Physics Informed Deep Learning : Applications to well opening and closing events

Antoine Lechevallier



Doctoral Thesis

Physics Informed Deep Learning

Application to well opening and closing events

---

by

Antoine Lechevallier

A thesis submitted and presented publicly 22/02/2024

in fulfillment of the requirements for the

Doctorate of the École Doctorale de Sciences Mathématiques de Paris Centre

Thesis director:	Frédéric Nataf
Co-supervisors:	Thibault Faney
	Eric Flauraud
	Sylvain Desroziers
Rapporteurs:	
	Emmanuel Frénod
	Alexander Heinlein
Jury president:	
	Bruno Després
Jury:	
	Tor Harald Sandve

Acknowledgements

I would like to begin by extending my sincere gratitude to DIM Math Innov for their financial support. Additionally, I wish to express my sincere apprecitation to IFP Energies Nouvelles and the Jacques Louis-Lions laboratory for their unwavering financial and logistical assistance throughout this research endeavor. Their support not only provided a conducive working environment but also afforded me the privilege of engaging with fellow researchers at conferences. Without their generous assistance, the completion of this thesis would not have been achievable.

Heartfelt appreciation goes to my thesis supervisors who provided me a very healthy and stimulating research environment. Firstly, I am deeply grateful to Frédéric Nataf for his kindness and dedication to the thesis work. His passion and creativity for the subject has been inspiring throughout this journey. Secondly, I express my sincere thanks to Thibault Faney for his sage counsel and infectious enthusiasm, which have been invaluable in guiding me through this work. Additionally, I extend my genuine gratitude to Sylvain Desroziers, who consistently made time to monitor the progress of the thesis and offer valuable insights, despite his busy schedule. Special acknowledgment is also due to Eric Flauraud, who, upon learning about the thesis, provided continuous support and guidance throughout the entire process, despite not being officially designated as a supervisor. His involvement has been greatly appreciated.

I would also like to thank the comity members including the jury and rapporteurs, for their valuable comments and suggestions regarding this thesis work. Their insightful input were truely heartwarming to the young researcher I am.

A very special thanks goes to my colleagues and friends for the time we spent together and for the time I have the privilege to hope that we will spend together in the future.

I owe a debt of gratitude to my partner, Morgane Vives, whose unwavering love, support, and wisdom have been my anchor during this journey. Additionally, I want to extend my thanks to my beloved feline companion, BOULETTE, for providing endless comfort, companionship and madness during study sessions.

Finally, I would like to extend my acknowledgements to my family, which members have always supported me throughout my entire journey. I could not thank you enough.

Abstract

The reduction of CO_2 emission into the atmosphere is mandatory to achieve ecological transition. CO_2 geological storage is an essential instrument for efficient Carbon Capture and Storage (CCS) policies.

Numerical simulations provide the solution to the multi-phase flow equations that model the behavior of the CO_2 injection site. They are an important tool to decide either or not to exploit a potential carbon storage site and to monitor the operations (potential gas leakage, optimal positioning of CO_2 injection wells, etc.).

However, numerical simulations of fluid flow in porous media are computationally demanding: it can take up to several hours on a HPC cluster in order to simulate one injection scenario for a large CO_2 reservoir if we want to accurately model the complex physical processes involved.

More specifically, well events (opening and closure) cause important numerical difficulties due to their instant impact on the system. This often forces a drastic reduction of the time step size to be able to solve the non-linear system of equations resulting from the discretization of the continuous mathematical model. However, these specific well events in a simulation are relatively similar across space and time: the degree of similarity between two well events depends on a few parameters such as the injection condition, the state of the reservoir at the time of the event, the boundary conditions or the porous media parameters (permeability and porosity) around each well.

Recent interest in machine learning applied to the prediction of physical processes has fueled the development of "Physics Informed Deep Learning", where machine learning models either replace or complement traditional numerical algorithms while preserving the inherent constraints from the physical model.

Therefore, the objective of this thesis is to adapt recent advances in physics informed deep learning in order to alleviate the impact of well events in the numerical simulation of multiphase flow in porous media. Our main contributions are separated in three parts.

In the first part, we replace the traditional numerical solver with a machine-learning model. We demonstrate the feasibility of learning parameter-to-solution operators for partial differential equation problems. However, when utilizing the machine-learning model for time iteration, we observe that the predicted solution diverges from the true solution.

Consequently, in the second part, we use an hybrid approach that complements the traditional non-linear solver with a machine-learning model while preserving numerical guarantees. In practice, we utilize and tailor to our purpose the hybrid Newton methodology, which involves predicting a global initialization for Newton's method closer to the solution than the standard one. We use the state-of-the-art Fourier Neural Operator machine-learning model as a predictive model. Our methodology is applied to two test cases and exhibits promising results by reducing up to 54% the number of Newton iterations compared to a reference method.

In the last part, we apply the hybrid Newton methodology to predict an initialization in the near-well region, where the main variations of CO_2 saturations occur. We investigate the impact of the local domain size and then demonstrate, for a 1D case, that it is possible to learn a local initialization for any well location. Then, we apply this local approach to a 2D case and compare the performances between the hybrid Newton strategy and a Domain Decomposition-inspired strategy. We speed up the handling of well events by around 45% in terms of Newton iterations.

