior in a fractured media, necessitating advanced modeling techniques, more information about the fractures and a solid background in programming

Chapter 3

Deep Learning

3.1 Introduction

We live in a world that is drowning in data, the needs to optimize and create solutions without human intervention has been increasingly popular over the years. There have always been dreams of creating systems capable of reproducing human intelligence, this has been attempted with AI. John McCarthy defined AI in 1955 as “The science and engineering of making intelligent machines”. AI tries to replicate human problem-solving abilities. They do this by taking data, processing it, and learning from their past in order to streamline and improve in the future. A normal computer program would need human interference in order to fix bugs and improve processes.

ML is a subfield of AI that enables a system to learn from data rather than through explicit programming, to achieve this, ML uses a variety of algorithms to predict outcomes. The ML model can improve its performance through several techniques such as data optimization, improvement in the model’s hyper-parameter settings and optimization in the way the model is trained.

DL is a more complex branch of ML that utilizes neural networks to learn from data in an iterative manner. The corresponding perspective on deep learning models is that they are engineered systems inspired by the biological brains so computers can be trained to deal with abstractions and complex problems. The neural perspective on deep learning is motivated by two main ideas. One idea is that the brain provides a proof by example that intelligent behavior is possible, and a conceptually straightforward path to building intelligence is to reverse engineer the computational principles behind the brain and duplicate its functionality. Another perspective is that it would be deeply interesting to understand the brain and the principles that underlie human intelligence, so machine learning models that shed light on these basic scientific questions are useful apart from their ability to solve engineering applications. (Ian Goodfellow & Courville, 2016)

In the context of petroleum engineering, ML and DL models have become very popular due to the multiple applications which they can be applied and therefore increase efficiency,

reduce costs, optimize traditional practices and improve decision-making. Today, the oil and gas industry faces numerous challenges for a project to be profitable and the growing need for sustainable practices have led to the creation of new technologies that can enhance exploration, drilling, production, and reservoir management processes.

Furthermore, reservoir simulation has benefited from ML and DL models and year after year more articles are published on the subject. Traditional reservoir simulation numerical models can be complex and time-consuming which often requires significant computational resources, nevertheless, AI-driven approaches have shown good performance and it has been found that similar results can be obtained in less time.

The aim of this chapter is to present a brief explanation of how some ML and DL models are built and how they work to understand why or why not the models can be adapted to simulate complex reservoir conditions.

3.2 Neural Networks

NNs is a ML model inspired by the structure and function of the human brain. A neural network can have several connected layers, each neuron within a layer performs a simple computation, but when combined into larger networks, they can learn complex representations.

Basic structure of a NN:

- * Input Layer: The first layer that receives the raw input data.
- * Hidden Layer: This layer process the data and extract features, the more hidden layers the NN has, more complex patterns can learn.
- * Output Layer: The final layer produces the output, which can be a classification or regression value.

Each neuron takes a set of inputs, processes the information by weights which determine the influence of one neuron on another (values the network adjusts during training) and sums them with a bias. The sum passes through an activation function to determine its output.

The neuron formula is presented by:

$$y = f \left(\sum_{i=1}^n w_i x_i + b \right) \quad (3.1)$$

Where:

- x_i :inputs of the neuron.
- w_i :weights.
- b :bias.
- f :activation function.
- y :output of the neuron.

ANN training process:

Initialization: The weights and biases of the neural network are set randomly.

Forward Propagation: The data passes through the layers, and the calculation of each neuron is performed generating a prediction.

Loss Function: Once the network generates the prediction, it is compared to the expected value.

Backpropagation: The calculated error is "backpropagated" through the network, starting from the output layer to the input layers.

Adjustment of the weights: The weights are updated to reduce the error. This adjustment is done using an optimization algorithm.

Iteration: The process of forward propagation, error calculation, back propagation, and weight adjustment is repeated multiple times until the network learns to make accurate predictions.

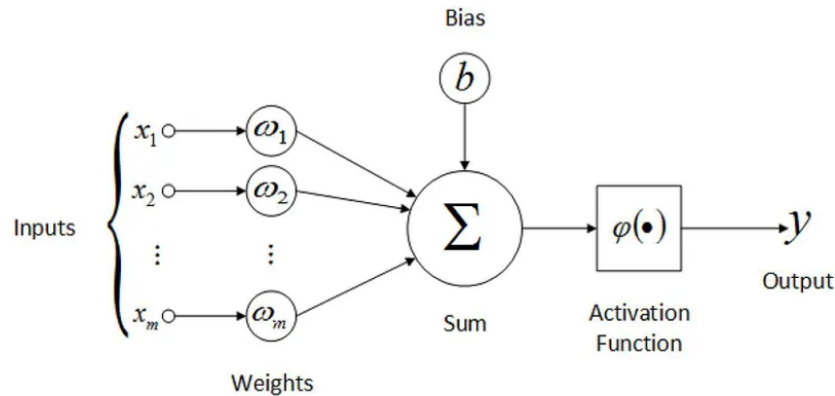


Figure 3.1: "Schematic representation of a neuron" (Roos, 2019)

3.2.1 Deep Feedforward Neural Network

Deep Feedforward Networks (DFNs) also often called feed-forward neural networks, or multilayer perceptrons (MLPs) are a type of neural networks whose goal is to approximate a function f^* . A feed-forward network defines a parametric function $\mathbf{y} = \mathbf{f}(x; \boldsymbol{\theta})$ where x represents our input features and $\boldsymbol{\theta}$ represent the group of adjustable parameters (weight and bias).

The information used in these models is considered **feed-forward** because it goes through the function being evaluated from x , through f , to obtain a predicted output y . Subsequently, the **back-propagation** algorithm (Rumelhart et al., 1986a), often simply called **backprop**, allows the information from the loss function to then flow backwards through the network, in order to compute the gradient with respect to the model parameters (Ian Goodfellow & Courville, 2016).

It is also important to consider that in these models there are no feedback connections in which outputs of the model are fed back into itself. There is a model that can have feed-back connection, this model is called Recurrent Neural Network (RNN).

The architecture of the feed-forward networks makes information goes only in one direction, from the input layer through the hidden layer to the output layer, they are called networks because normally they are composed by different functions. For example, here we have three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ connected in a chain, to form $f(\mathbf{x}) = f^{(1)}(f^{(2)}(f^{(3)}(\mathbf{x})))$. In this case this function has a **depth** of 3, the output layer is the outer-most function which is $f^{(1)}$ in this case. $f^{(3)}$ is the first layer of the network, $f^{(2)}$ is the second layer, and so on. Some of those layers are called hidden layers because the learning algorithm itself figures out how to use these layers in order to achieve the best approximation of f^* .

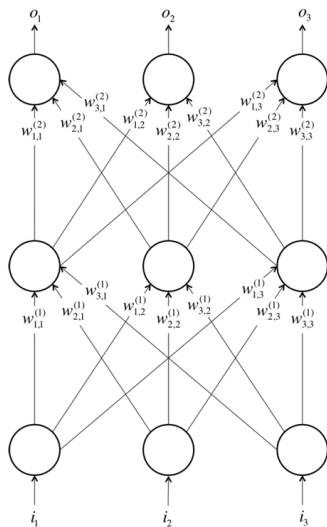


Figure 3.2: example of a feed-forward neural network with three layers (input, one hidden, and output) and three neurons per layer (Buduma, 2017)

3.2.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) operate in a similar way as humans process information. As you read this thesis you don't forget every word as you continue reading, instead, you understand each word based on your understanding of previous words, in other words, you retain the context to understand the full meaning.

As we discussed with the DFNN, NNs can't retain information, to achieve this, more complex models were needed, so in the 1980s RNNs were introduced.

The memory process of a RNN begins with incorporating recurrent connections that allows our model to "remember" information from previous steps and enabling the network to model temporal dependencies

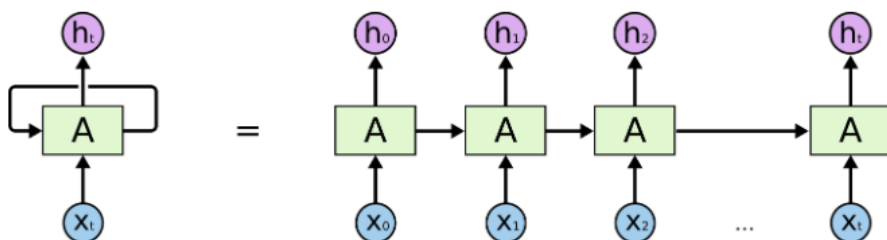


Figure 3.3: An unrolled recurrent neural network (Olah, 2015)

Figure 3.3 represent an unrolled RNN where A is our model which has an input x_t and outputs a value h_t . If we unroll the loop we can visualize that the nature of RNNs are related to sequences and lists.

In an RNN, each time step introduces new information to the model, it also carries forward a summary of what it has previously seen, and it updates its "memory" through the hidden state, in simple terms, it "remembers" earlier inputs while processing new ones. During training, RNN learns temporal patterns in the data by adjusting a set of weights and updating the hidden state at each step to minimize prediction error, enabling the network how much past information should influence future predictions. (Olah, 2015)

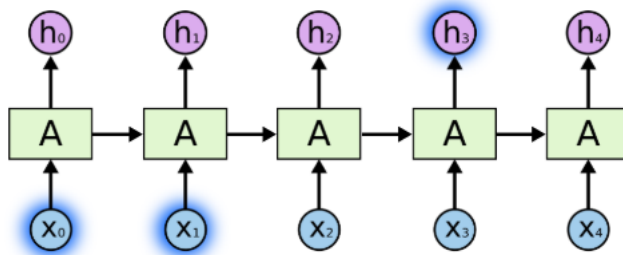


Figure 3.4: An unrolled RNN handling short-term dependencies (Olah, 2015)

3.2.3 Long Short Term Memory

The basic problem of learning long-term dependencies in recurrent networks is that gradients propagated over many stages tend to either vanish or explode, to address this issue Sepp Hochreiter and Jürgen Schmidhuber introduced the Long Short-Term Memory (LSTM) architecture.

The computational unit of an LSTM neural network is called a memory cell, memory layer, memory block or simply cell. Which contains three gates: forget gate, update gate and the output gate. In addition, each cell has an input data x_t , and data from the previous cell h_{t-1} . (Olah, 2015).

The key of the LSTM is the cell state which we can visualize it with the horizontal line in Figure 3.5. This cell commonly represented as C_t helps information pass through it adding or losing data.

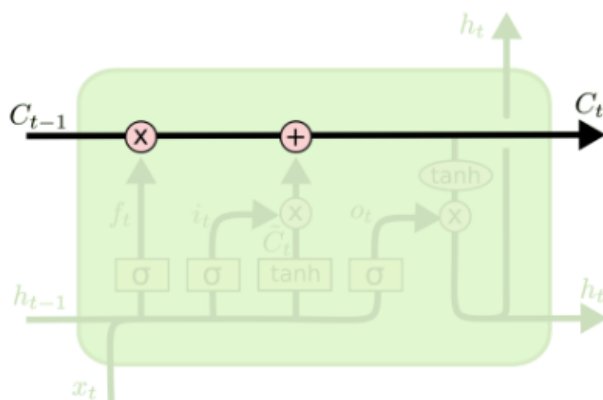


Figure 3.5: LSTM state cell (Olah, 2015)

Within each cell, the LSTM has structures called gates. The first gate is often called "forget gate layer" which is a sigmoid layer that filters unnecessary information.

Essentially, the forget gate layer is activated by looking at the previous hidden state h_{t-1} and the input data x_t , and output a number between 0 and 1 for each number in the cell state C_{t-1} . If the output is closer to 1 then it is labeled as relevant information and therefore, will remain in the cell state, on the contrary, if the output has a value of 0, the information is not relevant and will be disregarded.

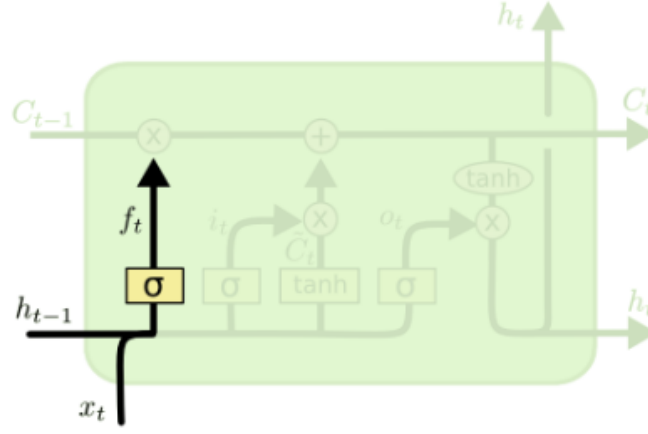


Figure 3.6: LSTM forget gate (Olah, 2015)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.2)$$

Next, we have the update gate cell that updates the cell state. This is divided into two parts. We have a sigmoid layer called "input gate layer" which decides the values we are going to update (i_t), the values are updated and then a tanh function creates a new vector of values ($\tilde{C}_t$).

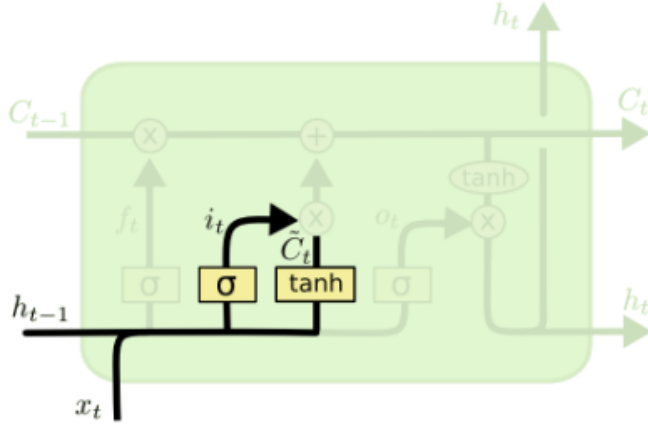


Figure 3.7: LSTM update gate (Olah, 2015)

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.3)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3.4)$$

A set of new value candidates is creating by updating the previous cell state C_{t-1} . It is multiplied by the forget gate layer f_t and added the update gate cell represented as $i_t * C_t$. This is how we get the new cell state C_t .