Résumé

La réduction des émissions de CO_2 dans l'atmosphère est primordiale afin d'accomplir la transition écologique. Le stockage géologique du CO_2 est un instrument essentiel parmi les stratégies de capture et de stockage du CO_2.

Les simulations numériques fournissent la solution aux équations de l'écoulement multiphasique qui modélisent le comportement du site d'injection de CO_2. Elles constituent un outil essentiel pour décider de l'exploitation ou non d'un site potentiel de stockage de CO_2.

Cependant, les simulations numériques d'écoulement en milieu poreux sont exigeantes en termes de calcul : il peut falloir plusieurs heures sur un cluster HPC pour simuler un scénario d'injection pour un grand réservoir de CO_2 afin de modéliser avec précision les processus physiques complexes impliqués.

En particulier, les événements liés aux puits (ouvertures et fermetures) posent d'importantes difficultés numériques en raison de leurs impacts immédiat sur le système. Cela force souvent une réduction drastique de la taille du pas de temps afin de résoudre le système d'équations non-linéaires résultant de la discrétisation du modèle mathématique continu. Cependant, ces événements spécifiques liés aux puits sont relativement similaires dans l'espace et le temps : le degré de similitude entre deux événements de puits dépend de quelques paramètres tels que le débit d'injection, l'état du réservoir au moment de l'événement, les conditions aux limites ou les paramètres du milieu poreux (perméabilité et porosité) autour de chaque puits.

L'intérêt récent pour l'application de l'apprentissage automatique à la prédiction des processus physiques a stimulé le développement de la "Physics Informed Deep Learning" (PIDL), où les modèles d'apprentissage automatique remplacent ou complètent les algorithmes numériques traditionnels tout en préservant les contraintes inhérentes au modèle physique.

Par conséquent, l'objectif de cette thèse est d'adapter les avancées récentes en PIDL afin de réduire l'impact des événements de puits dans la simulation numérique des écoulements multiphasiques en milieux poreux. Nos principales contributions sont divisées en trois parties.

Dans la première partie, nous remplaçons le solveur numérique traditionnel

par un modèle d'apprentissage automatique. Nous montrons qu'il semble possible d'apprendre des opérateurs 'parameter-to-solution' pour les problèmes d'équations aux dérivées partielles. Cependant, lorsque nous utilisons le modèle d'apprentissage automatique afin d'itérer en temps, la solution prédite s'éloigne de la solution réelle.

De ce fait, dans la deuxième partie, nous utilisons une approche hybride, qui complète le solveur non-linéaire traditionnel avec un modèle d'apprentissage automatique, tout en préservant les garanties numériques. En pratique, nous utilisons la méthode de Newton hybride, qui consiste à prédire une initialisation globale pour la méthode de Newton plus proche de la solution que l'initialisation standard. Le Fourier Neural Operator est utilisé comme modèle prédictif. Notre méthodologie est appliquée à deux cas tests et présente des résultats prometteurs en réduisant jusqu'à 54% le nombre d'itérations de Newton par rapport à une méthode de référence.

Dans la dernière partie, nous appliquons la méthode de Newton hybride pour prédire une initialisation dans la région proche du puits, où se situent les principales variations de saturations en CO_2. Nous étudions d'abord l'impact de la taille du domaine local et démontrons ensuite, sur un cas 1D, qu'il est possible d'apprendre une initialisation locale précise pour n'importe quel emplacement de puits. Nous appliquons ensuite cette approche locale à un cas 2D et comparons les performances entre la stratégie hybride de Newton et une stratégie inspirée de la décomposition de domaine. Nous accélérons la gestion des événements de puits d'environ 45% en termes d'itérations de Newton.

Contents

1 Introduction **18**
1.1 Context and Motivations 18
1.2 State of the art of Physics Informed Deep Learning 21
1.2.1 Physics Informed Neural Network 23
1.2.2 Neural Operators 33
1.2.3 Physics Informed Deep Learning for CO2 storage 38
1.2.4 Hybrid approaches and hybrid Newton's method 41
1.3 Outline and results summary of the thesis 43

2 Physical problem formulation and use case **54**
2.1 Incompressible two-phase flow problem 54
2.1.1 Standard well model 57
2.2 SHPCO2 benchmark 57
2.2.1 Physical configuration 59

3 Numerical resolution **63**
3.1 Mesh definition and notations 64
3.2 IMPES scheme 66
3.2.1 Solving the pressure problem 66
3.2.2 Solving the saturation problem 68
3.2.3 Taking wells into account 69
3.3 IMPIMS scheme 71
3.4 Fully Implicit scheme 72
3.5 Time-step management 74
3.5.1 Time stepping for each scheme 74
3.6 YADS reservoir simulation library 75
3.6.1 YADS feature blocks 77
3.7 Numerical behaviour during well event 82

4 Black-Box machine learning PDE solver 86
4.1 Problems considered . . . 88
4.1.1 Poisson equation . . . 88
4.1.2 Incompressible two-phase flow problem . . . 90
4.2 Conclusion . . . 98