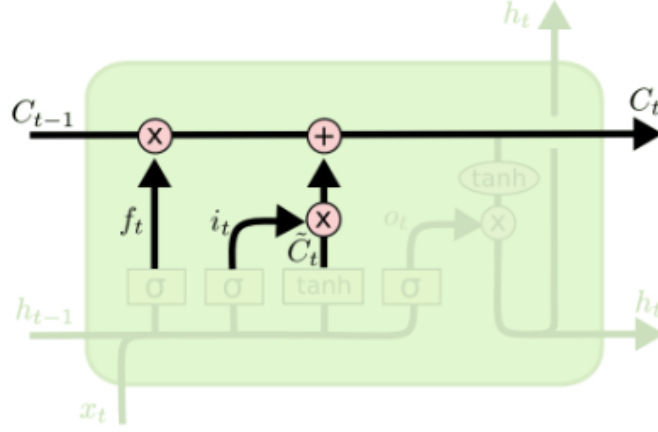


Figure 3.8: Union of information to be updated in the cell state (Olah, 2015)

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (3.5)$$

Finally we have the output gate o_t which decides our output for the time t . This gate acts as a filter thanks to the sigmoid function that creates a vector with values between 0 and 1. Then, the updated cell state C_t is transformed using the hyperbolic tangent function and it is multiplied by the output of the sigmoid gate.

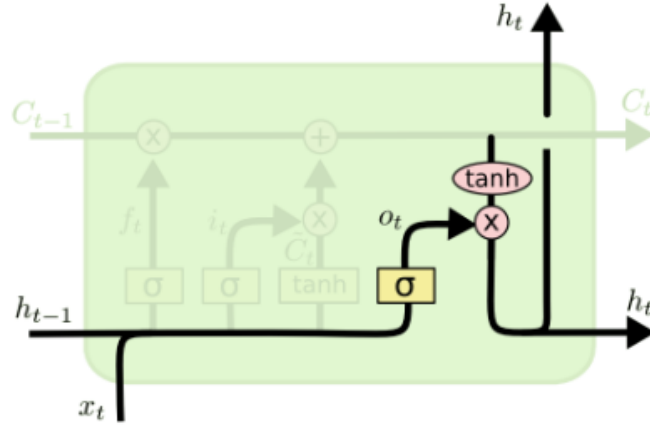


Figure 3.9: LSTM output gate (This gate filters the information to be passed to the next layer) (Olah, 2015)

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (3.6)$$

$$h_t = o_t * \tanh(C_t) \quad (3.7)$$

3.3 Optimization

Optimization is a process involved in the training part of a model. In order for the model to learn to perform a function in the most effective way, we compare the accuracy of a model in every iteration by tuning the model's configuration or hyperparameters. Optimization is measured through a loss or cost function, which is typically a way of defining the difference between the predicted and actual value of data, there are a wide range of optimization algorithms known as optimizers which employ distinct strategies to facilitate the learning process by iteratively refining the weights and biases based on the feedback received from the data.

Gradient Descent:

Gradient Descent (GD) is an optimization algorithm that iteratively adjusts a function's parameters to minimize a cost function, the parameters refer to both the coefficients in linear regression as well as the weights and biases of the neural networks. GD starts with initial values of weight and biases at a starting point, calculates the gradient that measures the change in all weights with regards to the change in error, and starts to move in the direction of the deepest descent. With each iteration, GD tries to reduce the value of the cost and eventually it will converge towards a minimum cost.

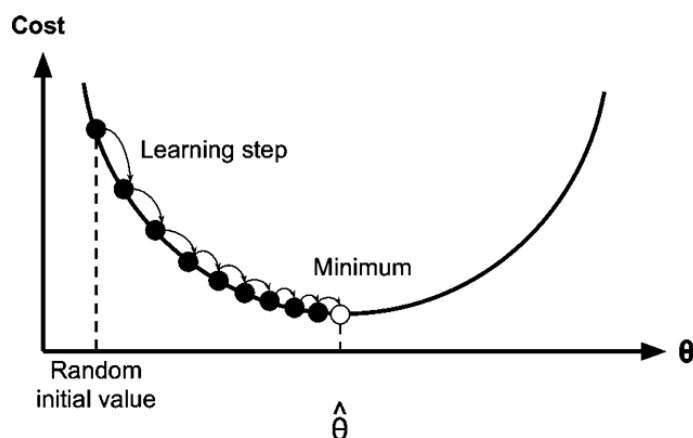


Figure 3.10: Gradient Descent algorithm (Ghassemzadeh, 2020)

Stochastic Gradient Descent:

Stochastic Gradient Descent (SGD) updates model parameters incrementally using small batches of data rather than the full dataset. This makes it computationally efficient, it also converges faster especially when dealing with non-convex loss functions. SGD introduces randomness, helping the algorithm escape local minima.

Minibatch Gradient Descent:

Mini-batch gradient descent, at every iteration, we split the dataset into small batches to compute errors and update model coefficients. This subset is called a minibatch which is also another hyperparameter.

Adaptive Moment Estimation:

Adaptive Moment Estimation (Adam) is best seen as a variant on the combination of momentum optimization and Root Mean Square Propagation (RMsp) with a very few important distinctions. Adam adapts learning rates for each parameter individually using two exponentially decaying averages: the first moment (mean) and the second moment (uncentered variance) of the gradients.

3.4 Regularization

A central problem in DL is how to make an algorithm that will perform well not just on the training data, but also on new inputs. Many strategies used are explicitly designed to reduce the test error, possibly at the expense of increased training error, these strategies are known collectively as regularization (Ian Goodfellow & Courville, 2016).

In this chapter, we will describe regularization focusing on regularization strategies for deep models or models that may be used as building blocks to form deep models.

3.4.1 L1 and L2 Regularization

L1 and L2 regularization add penalty terms to the loss function to control the magnitude of the parameters:

- **L1 Regularization (Lasso):** Penalizes the sum of the absolute values of the weights ($|w|$). This encourages simpler models by setting some weights to exactly zero:

$$J(\theta) = Loss + \lambda \sum |w| \quad (3.8)$$

- **L2 Regularization (Ridge):** Penalizes the sum of the squared values of the weights (w^2), promoting more uniform distributions of parameters:

$$J(\theta) = Loss + \lambda \sum w^2 \quad (3.9)$$

3.4.2 Early Stopping

When training large models with sufficient representational capacity to overfit the task, we often observe that training error decreases steadily over time, but validation set error begins to rise again. This means we can obtain a model with better validation set error (and thus, hopefully better test set error) by returning to the parameter setting at the point in time with the lowest validation set error. Every time the error on the validation set improves, we store a copy of the model parameters. When the training algorithm terminates, we return these parameters, rather than the latest parameters. The algorithm terminates when no parameters have improved over the best recorded validation error for some pre-specified number of iterations (Ian Goodfellow & Courville, 2016).

3.5 Hyperparameters

Hyperparameters play a crucial role in the model's performance as hyperparameters control the complexity of the model. Normally hyperparameters have a default value, manual configuration helps to control the learning process, this is achieved with hyperparameter tuning which helps to find the sweet spot of hyperparameter settings and thus, create a more efficient algorithm capable of generalize well with unseen data.

3.5.1 Number of Layers and Neurons

The number of layers and neurons in the network defines the model's capacity to learn patterns:

- **More layers and neurons:** Increases the ability to capture complex patterns but raises the risk of *overfitting*.
- **Fewer layers and neurons:** Reduces the risk of *overfitting* but may not be sufficient for complex patterns.

During the creation of the model's architecture it is important to know that the more layers and neurons a model has, the more complex it is going to be and it will require more hardware to run efficiently.

3.5.2 Learning Rate

The learning rate (η) controls how the model adjusts its parameters during training. Proper configuration ensures convergence and can be dynamically adjusted during training, as discussed in the optimization section.

3.5.3 Batch Size

The batch size determines how many training examples are used to update parameters:

- **Small batch:** Introduces noise in updates, which can help escape local minima.
- **Large batch:** Reduces noise but requires more memory and may hinder escaping local minima.

3.5.4 Number of Epochs

The number of epochs specifies how many times the model processes the entire dataset during training:

- **Too many epochs:** Can lead to *overfitting*.
- **Too few epochs:** Can cause *underfitting*.

Early stopping is useful for automatically determining the optimal number of epochs.

3.5.5 Activation Functions

Activation functions are mathematical operations that determine whether or not a given node should be triggered. Without activation functions, the NN would be equivalent to a linear regression model regardless of the number of layers. They introduce non-linearity into the model, which allows the NN to learn more complex patterns. Through the activation, NNs can adjust their weights during training to bring the model closer to the desired solution.

Sigmoid

Calculated with the formula 3.10. This formula tells us the raw value computed by the neuron before applying the activation function will have a very small or very large number, the output between these extremes is an S shape shown in Figure 3.11

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.10)$$

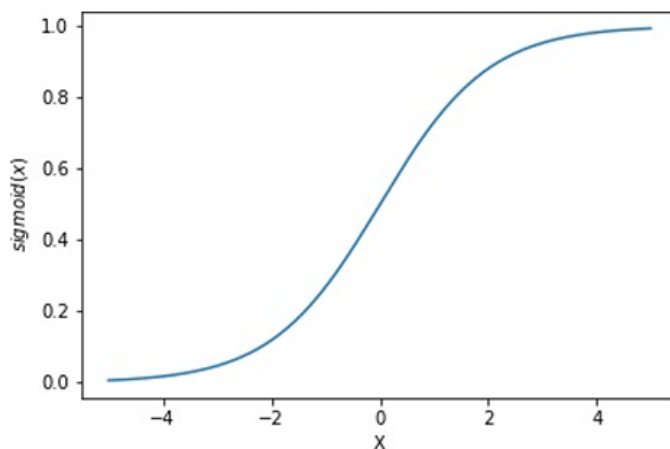


Figure 3.11: Sigmoid activation function (Buduma, 2017)

Rectified Linear Unit

The Rectified Linear Unit (ReLU) function is simply defined to be 0 if the input x is less than 0, if the input x is greater than 0, the output is equal to 1. ReLU has become popular over the years and it is the standard activation function in some cases as a result of having good convergence speed.

$$f(x) = \max(0, x) \quad (3.11)$$

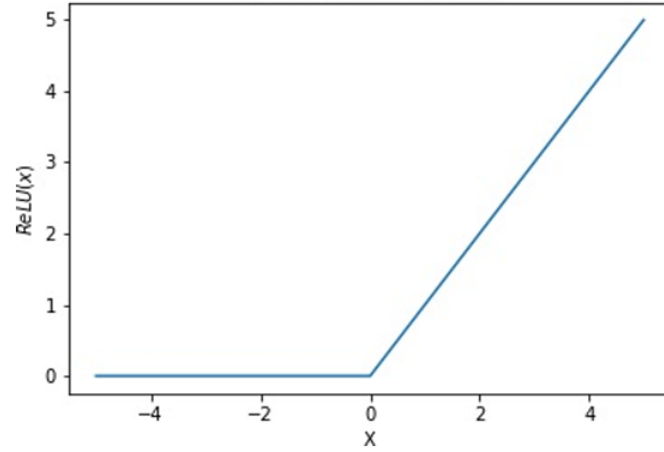


Figure 3.12: ReLU activation function (Buduma, 2017)

Hyperbolic Tangent

Hyperbolic Tangent (Tanh) is similar to the sigmoid function but it has a wider range of values, instead of ranging from 0 to 1, tanh neurons produces outputs from -1 to 1.

$$f(x) = \tanh(x) \quad (3.12)$$

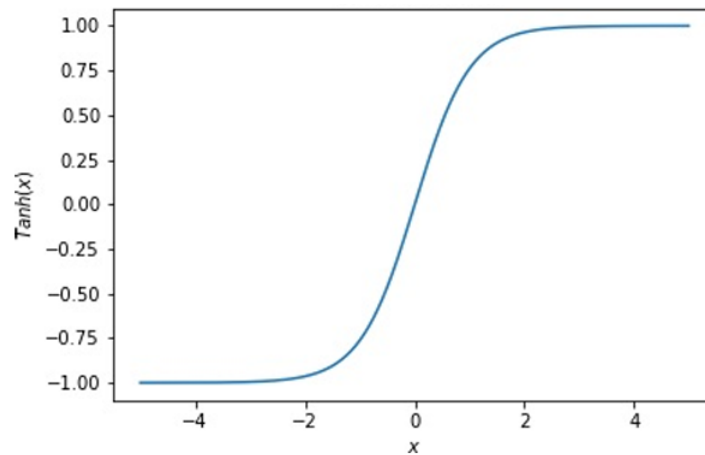


Figure 3.13: Tanh activation function (Buduma, 2017)

Fractured Reservoirs

4.1 Data collection

Naturally Fractured model

In the simulation of unconventional reservoirs, it is necessary to consider the behavior and impact natural fractures have on the depletion and production of this type of reservoirs, because of this, it is common that the formulations that have been developed for the simulation of naturally fractured reservoirs are implemented for the simulation of the effects of natural fractures present in unconventional reservoirs; however, many of these formulations consider a behavior of the matrix-fracture transfer function of PSS type.

That's why, in this case, a simulation of an unconventional model with hydraulic fracturing effects in a horizontal well will be performed, considering a dual porosity and dual permeability model (naturally fractured), logarithmic spacing and local refinement of the simulation grid, better known as “DK-LS-LGR” methodology.

Due to the PSS model generates a delay in the behavior of the pressure loss of the matrix regarding the USS model, for this thesis will be considered the USS model from Streltsova.