5 Hybrid Newton Method - Global approach 100
5.1 Introduction . . . 101
5.2 Methodology . . . 103
5.2.1 Hybrid Newton algorithm . . . 103
5.3 Initial guess construction . . . 103
5.3.1 Neural Network architecture . . . 105
5.4 Test case and Database Generation . . . 106
5.4.1 SHPCO2 benchmark . . . 106
5.4.2 Test cases . . . 108
5.5 Results and discussion . . . 111
5.5.1 Neural Network training . . . 111
5.5.2 Results . . . 113
5.5.3 Discussion . . . 116
5.6 Conclusion . . . 117

6 Hybrid Newton Method - Local approach 121
6.1 1D problem . . . 122
6.1.1 Impact of local domain size . . . 124
6.1.2 Impact of well location . . . 130
6.2 2D SHPCO2 . . . 135
6.2.1 Objectives and Workflow . . . 135
6.2.2 Domain Decomposition . . . 138
6.2.3 Data generation . . . 140
6.2.4 Results . . . 141

7 Conclusion and perspectives 153
7.1 Conclusion . . . 153
7.2 Perspectives . . . 155
7.2.1 Local approach - Generalized well model and multi-well simulations . . . 155
7.2.2 Moving towards realistic problems . . . 156

List of Tables

1.1 Comparison of different properties of deep learning models taken from [26]. 36

2.1 1D mesh parameters 59
2.2 2D mesh parameters 60
2.3 Petrophysical Parameters 60
2.4 Physical properties of fluids 61
2.5 Limit condition parameters 61

5.1 Petrophysical Parameters 108
5.2 Physical properties of fluids 109
5.3 Boundary condition parameters 109
5.4 2D mesh parameters 110
5.5 CO_2 injection well conditions 110

6.1 Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and the long-range local approaches applied on the global test case 1 dataset. 'Regime 1' correspond to the case where the local domain extension is smaller than the well impact extension and 'regime 2' is the opposite. 'All' regime refers to both regime 1 and 2. . 128
6.2 Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and the short-range with interpolation local approaches applied on the global test case 1 dataset. 132
6.3 Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and long-range with different well locations local approaches. 135

6.4 Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and long-range with different well locations local approaches. 145
6.5 Reduction of Newton iterations in % for each case compared to the standard approach for the three local approaches applied on the global test case 1 dataset. 'Regime 1' correspond to the case where the local domain extension is smaller than the well impact extension and 'regime 2' is the opposite. 'All' regime refers to both regime 1 and 2. 150

List of Figures

1.1 Possible natural reservoirs for CO_2 storage from earth sciences society [11] 19
1.2 Schematic showing the two different regions of influence associated to CO_2 storage in saline aquifers. The footprint area of the carbon dioxide plume leaking in superficial aquifers is defined as the near-field. The footprint area of elevated pressure, updip formation water flow and brine (highly saline groundwater) leakage into shallow units across abandoned boreholes are defined as the far-field. From [30]. 20
1.3 Architecture of a perceptron or neuron. 24
1.4 An example of MLP architecture with $\ell = 5$ layers. It contains an input layer, 4 hidden layers and an output layer. 25
1.5 Example of initial and boundary points sampled on a domain using the Latin Hypercube Sampling strategy from [46] 28

1.6 Schematic of a physics-informed neural network (PINN) composed of 3 blocks. An approximation block composed of a neural network parameterized by theta. The approximation corrector block which computes the loss function through automatic differentiation. Finally, an minimization block that minimizes the loss with respect to the neural network parameters leads to an approximation $u(x, \theta*)$ of the solution of a PDE. 29
1.7 DeepONet architecture is composed of two networks, a Branch net and a Trunk net. The branch takes the evaluation of the function u at different points and returns p basis. The trunk takes any point y on the domain of $G(u)$ and returns p coefficients. Finally, the two outputs are multiplied and summed to reconstruct the desired evaluation of the operator G. 35
1.8 Architecture of a Fourier layer. The Fourier layer starts with an input vector v, applies the Fourier transform $\mathcal{F}$ to it, then performs a linear transformation R on the lower Fourier modes while filtering out the higher modes. The inverse Fourier transform $\mathcal{F}^{-1}$ is then applied. A local linear transformation W is applied to the original input vector v. The output of the top and bottom operations are then added together and an activation function is applied. 39
1.9 Architecture of the Fourier Neural Operator. Given an input a, lift to a higher dimension channel space by a neural network P, then apply T layers of integral operator i.e Fourier layers. Finally, project back to the desired dimension using a neural network Q and get the output u. 39

2.1 1D from SHPCO2 case geometry [2] 59
2.2 2D SHPCO2 case geometry from [2] 60

3.1 Example of well event impact on the pressure and saturation profile. The left up and down figures represent the pressure and saturation before a well opening, while the up and down right figures represent the pressure and saturation after a well opening. 83

3.2 Example of scenario and it's impact on Newton's method. The scenario is composed of a well event opened after 2 time steps (red dashed line) and closed at 12 time steps (red dotted line). 84