3D numerical model, naturally fractured and with hydraulic fracturing effects

Considering the formulation presented by (Torcuk et al. 2013) that represents the fluid flow considering a semi-compressible fluid in a naturally fractured porous medium, where the natural fractures are the continuous medium (where the fluid flows into the wellbore) and the matrix blocks the discontinuous medium (where only the fluid contributes).

$$\frac{\partial}{\partial x}(\frac{k_{feff}}{\mu} \frac{P_f}{\partial x}) + \frac{\partial}{\partial y}(\frac{k_{feff}}{\mu} \frac{P_f}{\partial y}) + \frac{\partial}{\partial z}(\frac{k_{feff}}{\mu} (\frac{P_f}{\partial z} - \gamma \frac{\partial D}{\partial z})) + \hat{q} + \tau = (\emptyset C_t)_f \frac{P_f}{\partial t} \quad (4.1)$$

$$-\tau = (\emptyset C_t)_m \frac{\partial P_m}{\partial t} \quad (4.2)$$

With the previous development of single-phase flow equations for a reservoir with double porosity and double permeability, we can now carry out the three-dimensional simulation of a naturally fractured reservoir, without gravitational effects and considering a matrix-fracture transfer function of USS type, with hydraulic fracturing effects in 'y' direction and with a horizontal well in 'x' direction, as well as considering different geometries for the representation of matrix blocks (spheres, slabs and cubes).

Model application

The synthetic simulation model is based on the model to simulate fluid flow in fractured reservoirs, embedding Streltsova's unsteady-state matrix transfer into a dual-porosity framework (Teja-Juárez et al., 2025). The semi-analytical USS formulation proposed, in a tridimensional simulation grid of 35 X 35 X 15 cells (18375 cells), 1000 time steps, a slabs shape factor and 1000 additions corresponding to the fracture matrix transfer function, Likewise, a homogeneous distribution (isotropic model) of the petrophysical properties of the matrix and the natural fracture will be considered, this is, considering that throughout the reservoir domain there will be a single value of porosity and permeability of the matrix, and likewise, for the natural fracture, excluding the cells that will contain the hydraulic fracture properties through a double permeability (and double porosity) model, logarithmic spacing, and local refinement of the simulation grid, better known as DK-LS-LGR (Cipolla et al. 2009; Rubin 2010; Jimenez and Aguilera 2019).

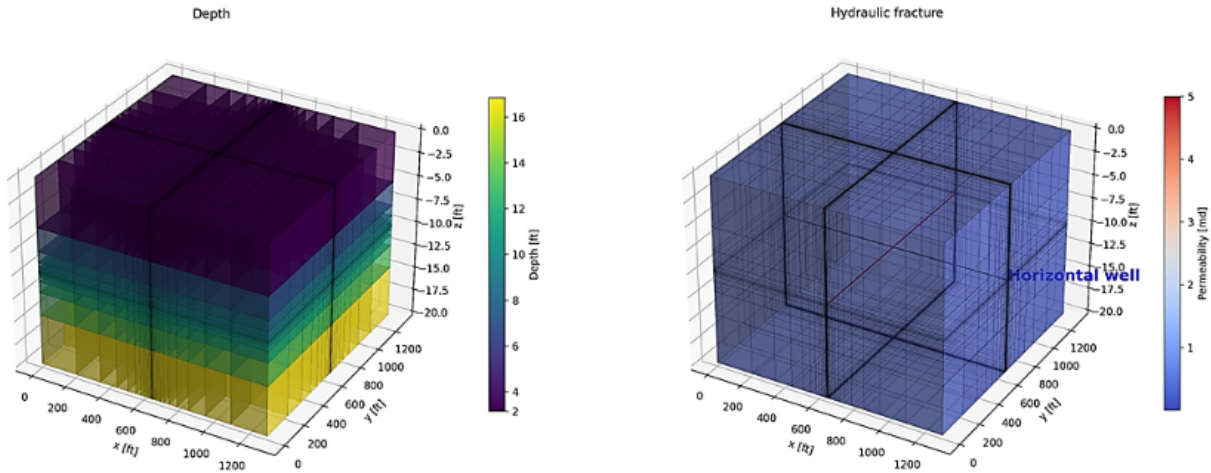


Figure 4.1: Three-dimensional simulation homogeneous model

Approximately 200 simulation runs were performed, varying petrophysical parameters such as permeability, porosity, and fracture width, generating a database of about 2 terabytes. This was achieved with the support of UNAM's supercomputer *Miztli*. Since the amount of data was considerable, we selected the flowing bottomhole pressure as the target variable in order to reduce data handling workload and focus on a parameter of great relevance in petroleum engineering.

PSS and USS parameters		
Parameters	Values	Units
Initial pressure, P_{ini}	5000	[Psi]
Bottomhole flowing pressure constrain, $P_{wf min}$	500	[Psi]
Gas production rate, q_g	500	[ft3/day]
Fracture length in x direction, L_x	1	[ft]
Fracture length in y direction, L_y	1	[ft]
Fracture length in z direction, L_z	1	[ft]
Radius of the matrix block, r_m	1	[ft]
Reservoir length in x direction	1300	[ft]
Reservoir length in y direction	1300	[ft]
Reservoir length in z direction	20	[ft]
Matrix porosity, ϕ_m	Random value	[Fraction]
Hydraulic fracture porosity, ϕ_{hf}	0.001	[Fraction]
Natural fracture width, w_f	Random value	[μ m]
Hydraulic fracture width, w_{hf}	0.001	[ft]
Total matrix compressibility, C_{zm}	1.00E-04	[1/Psi]
Total fracture compressibility, C_{zf}	5.00E-04	[1/Psi]
Total hydraulic fracture compressibility, C_{thf}	5.00E-04	[1/Psi]
Matrix permeability, k_m	Random value	[md]
Hydraulic fracture permeability, k_{hf}	10000	[md]
Viscosity, μ	0.02	[cp]
Well radius, r_w	0.22	[ft]
Horizontal well length in x direction	1300	[ft]
Wellbore storage	0	[ft3/Psi]
Number of nodes in x direction	35	[-]
Number of nodes in y direction	35	[-]
Number of nodes in z direction	15	[-]
Well node in x direction	18	[-]
Well node in y direction	18	[-]
Well node in z direction	8	[-]
Simulation time, t	3650	[Days]
Initial time differential, Δt_0	1.00E-08	[Days]
Time steps	1000	[-]
Number of sums of the semi-analytical model (this work)	1000	[-]

Figure 4.2: Reservoir parameters used in the homogeneous simulation model

4.2 Exploratory Data Analysis

The main purpose of Exploratory Data Analysis (EDA) is to help look and understand at data before making any assumptions. It can help identify some errors, as well as better understand of the data sets by summarizing and plotting patterns, distribution of the data, detect outliers and find relations among the variables. In general it is a good practice to conduct an EDA if we are not familiar with the problem that we can to replicate because without an EDA it would be difficult to determine which fields are related to the target column, the dispersion of the data and all of the info we get which was mentioned before.

1.- Retrieving data

This step consists of loading the dataset and display the first 5 rows of the dataset. It helps us to visualize a brief part of our dataset so we can know the number of columns we have, it helps visualize if we have an anomaly or null values.

```
df = pd.read_excel('Data_7.xlsx')
df.head()
```

	t Dias	Pwf num Psi	Delt_P_num Psi	Og Scf/Días	Derv Delt P num Psi	rw en ft	Long_x_pz (long pozo) en ft	Long_x (longitud del yac en x) en ft	Long_y (longitud del yac en y) en ft	Pf_ini (presión de la fract nat inicial) en Psi	Pm_ini (presión de la matriz inicial) en Psi	Ppozo_ini (presión inicial del pozo) en Psi	Visc del gas en cP	Khf (Permeabilidad real de la fract hidrlc) en mD	Por. (Porosid de fractu hidrlc) Porcent
0	1.054689e-08	4998.647739	1.352261	-500.0	25.584011	0.2	1300	1300	1300	5000	5000	5000	0.02	10000	0.0
1	1.083146e-08	4997.960753	2.039247	-500.0	26.031338	0.2	1300	1300	1300	5000	5000	5000	0.02	10000	0.0
2	1.112370e-08	4997.281655	2.738335	-500.0	26.493225	0.2	1300	1300	1300	5000	5000	5000	0.02	10000	0.0
3	1.142382e-08	4996.550086	3.449914	-500.0	26.969982	0.2	1300	1300	1300	5000	5000	5000	0.02	10000	0.0
4	1.173205e-08	4995.825613	4.174387	-500.0	27.461895	0.2	1300	1300	1300	5000	5000	5000	0.02	10000	0.0

Figure 4.3: First rows of the dataset

2.- Inspection of the data

Here we are going to check that our data set has been loaded correctly, we will also check for null values, dispersion and correlation of the data. These steps are going to help us to comprehend and to clean our dataset.

```
df.columns
```

```
Index(['t Días', 'Pwf num Psi', 'Delt_P_num Psi', 'Og Scf/Días',  
      'Derv Delt P num Psi', 'rw en ft', 'Long_x_pz (long pozo) en ft',  
      'Long_x (longitud del yac en x) en ft',  
      'Long_y (longitud del yac en y) en ft',  
      'Pf_ini (presión de la fract nat inicial) en Psi',  
      'Pm_ini (presión de la matriz inicial) en Psi',  
      'Ppozo_ini (presión inicial del pozo) en Psi', 'Visc del gas en cP',  
      'Khf (Permeabilidad real de la fract hidrlc) en mD',  
      'Por_hf (Porosidad de la fractura hidrlc) en Porcentaje',  
      'Whf_real (espesor real de la fractura hidráulica) en ft',  
      'Por_f_abslt (de la fract nat todo el yac) en Fraccion',  
      'Por_m_abslt (de la matriz todo el yac) en Fraccion',  
      'Porf_hf_abslt (de la fract hidrlc todo el yac) en Fraccion'],  
      dtype='object')
```

Figure 4.4: Columns of the dataset

```
df.dtypes
```

t Días	float64
Pwf num Psi	float64
Delt_P num Psi	float64
Qg Scf/Días	float64
Derv Delt P num Psi	float64
rw en ft	float64
Long_x_pz (long pozo) en ft	int64
Long_x (longitud del yac en x) en ft	int64
Long_y (longitud del yac en y) en ft	int64
Pf_ini (presión de la fract nat inicial) en Psi	int64
Pm_ini (presión de la matriz inicial) en Psi	int64
Ppozo_ini (presión inicial del pozo) en Psi	int64
Visc del gas en cP	float64
Khf (Permeabilidad real de la fract hidrlc) en mD	int64
Por_hf (Porosidad de la fractura hidrlc) en Porcentaje	float64
Whf_real (espesor real de la fractura hidráulica) en ft	float64
Por_f_abslt (de la fract nat todo el yac) en Fraccion	float64
Por_m_abslt (de la matriz todo el yac) en Fraccion	float64
Porf_hf_abslt (de la fract hidrlc todo el yac) en Fraccion	float64
dtype: object	

Figure 4.5: Checking data types

```
df.isna().sum().sort_values()
```

t Días	0
Por_f_abslt (de la fract nat todo el yac) en Fraccion	0
Whf_real (espesor real de la fractura hidráulica) en ft	0
Por_hf (Porosidad de la fractura hidrlc) en Porcentaje	0
Khf (Permeabilidad real de la fract hidrlc) en mD	0
Visc del gas en cP	0
Ppozo_ini (presión inicial del pozo) en Psi	0
Pm_ini (presión de la matriz inicial) en Psi	0
Por_m_abslt (de la matriz todo el yac) en Fraccion	0
Pf_ini (presión de la fract nat inicial) en Psi	0
Long_x (longitud del yac en x) en ft	0
Long_x_pz (long pozo) en ft	0
rw en ft	0
Derv Delt P num Psi	0
Qg Scf/Días	0
Delt_P num Psi	0
Pwf num Psi	0
Long_y (longitud del yac en y) en ft	0
Porf_hf_abslt (de la fract hidrlc todo el yac) en Fraccion	0
dtype: int64	

Figure 4.6: Checking for missing values

```
df.describe()
```

	t Días	Pwf num Psi	Delt_P_num Psi	Qg Scf/Días	Derv Delt P num Psi	rw en ft	Long_x_pz (long pozo) en ft	Long_x (longitud del yac en x) en ft	Long_y (longitud del yac en y) en ft	Pf_ini (presión de la fract nat inicial) en Psi	Pn (presión de la matriz inicial) en Psi
count	5.012800e+04	50128.000000	50128.000000	50128.000000	50128.000000	50128.000000	50128.000000	50128.000000	50128.000000	50128.000000	50128.000000
mean	4.651422e+01	3223.587692	1157.676272	-528.604750	165.590470	0.239963	1073.795085	1073.795085	1073.795085	4381.244015	4381.244015
std	2.299910e+02	1662.330355	1001.651652	326.575871	326.136973	0.024406	563.311948	563.311948	563.311948	1511.495280	1511.495280
min	1.054689e-08	500.000000	0.054697	-1200.000011	-2.514631	0.200000	300.000000	300.000000	300.000000	2000.000000	2000.000000
25%	3.315735e-06	1848.314517	276.555344	-800.000000	40.988360	0.220000	800.000000	800.000000	800.000000	3000.000000	3000.000000
50%	1.513238e-03	2841.314639	976.254563	-500.000000	80.563741	0.240000	900.000000	900.000000	900.000000	4000.000000	4000.000000
75%	9.255924e-01	4490.024448	1814.115153	-200.000000	156.333132	0.260000	1500.000000	1500.000000	1500.000000	6000.000000	6000.000000
max	3.650000e+03	6999.945303	6495.713496	0.019030	4752.144671	0.280000	2100.000000	2100.000000	2100.000000	7000.000000	7000.000000

Figure 4.7: Descriptive statistics of the dataset

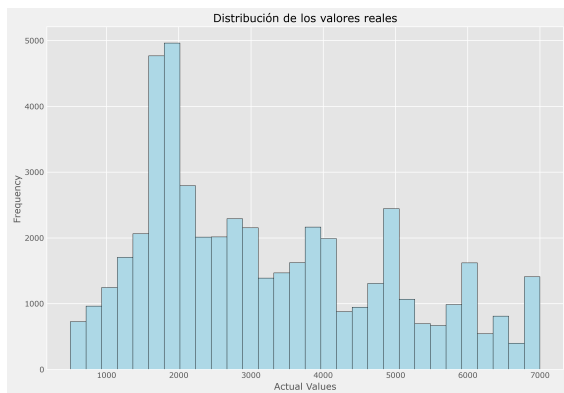


Figure 4.8: Distribution of the dataset

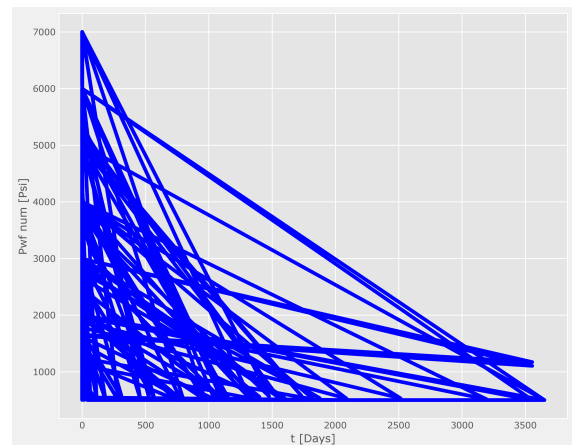


Figure 4.9: Relation between the time period and Pwf num [Psi]

4.3 Data Normalization

After examining the data, it is helpful to transform it to help our model train more efficiently and thus, to be more precise. Normalization is a technique which rescales the numerical features of a dataset to a standard range.

Advantages of Data Normalization

- Ensures that all features contribute equally during training resulting in faster model convergence.
- Standardized features improves consistency leading to more reliable results.
- Facilitates the use of techniques such as L1 and L2 regularization.

Min-Max Normalization

This normalization technique involves transforming the data in a linear way. The minimum and maximum values of the data are obtained and the values are scaled to a fixed range which usually is between 0 to 1. The formula for calculating the Min-Max Scaling is:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (4.3)$$

Z-score Normalization

Transforms the features so they can have the properties of a normal distribution with a mean of 0 and a standard deviation of 1:

$$Z = \frac{X - \mu}{\sigma} \quad (4.4)$$

where:

μ is the mean of the feature.

σ is the standard deviation.

4.4 Evaluation of possible models

Cross-validation is a technique that validates that a model can make accurate predictions with unseen data after training. This is accomplished by testing our model's behavior by making predictions on subsets of data it has not encountered in training. It assists by reducing overfitting and obtaining better decisions about the model structure and how it may just not perform greatly on training sets but can also function effectively in environments which it has never seen before. Here is a flowchart of typical cross validation workflow in model training. The best parameters can be determined by grid search techniques.

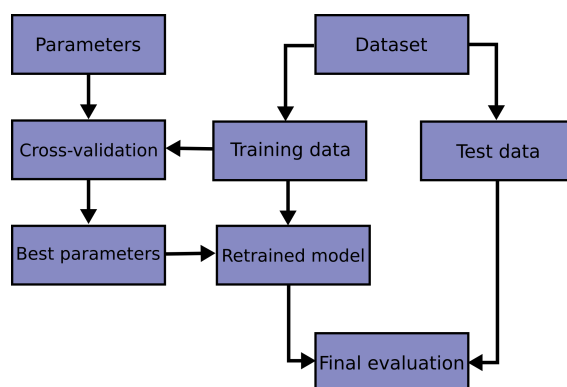


Figure 4.10: Grid Search workflow (Scikit-Learn, n.d.)

TimeSeriesSplit is a variation of k-fold which returns first k folds as train set and the $(k + 1)$ th fold as test set. Note that unlike standard cross-validation methods, successive training sets are supersets of those that come before them. Also, it adds all surplus data to the first training partition, which is always used to train the model.

This class can be used to cross-validate time series data samples that are observed at fixed time intervals.

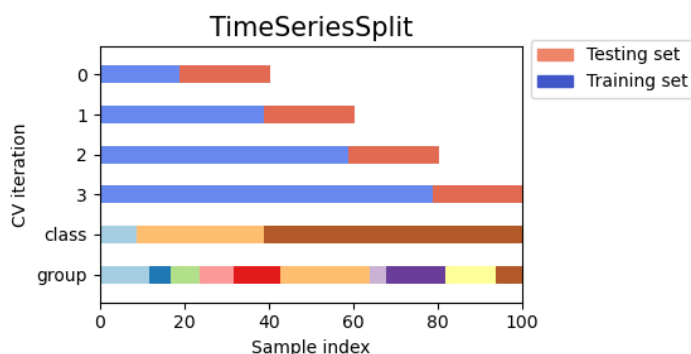


Figure 4.11: Cross-validation behavior (Scikit-Learn, n.d.)

4.5 Model selection and training

After finishing cross validation, we get to store the best parameters with the lowest loss for our NN. The next step is to train once again our model with the entire dataset with the parameters obtained from cross validation.

Our NN architecture was selected based on the results of the cross validation technique. Batch size, l2 regularizer, learning rate, neurons per layer and number of layers were the parameters that were used to find the best configuration.

Params	Fold	Loss
{'batch_size': 32, 'l2_regularizer': 0.01, 'learning_rate': 0.001, 'neurons_per_layer': 64, 'num_layers': 5}	1	0.0275
{'batch_size': 32, 'l2_regularizer': 0.01, 'learning_rate': 0.001, 'neurons_per_layer': 64, 'num_layers': 5}	2	0.0933
{'batch_size': 32, 'l2_regularizer': 0.01, 'learning_rate': 0.001, 'neurons_per_layer': 64, 'num_layers': 5}	3	0.2005
{'batch_size': 32, 'l2_regularizer': 0.01, 'learning_rate': 0.001, 'neurons_per_layer': 64, 'num_layers': 5}	4	0.0389
{'batch_size': 32, 'l2_regularizer': 0.01, 'learning_rate': 0.001, 'neurons_per_layer': 64, 'num_layers': 5}	5	0.0554

Table 4.1: Cross-validation results for different hyperparameter settings

In the picture above we can see the best configuration for our model, the full architecture is presented in the chart below.

NN architecture	
Layers	5
Nodes per layer	64
Optimizer	Adam
L2 Regularization	0.01
Activation function	ReLU
Epochs	1000

Table 4.2: Neural network architecture configuration

4.6 Model evaluation

The performance of a regression model must be reported as an error in those predictions. Regression models predict continuous numerical values so different metrics are used to assess their performance. These metrics focus on how close the predicted values are to the actual values and the model's ability to explain the variance in the target variable.

Mean absolute error

Mean Absolute Error (MAE) function computes mean absolute error, a risk metric corresponding to the expected value of the absolute error loss or l_1 -norm loss.

If $\hat{y}_i$ is the predicted value of the i -th sample, and y_i is the corresponding true value, then the MAE estimated over $n_{samples}$ is defined as:

$$MAE = \frac{1}{n_{samples}} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (4.5)$$

Mean squared error

Mean Square Error (MSE) computes a risk metric corresponding to the expected value of the squared (quadratic) error or loss.

If $\hat{y}_i$ is the predicted value of the i -th sample, and y_i is the corresponding true value, then the MSE estimated over $n_{samples}$ is defined as:

$$MSE = \frac{1}{n_{samples}} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (4.6)$$

Mean absolute percentage error

Mean Absolute Percentage Error (MAPE) also known as Mean Absolute Percentage Deviation (MAPD) is an evaluation metric for regression problems. The idea of this metric is to be sensitive to relative errors. It is for example not changed by a global scaling of the target variable.

If $\hat{y}_i$ is the predicted value of the i -th sample, and y_i is the corresponding true value, then the MAPE estimated over $n_{samples}$ is defined as:

$$MAPE = \frac{1}{n_{samples}} \sum_{i=1}^n \left| \frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)} \right| \quad (4.7)$$

Coefficient of determination

Coefficient of determination or R^2 score represents the proportion of variance (y) that has been explained by the independent variables in the model. It provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance.

As such variance is dataset dependent, R^2 may not be meaningfully comparable across different datasets. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected (average) value of y , disregarding the input features, would get an R^2 score of 0.0.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (4.8)$$

Validation and Application of Predictive Models

After training our model, it is very important to evaluate it. For this project, model evaluation techniques in the previous chapter were used.

In order to see if our model presents overfitting or underfitting, evaluation was performed with the train and test set, it was also compared the real values with respect to the predictions, this was carried out with representative graphs.

If the performance between the train and the test set is pretty similar, it will indicate that the model does not present overfitting.

It is essential to visualize the performance of the model because it allows us to detect some problems such as overfitting and underfitting, it can show us the behavior of the model, outliers can be detected and we can compare the actual values with the predictions.

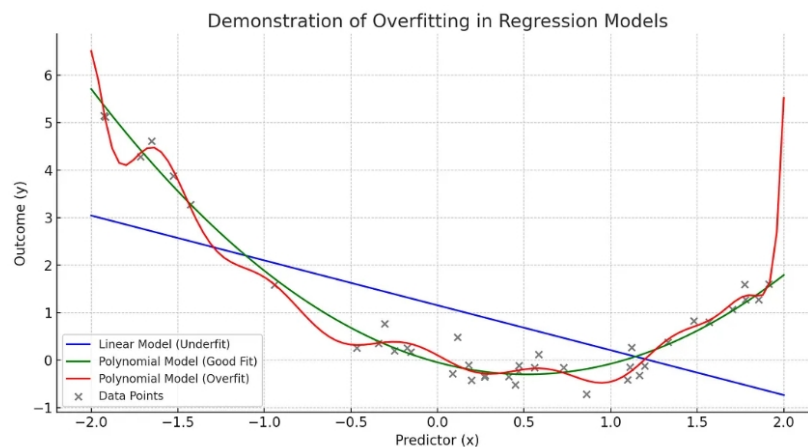


Figure 5.1: Overfitting in Regression models (Sahin, 2025)

5.1 Comparative Analysis and Model Application

DFNN model evaluation

DFNN evaluation		
Metric	Train	Test
MAE	5.4664	5.4483
MSE	225.4684	246.2830
R^2	0.9998	0.9998
MAPE	0.5528%	0.5544%

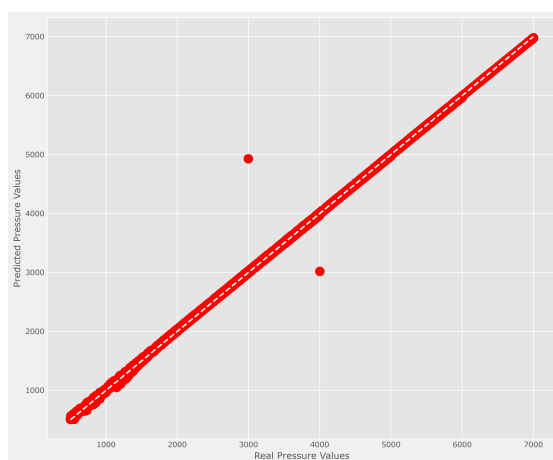


Figure 5.2: DFNN predictions vs real values

In Figure 5.2 a line plot was generated that shows the real or actual values as a continuous line, with the predicted values of the DFNN overlaid as individual points. This visualization allows to visualize how well the model fits to the actual data. The predictions that are close to the line indicate a better model performance, and, on the other hand, predictions further separated from the line can indicate poor model performance or outliers.

As we can see in the graph, the DFNN has remarkable performance making predictions as they are very close to the line, we can see that there are some predictions we can classify as outliers, however, we are going to see that these values do not have a significant impact when making new predictions.

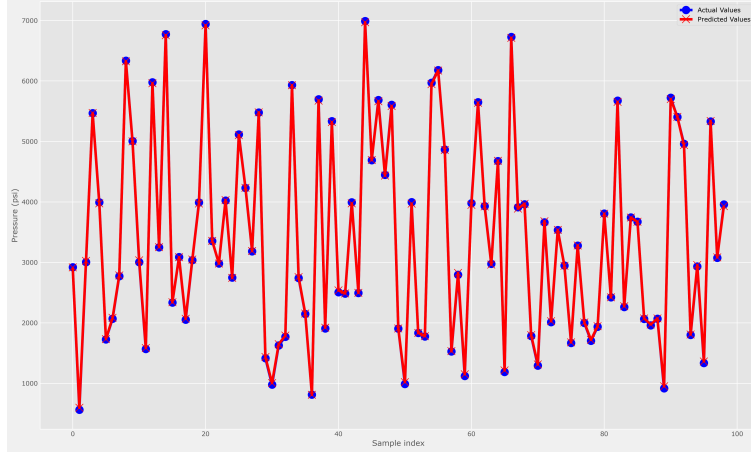


Figure 5.3: DFNN predictions vs real values

Due to the large volume of data, it was plotted a random subset of 100 samples to visualize the relationship between actual and predicted values more clearly. This visualization allows for a closer inspection to different ranges of the number of predictions made, this is helpful to look whether the model is performing well or if there are any outliers. As we can see, our model is performing well, indicating that the model is learning from the data.