4.1 Right figure: input parameters D maps and source δ maps for different positions and their corresponding solution on the left figure. 89
4.2 Results of the different fitting process for each method. Left figure shows the evolution of relative $L2$ error through the fitting process i.e training and right figure is the scree plot of the POD. 90
4.3 In and out of distribution parity plots using L_2 norm for each method between POD (upper left), FNO (upper right), DeepONet (lower left) and MLP (lower right). The red dotted line corresponds to the identity. The x-axis corresponds to the finite difference solution norm while the y-axis corresponds to the prediction norm. 91
4.4 Example of saturation and pressure solutions at different times using a left boundary pressure of 19.5×10^6 Pa. 92
4.5 Evolution of relative L_2 error for training and validation through epochs. 93
4.6 Example of exact and predicted saturation and pressure at different times with a left boundary pressure of 14.1×10^6 Pa. 94
4.7 Parity plot of the predicted versus exact saturation (left figure) and pressure (right figure) using the L_2 norm for the three datasets. 94
4.8 Example of saturation and pressure snapshots obtained using the machine-learning model to iterate through time starting from a time $t = 0$ and with a left boundary pressure of 17.5×10^6 Pa. 95
4.9 Example of saturation and pressure snapshots obtained using the machine-learning model to iterate through time starting from a time $t = 0$ and with a left boundary pressure of 14.1×10^6 Pa. 96

4.10 Example of saturation and pressure snapshots obtained using the machine-learning model and saturation clipping to iterate through time starting from a time $t = 0$ and with a left boundary pressure of 14.1×10^6 Pa. The predictions are represented in red dots while the solutions are in blue dots. 97

5.1 Example of scenario and it's impact on Newton's method. The scenario is composed of a well event opened after 2 time steps (red dashed line) and closed at 12 time steps (red dotted line). 102
5.2 Selected neural network architecture composed by an uplifting dense layer, four Fourier layers, a dense layer and a projection dense layer. **Nc** and **Ni** are parameters respectively representing the number of channels and the number of inputs. 106
5.3 The Fourier layer starts with an input vector v, applies the Fourier transform $\mathcal{F}$ to it, then performs a linear transformation R on the lower Fourier modes while filtering out the higher modes. The inverse Fourier transform $\mathcal{F}^{-1}$ is then applied. A local linear transformation W is applied to the original input vector v. The output of the top and bottom operations are then added together and an activation function is applied. . . 107
5.4 Qualitative view of Neural network input feature possible assembly. 107
5.5 Adapted 2D SHPCO_2 case geometry 108
5.6 Test case 1 example of reservoir Pressure (left) and Saturation (right) obtained after a time-step with $S0 = 3.81 \times 10^{-4}$, $q_g = 7.61 \times 10^{-4} m^2/s$ and $dt = 2.4 \times 10^{+8} s$. 109
5.7 Test case 2 workflow with multiple well openings and closures. A step of time with a null well flow is realised between each closure and opening. 110
5.8 Example of reservoir Pressure (left) and Saturation (right) obtained after a time-step with $q_g = 9.8 \times 10^{-4}$ m^2/s and $dt = 3.1 \times 10^{+8} s$. The simulation required 12 Newton iterations to converge. 111
5.9 L2 loss evolution through epochs for test case 1. The lowest test loss value is 1.9×10^{-3} reached at epoch 17285. The corresponding train loss value is 2.0×10^{-3}. 112

5.10 L2 loss evolution through epochs for the test case 2. The lowest test loss value is 8.7×10^{-4} reached at epoch 7295. The corresponding train loss value is 9.3×10^{-4}. We start with a learning rate of $1. \times 10^{-4}$, at iteration number 1000, we change it to $5. \times 10^{-5}$, then $1. \times 10^{-5}$ at iteration number 1600 and finally we set the learning rate to $5. \times 10^{-7}$ at iteration number 3600 and until the end. 113
5.11 Hybrid Newton approach applied to the example scenario and it's impact on Newton's method. 114
5.12 Test case 1 scatter plot of the number of Newton iterations needed to converge using reference (classic) methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using reference (classic) and Hybrid methodologies for the train and test set respectively. 115
5.13 Test case 2 scatter plot of the number of Newton iterations needed to converge using reference (classic) methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using reference (classic) and Hybrid methodologies for the train and test set respectively. 116
5.14 Distribution of Newton iterations considering well event parameters q_g the well injection flow and dt the time-step in seconds. The black points represent the simulations where the hybrid methodology requires more Newton iterations than the standard methodology. 117

6.1 Example of saturation (left) and pressure (right) obtained through the numerical resolution using $|q_g| = 9.5e-4m^2/s$ and $dt = 1.6 \times 10^8 s$ (i.e ≈ 5 years). 123
6.2 Saturation (left) and residual (right) evolution through Newton iterations. The final solution is presented in the figure 6.1. 124
6.3 Local approach 1D test case distribution of Newton iterations considering well event parameters q_g the well injection flow rate and dt the time-step. 125
6.4 Distribution of well extension impacts in terms of number of cells. 125