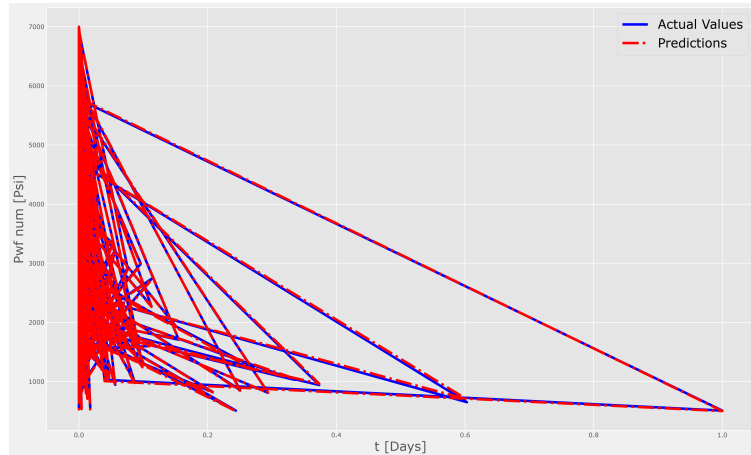


Figure 5.4: DFNN predictions vs real values of target feature

A scatter plot was created to compare the actual values of the target variable against the predictions of the model. In an ideal scenario, all points would lie on the identity line, indicating perfect predictions. As we can see in the visualization, the predictions overlap almost ideally with the actual values of the target variable.

LSTM model evaluation

LSTM evaluation		
Metric	Train	Test
MAE	1.6876	1.6886
MSE	148.9747	173.4291
R^2	0.9999	0.9999
MAPE	0.1116%	0.1159%

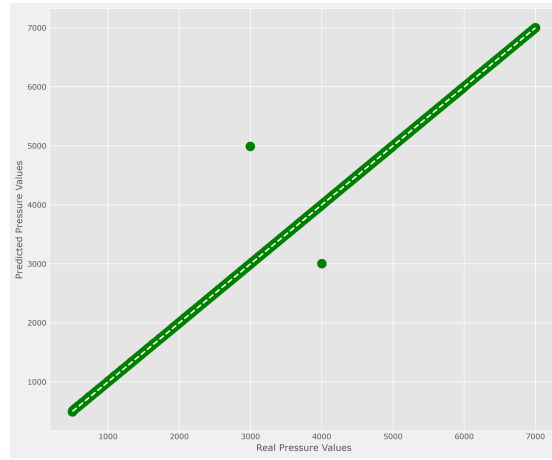


Figure 5.5: LSTM predictions vs real values

Once again, the line plot of the real values against the predictions was presented, and as we can see the predictions have some dispersion to the actual data but no outliers are presented compared to the previous model.

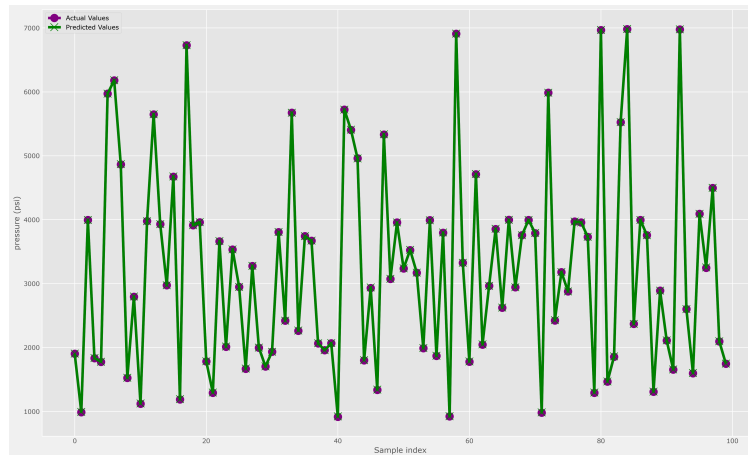


Figure 5.6: LSTM predictions vs real values

As mentioned before, this is a subset of all the predictions made so we can better visualize how the predictions fit the real data. In this graph, we can see that the predictions made by the model fit in an ideal way at least for the predictions plotted.

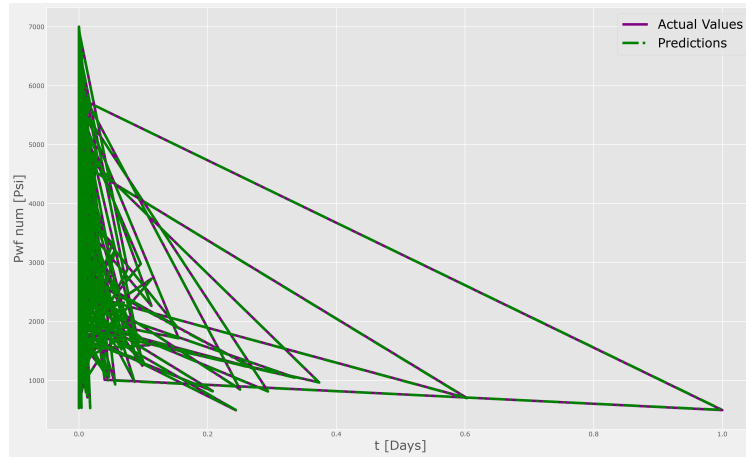


Figure 5.7: LSTM predictions vs real values of target feature

In figure 5.7, we can realize that the predictions are as accurate as expected, increasing accuracy compared to the DFNN.

XGBoost model evaluation

XGBoost evaluation		
Metric	Train	Test
MAE	13.1581	13.2328
MSE	1143.8608	1141.2776
R^2	0.9992	0.9992
MAPE	1.4999%	1.5449%

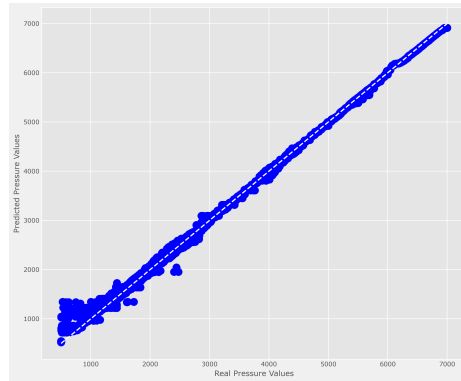


Figure 5.8: XGBoost predictions vs real values

In figure 5.8 we can visualize the performance of our ensemble model, as it is clear, the performance decreases when compared to the NNs, this is due to how different models learn as we saw in chapter 3. We can visualize that the model is having problems when handling lower values, as a result, we have more dispersion in the predicted values.

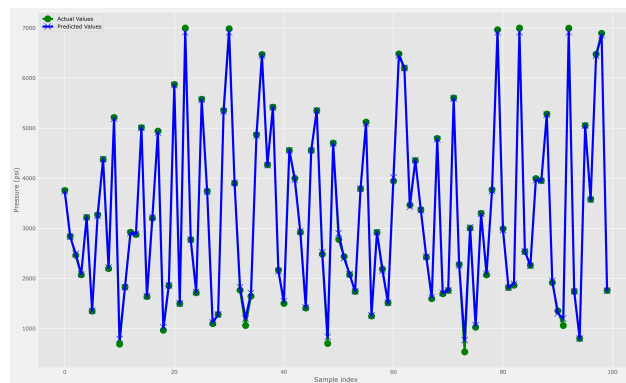


Figure 5.9: XGBoost predictions vs real values

In figure 5.9 we can also see the lower performance of the model, analyzing that the predictions are further away from the actual values.

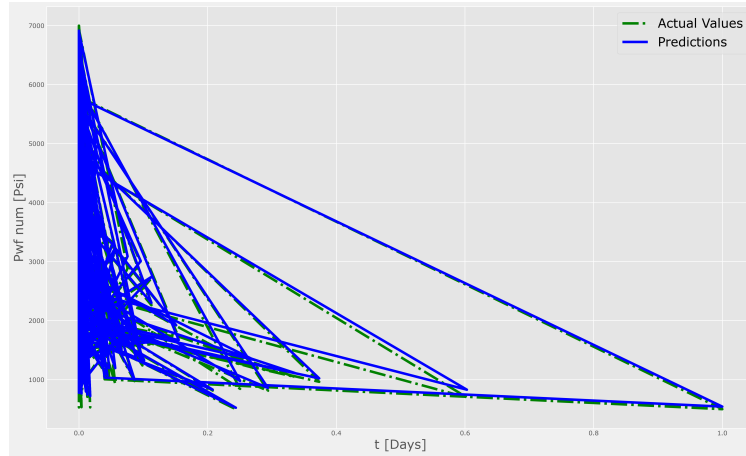


Figure 5.10: XGBoost predictions vs real values of target feature

In this figure is evident how in some parts of the visualization the predictions are slightly distant from the real values but having overall good performance.

5.2 Predictive models comparison

Metric	DFNN	LSTM	XGBoost
MAE	5.4483	1.6886	13.2328
MSE	246.283	173.4291	1141.2776
R^2	0.9998	0.9999	0.9992
MAPE (%)	0.5544%	0.1159%	1.544%
Training Time (min)	15	28	<1

Table 5.1: Evaluation metrics comparison for different models

The table above is a summary of the results of the test set metrics. It can be observed that:

1.-For the LSTM model, the MAE metric indicates that on average its predictions are closer to the real values compared to the DFNN and the XGBoost models.

2.-The three models present a high coefficient of determination indicating all of them fit the data very well and manage to capture nearly all the patterns or variations present in the dataset.

3.-The XGBoost is the most time-efficient model, training in less than one minute, whereas DFNN and LSTM require significantly more time. Although XGBoost is very fast, its performance on error metrics is worse. Conversely, LSTM achieves the highest accuracy but at the cost of longer training time. Therefore, depending on the application, if precision is critical, LSTM is the preferred choice; if speed is a priority, XGBoost may be more suitable.

5.3 Case Studies: Model Implementation and Performance

So far we have seen the performance of the models with the data used to train and test them. This chapter aims to evaluate the models with unseen data by comparing the results obtained by the models against information not seen during training. To achieve this, three runs of the original simulator were carried out using random input data.

Although we are working with synthetic cases, these constitute a reliable basis for analysis, as they allow different conditions to be represented in a controlled manner, ensure the consistency of results, and provide a reference against which to compare models.

Every use case is structured as follows:

- Simulator input parameters, summarized in a table describing the initial conditions of the scenario.
- Simulator results, considered as reference or “real” values, including representative tables and the flowing bottom-hole pressure graphs.
- Results of deep learning models, obtained from the same input parameters.
- Comparison and analysis, discussing similarities and discrepancies between both outputs, based on error metrics and comparative visualizations.

This way, the case studies not only show how the models behave when faced with unseen information but also identify their strengths and limitations in approximating the simulated phenomenon.

5.3.1 Use case 1

This case study aims to analyze the behavior of the bottom-hole flowing pressure (P_{wf}) in a synthetic dry gas reservoir. The system represents a formation with natural fractures and high-conductivity hydraulic fractures, without the presence of a liquid phase, allowing the evaluation of single-phase gas flow under isothermal and homogeneous conditions.

In figure 5.11 we can appreciate the evolution of the flowing bottomhole pressure (P_{wf}) which exhibits three characteristic flow regimes. At early times, the pressure exhibits a rapid and nearly linear decline dominated by flow through high-conductivity fractures. As time passes, the pressure response stabilizes, marking the transition from fracture-controlled flow to matrix-fracture interaction, where gas transfer occurs under an unsteady interporosity flow regime and the pressure difference between both media gradually decreases. At late times, the pressure (P_{wf}) declines again before reaching a new stabilization trend, reflecting overall reservoir depletion and the establishment of a pseudo-steady-state regime characterized by a uniform pressure response throughout the system.

Parameters	Values	Units
Initial pressure	5000	[Psi]
Gas production rate q_g	-500	[ft ³ /day]
Reservoir length in x direction	1300	[ft]
Reservoir length in y direction	1300	[ft]
Matrix porosity	0.0827	[Fraction]
Fracture porosity	0.0001032	[Fraction]
Natural fracture width	0.01	[ft]
Horizontal well length in x direction	1300	[ft]
Well radius r_w	0.2	[ft]
Viscosity μ	0.02	[cp]
Hydraulic fracture permeability	10000	[md]
Hydraulic fracture porosity	0.0001032	[Fraction]
Initial time differential Δt_0	1.05E-08	[Days]
Time steps	955	[-]

Table 5.2: Simulator and Model Inputs

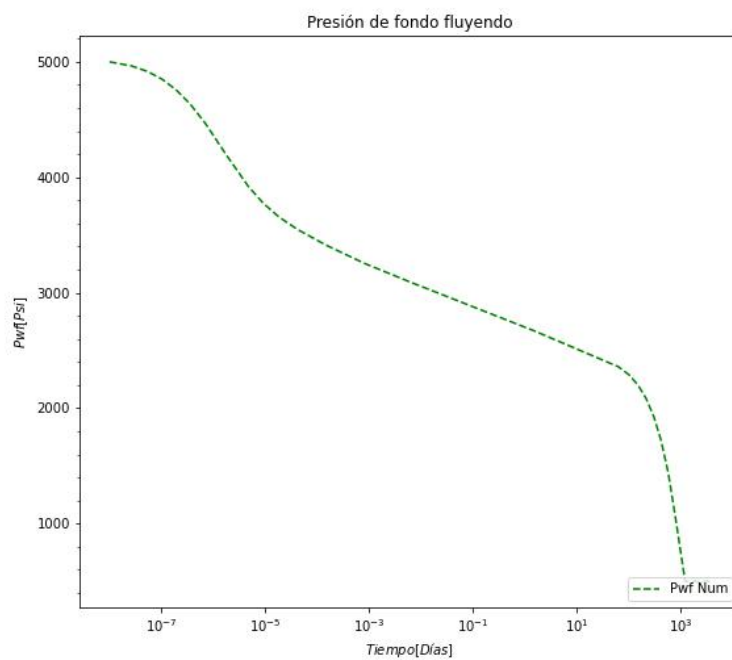


Figure 5.11: Use case 1 simulator graph

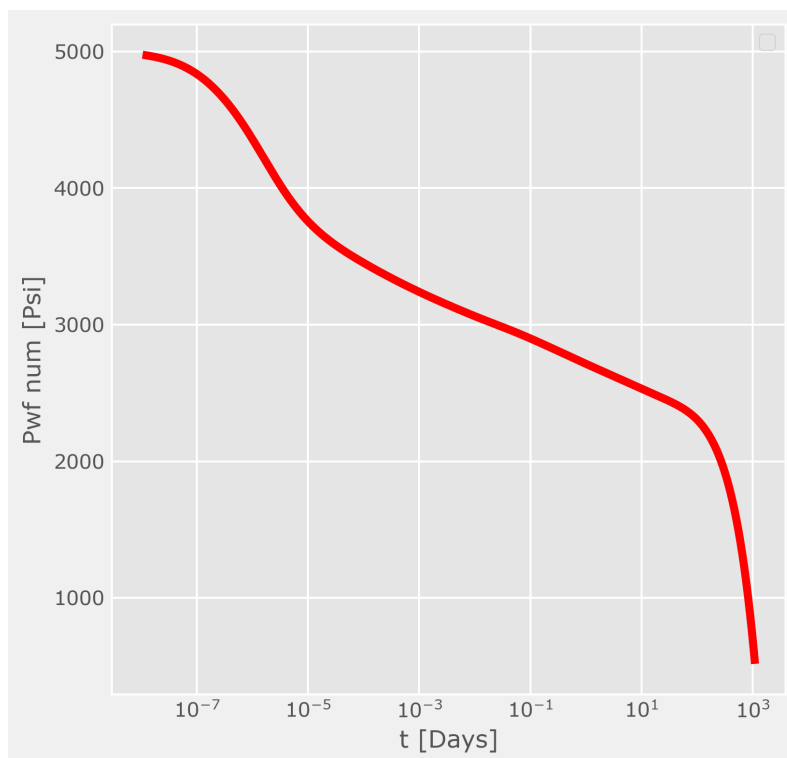


Figure 5.12: DFNN predictions for use case 1

t [Days]	Pwf (Predictions)	Pwf (Real)	% Error
1.05469E-08	4984.2294	4998.6477	0.29%
1.08315E-08	4983.5522	4997.9607	0.31%
1.11237E-08	4982.8618	4997.2616	0.27%
1.14238E-08	4982.1601	4996.5500	0.30%
1.17320E-08	4981.4438	4995.8256	0.28%
1.20486E-08	4980.7163	4995.0878	0.32%
1.23737E-08	4979.9741	4994.3363	0.26%
1.27075E-08	4979.2187	4993.5707	0.29%
1.30504E-08	4978.4482	4992.7905	0.31%
1.34025E-08	4977.6635	4991.9953	0.27%
1.37641E-08	4976.8623	4991.1847	0.28%
1.41355E-08	4976.0458	4990.3583	0.25%
1.45168E-08	4975.2148	4989.5155	0.30%
1.49085E-08	4974.3666	4988.6560	0.29%
Average % Error			0.4%

Table 5.3: DFNN Predictions vs Real Values

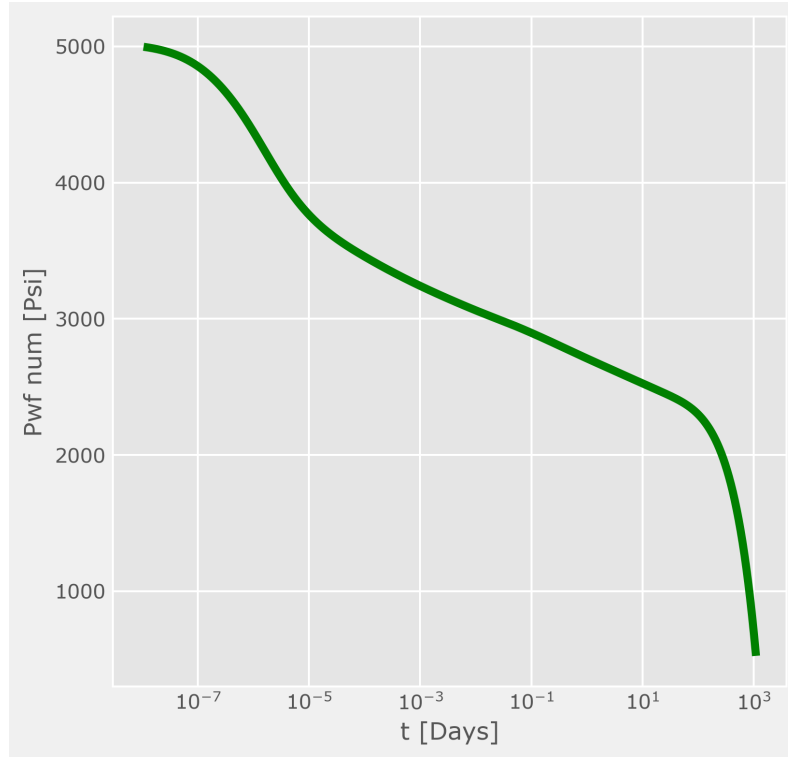


Figure 5.13: LSTM predictions for use case 1

t [Days]	Pwf (Predictions)	Pwf (Real)	% Error
1.05469E-08	4993.5649	4998.6477	0.1017%
1.08315E-08	4992.8798	4997.9607	0.1017%
1.11237E-08	4992.1821	4997.2616	0.1016%
1.14238E-08	4991.4716	4996.5500	0.1016%
1.17320E-08	4990.7490	4995.8256	0.1016%
1.20486E-08	4990.0126	4995.0878	0.1016%
1.23737E-08	4989.2626	4994.3363	0.1016%
1.27075E-08	4988.4985	4993.5707	0.1016%
1.30504E-08	4987.7202	4992.7905	0.1016%
1.34025E-08	4986.9267	4991.9953	0.1015%
Average % Error			0.04%

Table 5.4: LSTM Predictions vs Real Values

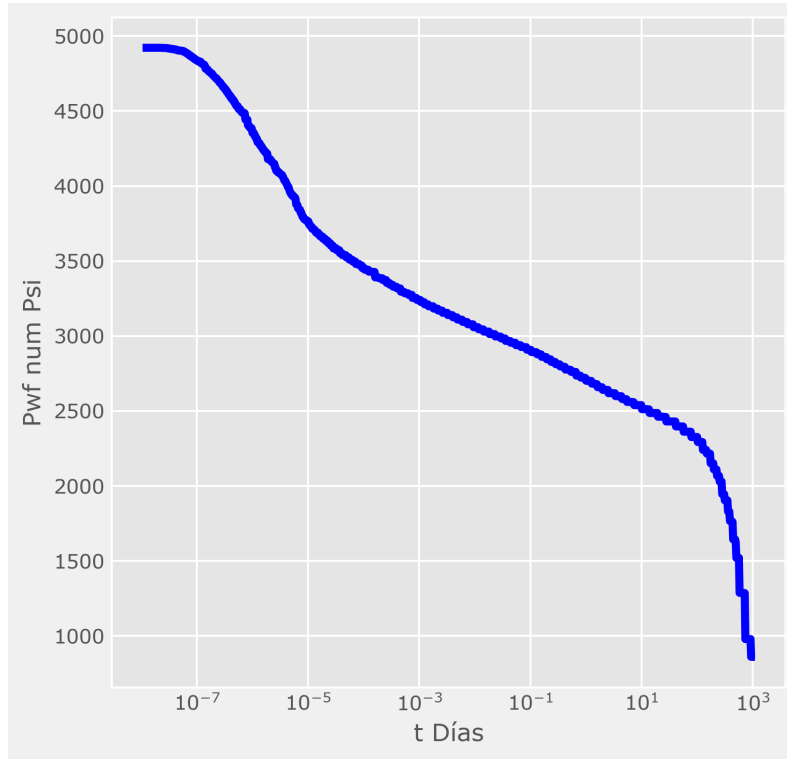


Figure 5.14: XGBoost predictions for use case 1

t [Days]	Pwf (Predictions)	Pwf (Real)	% Error
1.05469E-08	4919.9296	4998.6477	1.5748%
1.08315E-08	4919.9296	4997.9607	1.5613%
1.11237E-08	4919.9296	4997.2616	1.5475%
1.14238E-08	4919.9296	4996.5500	1.5335%
1.17320E-08	4919.9296	4995.8256	1.5192%
1.20486E-08	4919.9296	4995.0878	1.5046%
1.23737E-08	4919.9296	4994.3363	1.4898%
1.27075E-08	4919.9296	4993.5707	1.4747%
1.30504E-08	4919.9296	4992.7905	1.4593%
1.34025E-08	4919.9296	4991.9953	1.4436%
Average % Error			0.78%

Table 5.5: XGBoost Predictions vs Real Values

With these results, we can now be sure that our models are getting the behavior as expected. The LSTM model had the best performance, having 0.04% error. The XGBoost was the "worst" model despite having a percentage of error less than 1%.

Although the predictions look promising, the 3 models had their worst performance when making the firsts and lasts predictions, reaching almost a 70% error rate with the XGBoost model. It is also important to mention that all 3 models generated the predictions in less than 1 minute, this represents a 93% of time reduction.

The pressure behavior is consistent in the predictions of the 3 models with the semi-analytical results reported by (Teja-Juárez et al., 2025), where the transient-to-steady-state transition is captured through the Streltsova-type interporosity flow function. Such behavior confirms the model's ability to reproduce the characteristic flow regimes of naturally fractured gas reservoirs under single-phase conditions.

5.3.2 Use case 2

This case study aims to analyze the behavior of the bottom-hole flowing pressure (P_{wf}) in a synthetic dry gas reservoir. The system represents a formation with natural fractures and high-conductivity hydraulic fractures, without the presence of a liquid phase, allowing the evaluation of single-phase gas flow under isothermal and homogeneous conditions.

In figure 5.15 we can appreciate the evolution of the flowing bottomhole pressure (P_{wf}) which exhibits three characteristic flow regimes. At early times, the pressure exhibits a rapid and nearly linear decline dominated by flow through high-conductivity fractures. As time passes, the pressure response stabilizes, marking the transition from fracture-controlled flow to matrix-fracture interaction and the pressure difference between both media gradually decreases. At late times, the pressure (P_{wf}) declines again before reaching a new stabilization trend, reflecting overall reservoir depletion and the establishment of a pseudo-steady-state regime.

Parameters	Values	Units
Initial pressure	6000	[Psi]
Gas production rate q_g	-800	[ft ³ /day]
Reservoir length in x direction	900	[ft]
Reservoir length in y direction	900	[ft]
Matrix porosity	0.08421	[Fraction]
Fracture porosity	0.0001019	[Fraction]
Natural fracture width	0.01	[ft]
Horizontal well length in x direction	900	[ft]
Well radius r_w	0.25	[ft]
Viscosity μ	0.0278	[cp]
Hydraulic fracture permeability	16000	[md]
Hydraulic fracture porosity	0.0001020	[Fraction]
Initial time differential Δt_0	1.05E-08	[Days]
Time steps	928	[-]

Table 5.6: Simulator and Model Inputs

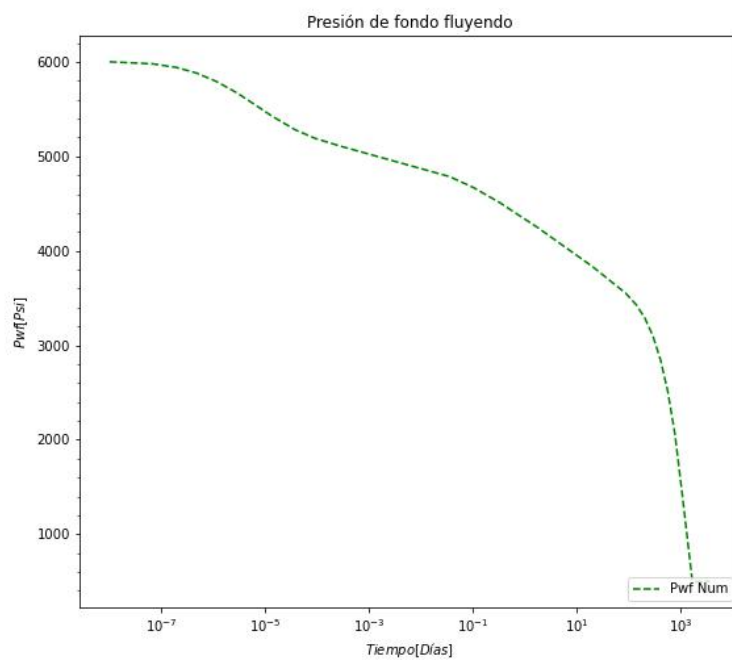


Figure 5.15: Use case 2 Simulator graph

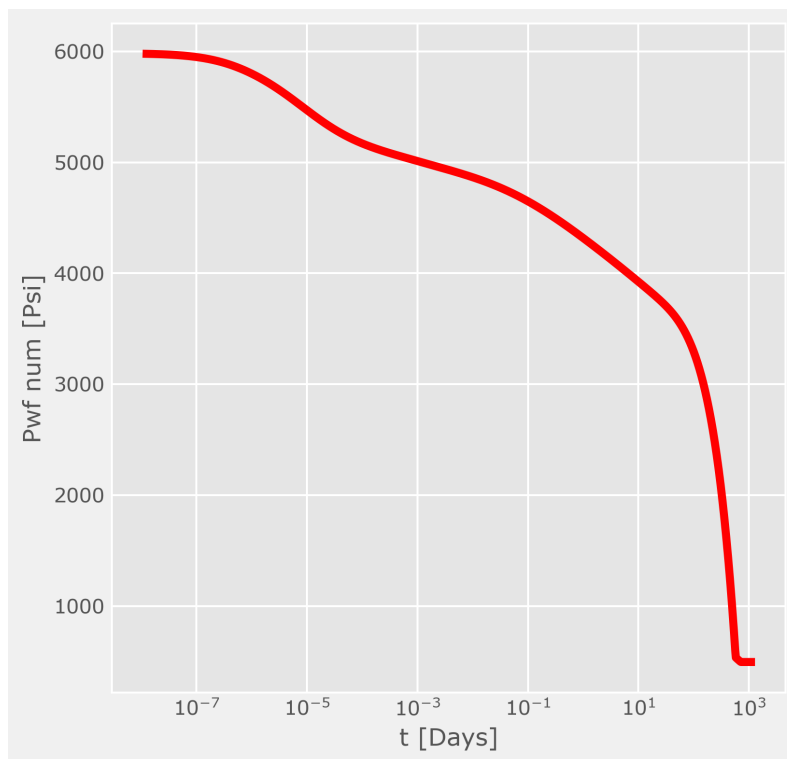


Figure 5.16: DFNN predictions for use case 2

t [Days]	Pwf (Predictions)	Pwf (real)	% Error
1.0547E-08	5983.9653	5999.7811	0.2636%
1.0832E-08	5983.8550	5999.6680	0.2636%
1.1124E-08	5983.7388	5999.5520	0.2636%
1.1424E-08	5983.6221	5999.4332	0.2635%
1.1732E-08	5983.5020	5999.3113	0.2635%
1.2049E-08	5983.3779	5999.1864	0.2635%
1.2374E-08	5983.2524	5999.0584	0.2635%
1.2708E-08	5983.1211	5998.9271	0.2635%
1.3050E-08	5982.9893	5998.7926	0.2634%
1.3402E-08	5982.8530	5998.6547	0.2634%
1.3764E-08	5982.7129	5998.5133	0.2634%
1.4136E-08	5982.5703	5998.3684	0.2634%
1.4517E-08	5982.4238	5998.2198	0.2633%
1.4909E-08	5982.2749	5998.0675	0.2633%
1.5311E-08	5982.1191	5997.9114	0.2633%
Average % Error			0.2546%