6.5 Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 3.56×10^{-3} reached at epoch 967. The corresponding train loss value is 4.61×10^{-3}. 126
6.6 Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 5.29×10^{-3}. The corresponding train loss value is 6.88×10^{-3}. 127
6.7 Example of reconstructed initial saturation guess composed of two parts, local neural network prediction and initial saturation in the reservoir. The left figure highlights a case where the local domain does not reach the well extension impact while the right figure shows a case where the size of the local domain is superior or equal to the well extension impact. 127
6.8 Short-range (figure a) and long-range (figure b) local approach 1D test case scatter plot of the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using standard and Hybrid methodologies for the train and test set respectively. Finally the red vertical line shows the separation between the two main regimes. . . . 129
6.9 Reconstructed global saturation using the initial saturation (blue), the local predicted saturation (orange) and a linear interpolation (green). 130
6.10 Local approach 1D test case scatter plot of the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology with linear interpolation on the train set (left figure) and on the test set (right figure). . . 131
6.11 Comparison of global (left) and local (upper and lower mid-left) implicit pressure for different well locations and their resulting true global (mid-right) and local (upper and lower right) saturations obtained using the parameters: $q_g = 2 \times 10^{-4} m^2/s$ and $dt = 9.1 \times 10^7 s$. 132
6.12 Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 3.26×10^{-3}. The corresponding train loss value is 4.35×10^{-3}. 133

6.13 Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 2.72×10^{-3}. The corresponding train loss value is 4.27×10^{-3}. 134
6.14 Example of reconstructed initial saturation guess composed of two parts, local neural network prediction and initial saturation in the reservoir. The left figure highlights a case where the local domain does not reach the well extension impact while the right figure shows a case where the size of the local domain is superior or equal to the well extension impact. 135
6.15 Short-range (figure a) and long-range (figure b) local approach 1D with a variable well location test case scatter plot of the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using standard and Hybrid methodologies for the train and test set respectively. Finally the red vertical line shows the separation between the two main regimes. 136
6.16 2D SHPCO2 case geometry with three different local domains. Each local domain has a well at its center. 138
6.17 Example of global domain Ω_1 and local domain Ω_2. The cell function ϕ applied on a face σ from the boundary of Ω_2 is equal to ϕ applied to the cell k. 139
6.18 Qualitative view of local input features. 141
6.19 Relative L_2 loss error through 1500 epochs. The lowest test value is 2.5×10^{-3} reached at epoch 1368 and the corresponding train loss value is 2.5×10^{-3}. 142
6.20 Case 1 local saturation machine learning prediction example (upper left), global saturation guess (upper right), saturation solution (lower left) and saturation error between the global guess and the solution (lower right). 143
6.21 Case 1 local domain decomposition saturation solution example (upper left), global saturation guess (upper right), saturation solution (lower left) and saturation error between the global guess and the solution (lower right). 143
6.22 Residual evolution through Newton iterations. 144

6.23 Test case 1 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). 145
6.24 Test case 1 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using domain decomposition methodology on the train set (left figure) and on the test set (right figure). 146
6.25 Case 2 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). 147
6.26 Case 2 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using domain decomposition methodology on the train set (left figure) and on the test set (right figure). 148
6.27 Case 3 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). 149
6.28 Case 3 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using domain decomposition methodology on the train set (left figure) and on the test set (right figure). 149

7.1 Example of pressure obtained through the developed workflow. 156
7.2 Example of permeability field (K) at the layer 84 from the SPE10 comparative solution project. 157

Chapter 1

Introduction

Contents

1.1 Context and Motivations **18**

1.2 State of the art of Physics Informed Deep Learning **21**

1.2.1 Physics Informed Neural Network 23

1.2.2 Neural Operators 33

1.2.3 Physics Informed Deep Learning for CO2 storage 38

1.2.4 Hybrid approaches and hybrid Newton's method 41

1.3 Outline and results summary of the thesis **43**

1.1 Context and Motivations

The reduction of CO_2 emission into the atmosphere is mandatory to achieve ecological transition. CO_2 geological storage is an essential instrument for efficient Carbon Capture and Storage (CCS) policies. According to the International Agency Energy's 2017 report, we must sequester 940 Megatonnes of CO_2 ($MtCO_2$) through CCS by 2060 to limit global warming to 2 degrees [2]. In 2020, there were 21 Carbon Capture, Utilisation and Storage facilities with the capacity to store up to $40\,MtCO_2$ annually [3]. Among these 21 facilities, only 5 sites -Sleipner, Snøhvit, Quest, IllinoisBasin, and Gorgon- are dedicated to geological CO_2 storage, accounting for approximately $7MtCO_2/year$

[3]. The primary storage type for the other sites is for Enhanced Oil Recovery (EOR), which involves injecting CO_2 into an oil reservoir to increase oil production. This is one the most common applications of CO_2 utilisation. In the context of CCS, the underground geological storage of carbon dioxide (CO_2) in offshore deep saline aquifers is being explored to reduce greenhouse gases emissions. This method accounts for approximately 80% of the theoretical global capacity storage, which is estimated at $10,000$ gigatonnes of CO_2 according to the Sixth assessment report of the Intergovernmental Panel on Climate Change [1]. There are three main types of subsurface reservoirs considered for CO2 geological storage : depleted oil and gas fields, unminable coal beds and deep saline aquifers, as depicted on figure 1.1.