Table 5.7: DFNN Predictions vs Real Values

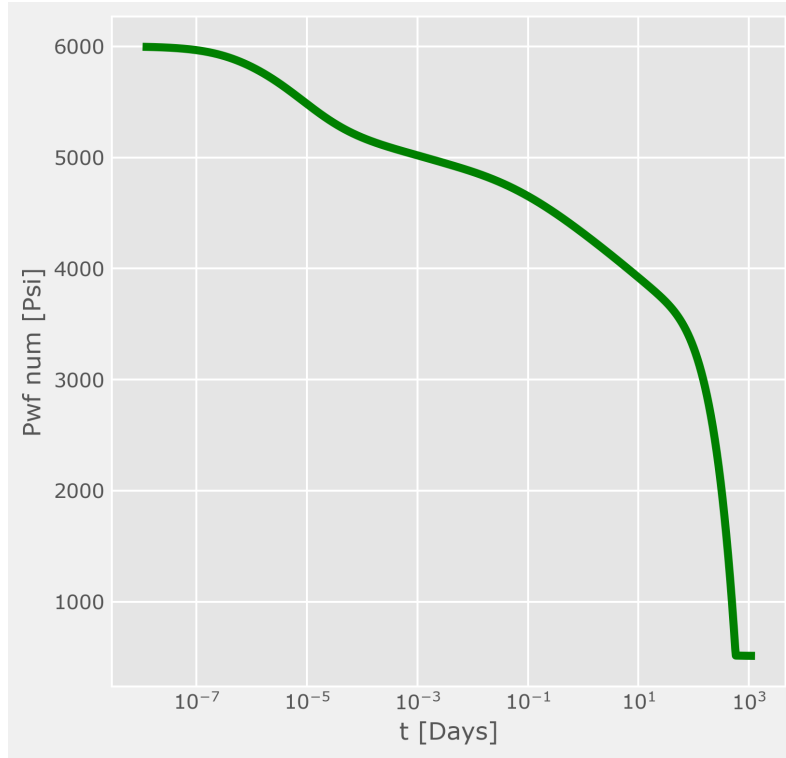


Figure 5.17: LSTM predictions for use case 2

t [Days]	Pwf (Predictions)	Pwf (real)	% Error
1.0547E-08	5992.8257	5999.7811	0.1159%
1.0832E-08	5992.7124	5999.6680	0.1159%
1.1124E-08	5992.5972	5999.5520	0.1159%
1.1424E-08	5992.4785	5999.4332	0.1159%
1.1732E-08	5992.3569	5999.3113	0.1159%
1.2049E-08	5992.2334	5999.1864	0.1159%
1.2374E-08	5992.1045	5999.0584	0.1159%
1.2708E-08	5991.9741	5998.9271	0.1159%
1.3050E-08	5991.8394	5998.7926	0.1159%
1.3403E-08	5991.7017	5998.6547	0.1159%
1.3764E-08	5991.5605	5998.5133	0.1159%
1.4136E-08	5991.4160	5998.3684	0.1159%
1.4517E-08	5991.2686	5998.2198	0.1159%
1.4909E-08	5991.1162	5998.0675	0.1159%
1.5311E-08	5990.9604	5997.9114	0.1159%
Average % Error			0.0927%

Table 5.8: LSTM Predictions vs Real Values

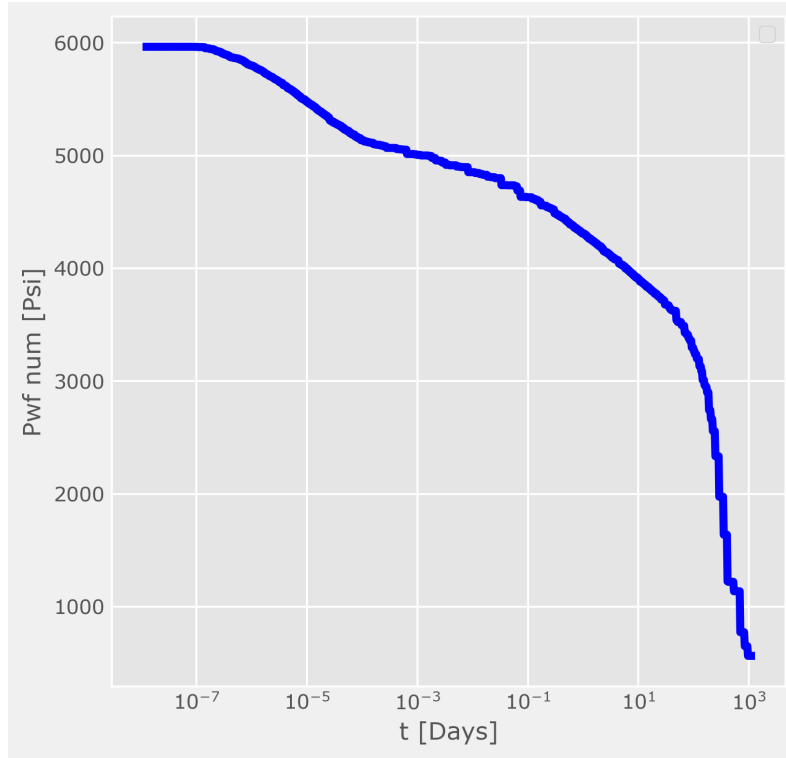


Figure 5.18: XGBoost predictions for use case 2

t [Days]	Pwf (Predictions)	Pwf (real)	% Error
1.0547E-08	5963.9624	5999.7811	0.5970%
1.0832E-08	5963.9624	5999.6680	0.5951%
1.1124E-08	5963.9624	5999.5520	0.5932%
1.1424E-08	5963.9624	5999.4332	0.5912%
1.1732E-08	5963.9624	5999.3113	0.5892%
1.2049E-08	5963.9624	5999.1864	0.5871%
1.2374E-08	5963.9624	5999.0584	0.5850%
1.2708E-08	5963.9624	5998.9271	0.5828%
1.3050E-08	5963.9624	5998.7926	0.5806%
1.3403E-08	5963.9624	5998.6547	0.5783%
1.3764E-08	5963.9624	5998.5133	0.5760%
1.4136E-08	5963.9624	5998.3684	0.5736%
1.4517E-08	5963.9624	5998.2198	0.5711%
1.4909E-08	5963.9624	5998.0675	0.5686%
1.5311E-08	5963.9624	5997.9114	0.5660%
Average % Error			0.8341%

Table 5.9: XGBoost Predictions vs Real Values

With these results, we can now be sure that our models are getting the behavior as expected. The LSTM model had the best performance, having 0.04% error. The XGBoost was the "worst" model despite having a percentage of error of 0.78%.

Although the predictions are consistent, the 3 models struggled when making the firsts and lasts predictions. It is important to mention that all 3 models generated the predictions in less than 1 minute, this represents a 93% of time reduction.

The pressure behavior is consistent in the predictions of the 3 models with the semi-analytical results reported by (Teja-Juárez et al., 2025), where the transient-to-steady-state transition is captured through the Streltsova-type interporosity flow function. Such behavior confirms the model's ability to reproduce the characteristic flow regimes of naturally fractured gas reservoirs under single-phase conditions.

5.3.3 Use case 3

This case study aims to analyze the behavior of the bottom-hole flowing pressure (P_{wf}) in a synthetic dry gas reservoir. The system represents a formation with natural fractures and high-conductivity hydraulic fractures, without the presence of a liquid phase, allowing the evaluation of single-phase gas flow under isothermal and homogeneous conditions.

In figure 5.19 we can appreciate the evolution of the flowing bottomhole pressure (P_{wf}) which exhibits three characteristic flow regimes. At early times, the pressure exhibits a rapid and nearly linear decline dominated by flow through high-conductivity fractures. As time passes, the pressure response stabilizes, marking the transition from fracture-controlled flow to matrix-fracture interaction, where gas transfer occurs under an unsteady interporosity flow regime and the pressure difference between both media gradually decreases. At late times, the pressure (P_{wf}) declines again before reaching a new stabilization trend, reflecting overall reservoir depletion and the establishment of a pseudo-steady-state regime characterized by a uniform pressure response throughout the system.

Parameters	Values	Units
Initial pressure	3000	[Psi]
Gas production rate q_g	-200	[ft ³ /day]
Reservoir length in x direction	900	[ft]
Reservoir length in y direction	900	[ft]
Matrix porosity	0.0834	[Fraction]
Fracture porosity	0.0001049	[Fraction]
Natural fracture width	0.0001	[ft]
Horizontal well length in x direction	900	[ft]
Well radius r_w	0.27	[ft]
Viscosity μ	0.015	[cp]
Hydraulic fracture permeability	8000	[md]
Hydraulic fracture porosity	0.0001049	[Fraction]
Initial time differential Δt_0	1.05E-08	[Days]
Time steps	948	[-]

Table 5.10: Simulator and Model Inputs

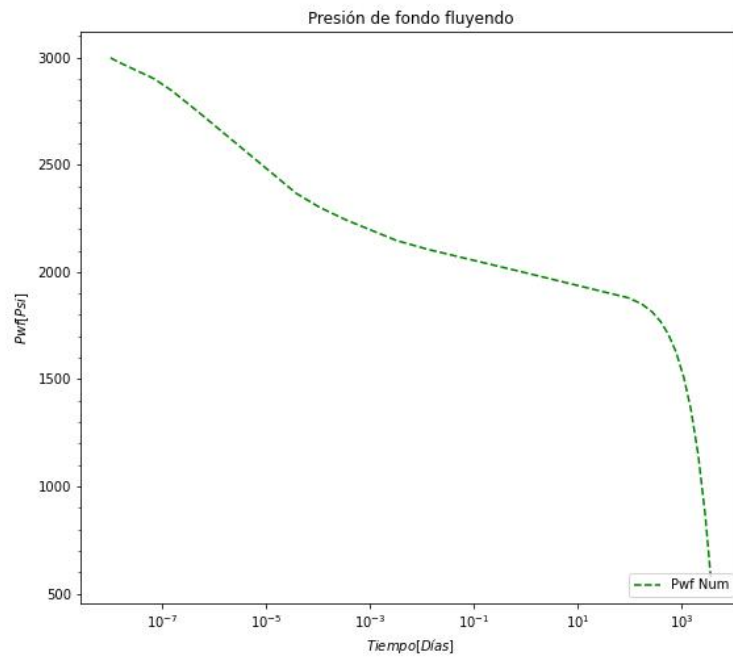


Figure 5.19: Use case 3 simulator graph

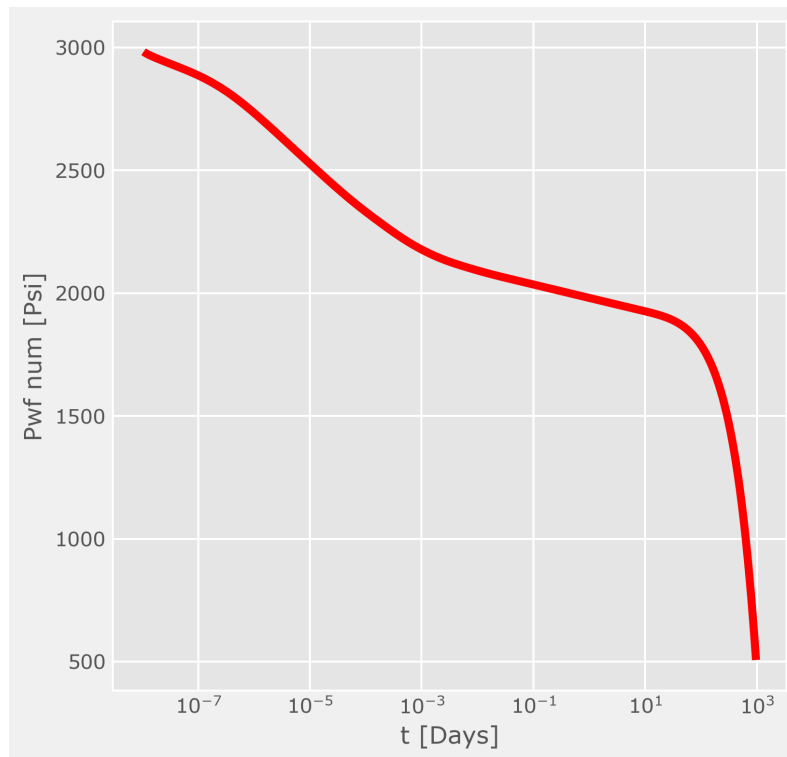


Figure 5.20: DFNN predictions for use case 3

t [Days]	Pwf (Predictions)	Pwf (Real)	% Error
1.0547E-08	2989.9980	2995.7959	0.1935%
1.0832E-08	2988.1306	2993.9046	0.1929%
1.1124E-08	2986.3782	2992.1287	0.1922%
1.1424E-08	2984.7234	2990.4535	0.1916%
1.1732E-08	2983.1565	2988.8657	0.1910%
1.2049E-08	2981.6643	2987.3537	0.1904%
1.2374E-08	2980.2358	2985.9068	0.1899%
1.2708E-08	2978.8628	2984.5159	0.1894%
1.3050E-08	2977.5366	2983.1729	0.1889%
1.3403E-08	2976.2517	2981.8706	0.1884%
1.3764E-08	2974.9995	2980.6029	0.1880%
1.4136E-08	2973.7776	2979.3645	0.1875%
1.4517E-08	2972.5793	2978.1508	0.1871%
1.4909E-08	2971.4021	2976.9580	0.1866%
1.5311E-08	2970.2417	2975.7828	0.1862%
Average % Error			0.2311%

Table 5.11: DFNN Predictions vs Real Values

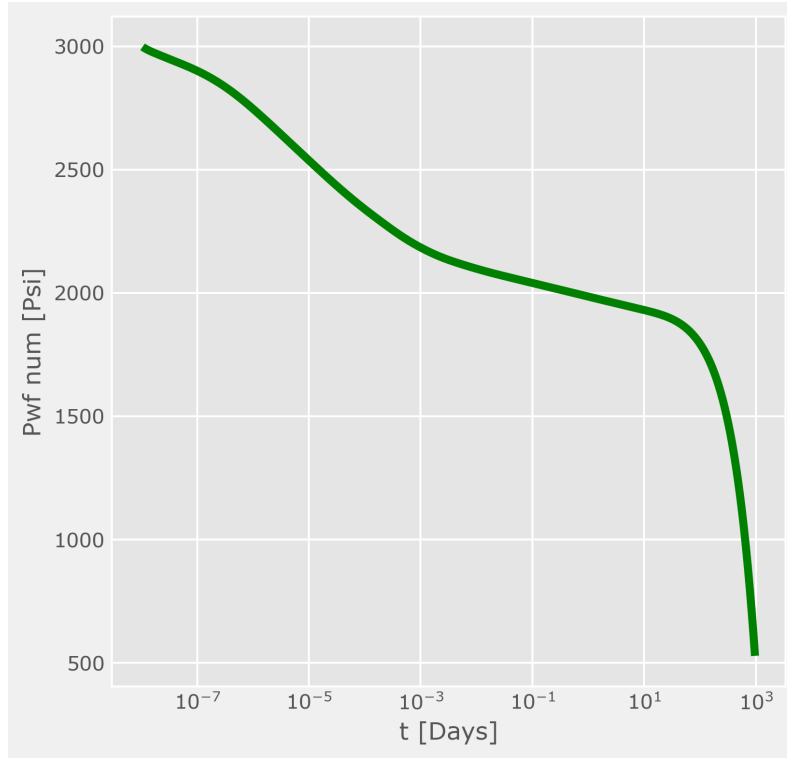


Figure 5.21: LSTM predictions for use case 3

t [Days]	Pwf (Predictions)	Pwf (Real)	% Error
1.0547E-08	2991.2239	2995.7959	0.1526%
1.0832E-08	2989.3367	2993.9046	0.1526%
1.1124E-08	2987.5647	2992.1287	0.1525%
1.1424E-08	2985.8931	2990.4535	0.1525%
1.1732E-08	2984.3086	2988.8657	0.1525%
1.2049E-08	2982.7996	2987.3537	0.1524%
1.2374E-08	2981.3557	2985.9068	0.1524%
1.2708E-08	2979.9678	2984.5159	0.1524%
1.3050E-08	2978.6272	2983.1729	0.1524%
1.3403E-08	2977.3279	2981.8706	0.1523%
1.3764E-08	2976.0630	2980.6029	0.1523%
1.4136E-08	2974.8271	2979.3645	0.1523%
1.4517E-08	2973.6160	2978.1508	0.1523%
1.4909E-08	2972.4255	2976.9580	0.1523%
1.5311E-08	2971.2527	2975.7828	0.1522%
Average % Error			0.1468%

Table 5.12: LSTM Predictions vs Real Values

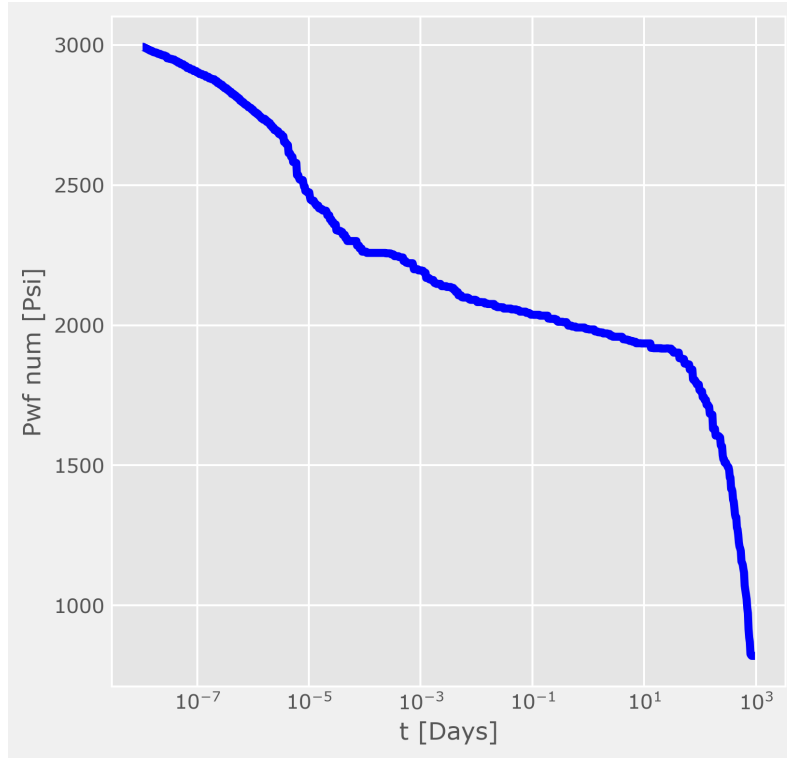


Figure 5.22: XGBoost predictions for use case 3

t [Days]	Pwf (Predictions)	Pwf (Real)	% Error
1.0547E-08	2993.0168	2995.7959	0.0928%
1.0832E-08	2993.0168	2993.9046	0.0297%
1.1124E-08	2991.6343	2992.1287	0.0165%
1.1424E-08	2989.6460	2990.4535	0.0270%
1.1732E-08	2988.9758	2988.8657	0.0037%
1.2049E-08	2988.4834	2987.3537	0.0378%
1.2374E-08	2986.6479	2985.9068	0.0248%
1.2708E-08	2985.1775	2984.5159	0.0222%
1.3050E-08	2985.1775	2983.1729	0.0672%
1.3403E-08	2983.4163	2981.8706	0.0518%
1.3764E-08	2981.4875	2980.6029	0.0297%
1.4136E-08	2981.4875	2979.3645	0.0713%
1.4517E-08	2978.9441	2978.1508	0.0266%
1.4909E-08	2978.9441	2976.9580	0.0667%
1.5311E-08	2978.9441	2975.7828	0.1062%
Average % Error			1.184%

Table 5.13: XGBoost Predictions vs Real Values

With these results, we can now be sure that our models are getting the behavior as expected. The LSTM model had the best performance, having 0.14% error. The XGBoost was the "worst" model despite having a percentage of error less than 1%.

Although the predictions look promising, the 3 models had their worst performance when making the firsts and lasts predictions, reaching a little more than 1% error rate with the XGBoost model. It is also important to mention that all 3 models generated the predictions in less than 1 minute, this represents a 93% of time reduction.

The pressure behavior is consistent in the predictions of the 3 models with the semi-analytical results reported by (Teja-Juárez et al., 2025), where the transient-to-steady-state transition is captured through the Streltsova-type interporosity flow function. Such behavior confirms the model's ability to reproduce the characteristic flow regimes of naturally fractured gas reservoirs under single-phase conditions.

Conclusion

This thesis had the purpose of developing AI models to predict the pressure profile that occurs in the fluid flow of a fractured reservoir, so that we have an alternative when estimating oil recovery in a hydrocarbon reservoir. Throughout the thesis, a methodological framework was developed, encompassing the understanding of why fractured reservoirs represent a challenge, how reservoir simulation in fractured reservoirs is carried out nowadays, the collection of data used to train the models, the preparation, implementation and evaluation of different DL algorithms through standard metrics.

The results obtained demonstrated that DL models, despite being more complex to train and understand the logic behind them, provide a better ability to generalize in scenarios characterized by low data variability; in other words, in this case the algorithms were more efficient than a traditional deterministic model.

In terms of performance, the models presented a great efficiency despite the data did not have a strong correlation. The LSTM model was the most accurate and the most consistent, The DFNN stood out for being the most balanced model in terms of precision and complexity, The XGBoost model although it was not the most accurate, his accuracy was not bad at all, and it was the fastest to train. The results confirmed the initial hypothesis of which the models can indeed replicate the outcome of a traditional reservoir simulator, and in the meantime the models reduced the simulation time.

Future work should focus on replicating the methodology proposed in this thesis using a larger dataset with greater variability in the input variables to further evaluate the performance and generalization capability of the developed models. In addition, it is recommended to to create a Graphical User Interface (GUI) so it will be easier for the final user to generate new predictions and to obtain graphs in different formats, in different devices.

Bibliography

- Aguilera, R. (1995). *Naturally fractured reservoirs*. PennWell Books.
- AndrewTch. (2017). The mostly complete chart of neural networks, explained. *Towards Data Science*.
- Barenblatt, G. I., Zheltov, I. P., & Kochina, I. N. (1960). Basic concepts in the theory of seepage of homogeneous liquids in fissured rocks. *Journal of Applied Mathematics and Mechanics*, 24(5), 1286–1303.
- Bourbiaux, B. (2010). Fractured reservoir simulation: A challenging and rewarding issue. *Oil & Gas Science and Technology – Revue d’IFP Energies nouvelles*, 65(2), 227–238. <https://doi.org/10.2516/ogst/2009052>
- Buduma, N. (2017). *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. O’Reilly Media.
- Ghassemzadeh, S. (2020). *A novel approach to reservoir simulation using supervised learning* [Doctoral dissertation, University of Adelaide].
- Guerra Abad, J. (2002). *Nuevo modelo de flujo para yacimientos naturalmente fracturados* [Master’s thesis, Universidad Nacional Autónoma de México].
- Hubbert, M. K., & Willis, D. G. (1955). Important fractured reservoirs in the united states [Section I/A-1]. *Proceedings of the 4th World Petroleum Congress*, 58–81.
- Ian Goodfellow, Y. B., & Courville, A. (2016). *Deep learning*. MIT Press.
- Kazemi, H. (1969). Pressure transient analysis of naturally fractured reservoirs with uniform fracture distribution. *Society of Petroleum Engineers Journal*, 9(4), 451–462. <https://doi.org/10.2118/2156-PA>
- Nelson, R. A. (2001). *Geologic analysis of naturally fractured reservoirs* (2nd). Gulf Professional Publishing.
- Olah, C. (2015). Understanding lstm networks. *Blog*.
- Pandas. (n.d.). `Pandas.dataframe.describe`. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html>

-
- Pollitt, S. S. D. A. (2020). Optimising development and production of naturally fractured reservoirs using a large empirical dataset. *GSL*.
- Roos, M. (2019). Deep learning neurons versus biological neurons. *Medium*.
- Sahin, S. (2025). *Regularization in regression: Tackling overfitting for enhanced model performance*. <https://medium.com/@sahin.samia/mregularization-in-regression-tackling-overfitting-for-enhanced-model-performance-553f2cef02df#:~:text=Definition%20and%20Explanation,performance%20on%20new%2C%20unseen%20data>.
- Samaniego, R., & Cinco Ley, H. (1982). Pressure transient analysis for naturally fractured reservoirs. *Journal of Petroleum Technology*.
- Scikit-Learn. (n.d.). Cross-validation: Evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html
- Scikit-learn. (2025). Mean absolute percentage error. https://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-percentage-error
- Stearns, D. W., & Friedman, M. (1972). Reservoirs in fractured rock. In R. E. King (Ed.), *Stratigraphic oil and gas fields: Classification, exploration methods, and case histories* (Vol. 16). American Association of Petroleum Geologists.
- Streltsova, I. (1983). Well pressure behavior of a naturally fractured reservoir. *SPE Formation Evaluation*, 8(1), 89–96.
- Teja-Juárez, V. L., Torres-López, J. A., Mariscal-Romero, R. M., Camacho-Velázquez, R. G., Gómora-Figueroa, A. P., & Solano-Rojas, D. E. (2025). *A semi-analytical model to simulate fluid flow in fractured reservoirs*. <https://doi.org/10.1007/s11242-025-02203-2>
- Tiab, D., & Donaldson, E. C. (2016). Copyright. In *Petrophysics (fourth edition)* (Fourth Edition, pp. 485–492). Gulf Professional Publishing. <https://doi.org/https://doi.org/10.1016/B978-0-12-803188-9.09990-3>
- Warren, J., & Root, P. (1963). The behavior of naturally fractured reservoirs. *Society of Petroleum Engineers Journal*, 3(3), 245–255.
- Weidman, S. (2019). *Deep learning from scratch: Building with python from first principles*. O'Reilly Media.
- Wu, Y.-S. (2016). *Multiphase fluid flow in porous and fractured reservoirs*. Elsevier.