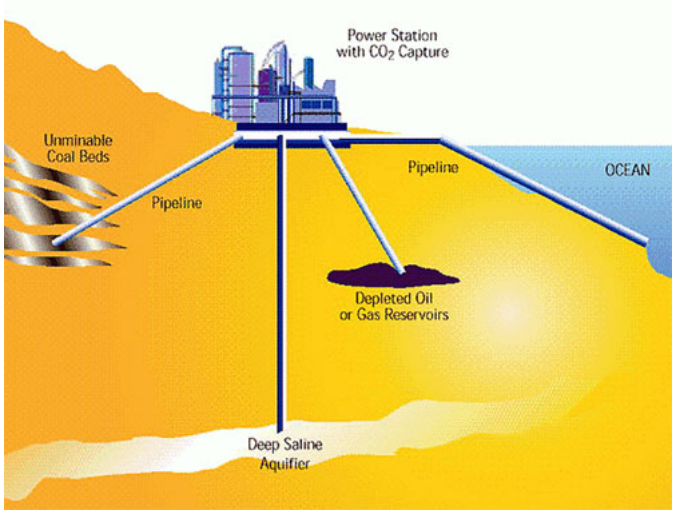


Figure 1.1: Possible natural reservoirs for CO_2 storage from earth sciences society [11]

Carbon dioxide injection in the underground has two main impacts: the near-field impact due to the upward vertical migration of free-phase CO_2 to superficial aquifers, and the far-field impact caused by large-scale displacement of formation waters by the injected CO_2 [30]. Potential leaks of free phase CO_2 exist along abandoned or active wells (see figure 1.2) and through the caprock when the entry pressure is reached and when fractures or faults are open for flow, naturally or because of excessive injection pressure build-up [30].

Numerical simulations provide the solution to the multi-phase flow equations

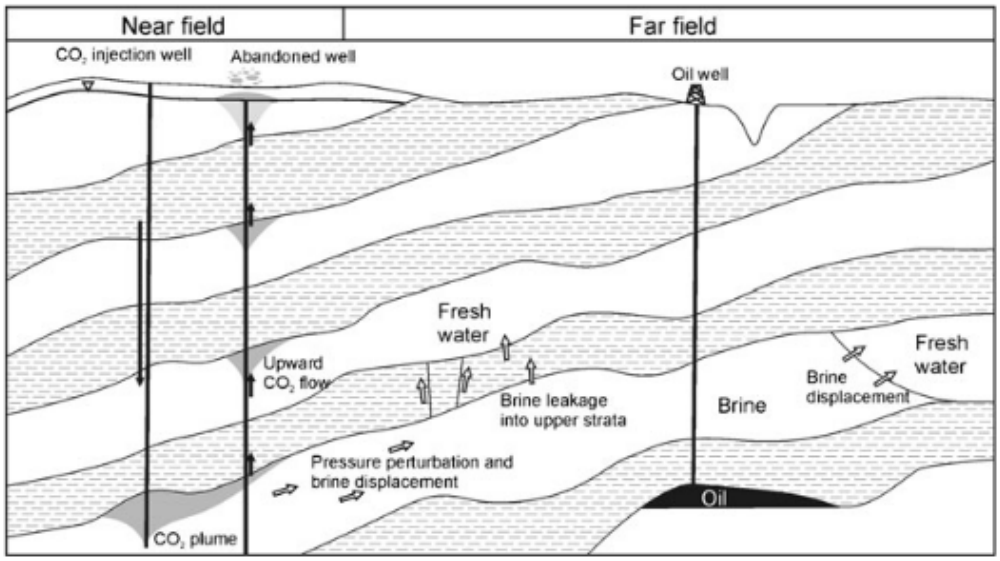


Figure 1.2: Schematic showing the two different regions of influence associated to CO_2 storage in saline aquifers. The footprint area of the carbon dioxide plume leaking in superficial aquifers is defined as the near-field. The footprint area of elevated pressure, updip formation water flow and brine (highly saline groundwater) leakage into shallow units across abandoned boreholes are defined as the far-field. From [30].

that model the behavior of the CO_2 injection site. They are an important tool to decide either or not to exploit a potential carbon storage site and to monitor the operations (long term storage of injected CO_2, potential gas leakage, optimal positioning of CO_2 injection wells, etc.).

However, numerical simulations of fluid flow in porous media are computationally demanding : it can take up to several hours on a HPC cluster in order to simulate one injection scenario for a large CO_2 reservoir if we want to accurately model the complex physical processes involved. This becomes a limiting issue when performing a large number of simulations, e.g. in the process of "history matching" : in order to fit the various model parameters to match the available historical data, a large number of simulations corresponding to various parameter sets needs to be performed.

More specifically, well events (opening and closure) cause important numerical difficulties due to their instant impact on the system. This often forces a

drastic reduction of the time step size to be able to solve the non-linear system of equations resulting from the discretization of the continuous mathematical model. However, these specific well events in a simulation are relatively similar across space and time : the degree of similarity between two well events depends on a few parameters such as the injection condition, the state of the reservoir at the time of the event, the boundary conditions or the porous media parameters (permeability and porosity) around each well.

Recent interest in machine learning applied to the prediction of physical processes has fueled the development of "Physics Informed Deep Learning", where machine learning models either replace or complement traditional numerical algorithms while preserving the inherent constraints from the physical model. These models can be trained in a supervised or unsupervised manner. In supervised learning, the objective is to match the labeled data available from experiment or previous simulations. In unsupervised learning, no labeled data is available and the objective is simply to directly enforce physical constraints, e.g. by minimizing the residual of the partial differential equations describing the evolution of the solution, by penalizing deviations to mass conservation, etc.

Therefore, the objective of this thesis is to adapt recent advances in physics informed deep learning in order to alleviate the impact of well events in the numerical simulation of multiphase flow in porous media.

1.2 State of the art of Physics Informed Deep Learning

Physics Informed Deep Learning (PIDL) is an emerging field of research that aims to combine the strengths of both physics-based modeling and deep learning to improve the accuracy and efficiency of prediction models. PIDL seeks to integrate domain-specific knowledge from physics or other sciences into deep learning models to enhance their performance and interpretability.

In the current literature, there is a lack of consensus regarding the terminology used to describe the integration of prior knowledge of a physical phenomenon with deep learning. Terms such as 'physics-informed,' 'physics-based,' 'physics-guided,' and 'theory-guided' are commonly employed. In

order to address this issue, [24] have devised a comprehensive taxonomy termed 'informed deep learning.' The taxonomy is structured around three fundamental conceptual stages: (i) the type of deep neural network utilized, (ii) the representation of physical knowledge, and (iii) the manner in which physical information is integrated.

In traditional deep learning, data-driven models are trained on large datasets to learn complex patterns and relationships in the data. However, these models may not be able to incorporate prior knowledge of physical laws or constraints, making them less suitable for tasks that require physical reasoning. In contrast, physics-based models are built on physical principles and laws, but they may lack the flexibility to capture complex and possibly unknown nonlinear relationships in the data.

PIDL seeks to address these limitations by integrating physics-based knowledge into deep learning models. This is typically achieved by incorporating physical constraints or equations into the loss function of a deep learning model. By doing so, the model can learn to satisfy both the data-driven constraints and the physical laws, leading to more accurate and physically meaningful predictions. There are three main approaches: i) add physical constraints, ii) design specific neural architectures for physical problems, iii) hybrid approaches mixing neural networks and traditional numerical methods for physical problems.

Regarding the approach i) add physical constraints, it is possible to add 'soft' or 'hard' constraints. An example of soft constraint is including the residual of a PDE in the objective function of a deep learning model while an example of hard constraint is enforcing the the initial and boundary conditions directly in neural network construction. We detail those two specific methodologies in the 1.2.1 section.

PIDL has applications in a wide range of fields, including fluid dynamics, materials science, robotics, and geosciences, among others. It has the potential to significantly improve the accuracy and efficiency of modeling tasks that require both physical reasoning and complex nonlinear relationships. We present a few examples and references from the flourishing literature.

- Fluid Dynamics: PIDL has been applied to the simulation of fluid dynamics problems, such as predicting fluid flow behavior around obstacles [69] or designing optimal flow control strategies [45].

- Materials Science: PIDL has been used to design new materials with desired properties. By integrating physical principles into deep learning models, PIDL helps predicting material properties such as strength, conductivity, and elasticity, which can be used to optimize material design and discovery [63].
- Robotics: PIDL has been applied to the control of robotic systems, such as predicting the motion of robotic arms or improving the stability of walking robots. By incorporating physical constraints such as energy conservation or force balance into deep learning models, PIDL can improve the performance of robotic systems while reducing the need for extensive training data [62] [23].
- Geosciences: PIDL has been used to predict and analyze geophysical phenomena such as earthquakes [37], climate change [16] [43], and ocean currents [12][20]. By integrating physical models into deep learning models, researchers can improve the accuracy of predictions and gain new insights into complex systems.

The straightforward and popular theoretical application of PIDL are Physics Informed Neural Networks.

1.2.1 Physics Informed Neural Network

1.2.1.1 Neural Network Reminder: Multi-layer perceptron

A *perceptron*, also referred to as a *neuron* [48] serves as the foundational and simplest model within the realm of deep learning. Figure 1.3 depicts an illustration of a perceptron's architecture. In this context, $x \in \mathbb{R}^d$ is the input vector, the vector $w = (w_1, \ldots, w_d)$ symbolizes the weights, b represents the bias and σ stands for a nonlinear (more precisely a non polynomial, see Theorem 1 below) and monotonous function known as the *activation function*.

Activation functions play a pivotal role in neural networks and continue to stand out as one of the most crucial components responsible for the remarkable success of Deep Neural Networks (DNNs) [10]. They serve the vital function of introducing nonlinearity into the model, thus enabling the network to grasp intricate patterns and relationships in the data. In the absence of activation functions, a neural network would merely perform linear trans-

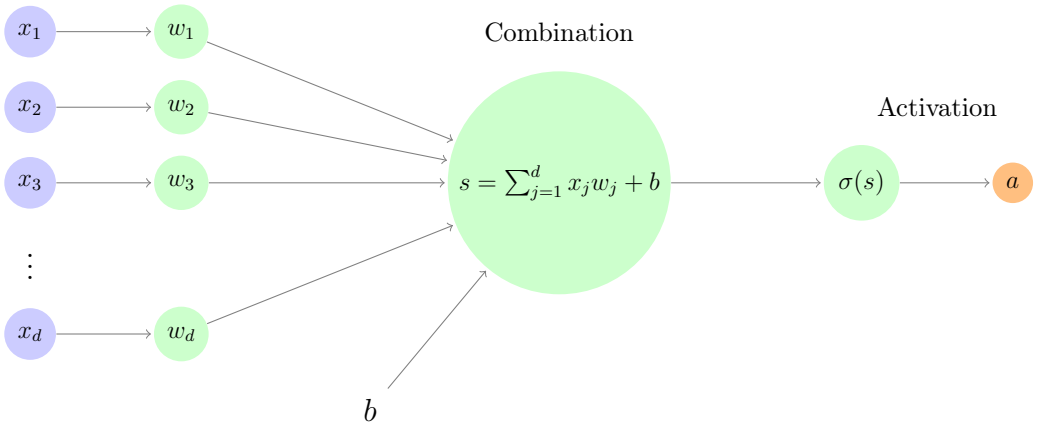


Figure 1.3: Architecture of a perceptron or neuron.

formations, severely restricting its ability to learn.

Each activation function possesses unique characteristics, making some more suitable for particular types of problems than others. Consequently, the choice of an activation function hinges on the specific problem at hand. For instance, the *sigmoid* function [41] is common for binary classification tasks since it maps neuron outputs to probabilities ranging from 0 to 1, interpretable as the probability of inputs belonging to specific classes. The *hyperbolic tangent function* (tanh) operates similarly to the sigmoid but maps the output to a range between -1 and 1. In contrast, the *Rectified Linear Unit* (ReLU) [14], defined by $ReLu(x) = max(0, x)$, has gained popularity in recent years due to its simplicity and effectiveness.

A *feed-forward neural network* (FNN) represents a category of artificial neural networks structured with multiple layers where data propagation occurs in a unidirectional manner, strictly from the input layer to the output layer, with no cycles nor loops within the network. A *multi-layer perceptron* (MLP), also known as a *fully connected network* (FCN), falls within the FNN category. It comprises a series of layers of interconnected neurons, as depicted in Figure 1.4. Notably, with the exception of the neurons in the input layer, each neuron in the network employs an activation function.

The neurons within a hidden layer take as input the outputs of neurons from the preceding layer and transmit their own outputs to the neurons in the subsequent layer by means of synaptic weights.

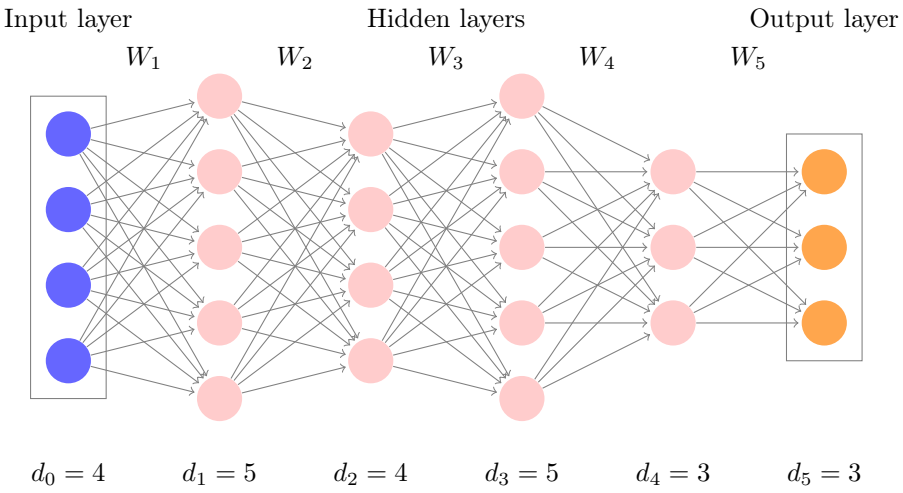


Figure 1.4: An example of MLP architecture with $\ell = 5$ layers. It contains an input layer, 4 hidden layers and an output layer.

Likewise, the neurons in the output layer receive as input the outputs originating from the final hidden layer. Formally, in the context of an ℓ-layer MLP denoted as f_θ parametrized by θ, the output:

$$z := a_\ell = f_\theta(x) \in \mathbb{R}^{d_\ell} \tag{1.1}$$

from the input $x \in \mathbb{R}^{d_0}$ is computed as

$$a_0 := x, \qquad s_i = W_i \bar{a}_{i-1}, \qquad a_i = \sigma_i(s_i), \qquad \text{for } i \text{ from 1 to } \ell, \tag{1.2}$$

with $W_i \in \mathbb{R}^{d_i \times (d_{i-1}+1)}$ corresponds to the weight matrix associated with layer i. s_i is termed to as pre-activation of the layer. The augmented activation vector, denoted as $\bar{a}_{i-1} = (1, a_i^T)^T$ where the value 1 is used for the bias, and where σ_i is the activation function at layer i. The number of neurons in layer i is d_i and the overall count of parameters is denoted as $p = \sum_{i=1}^{\ell} d_i(d_{i-1}+1)$. The MLP is parameterised by θ, encompassing all of its weights,

$$\theta = [\text{vec}(W_1)^T, \text{vec}(W_2)^T, \ldots, \text{vec}(W_\ell)^T]^T \in \mathbb{R}^p,$$

with "vec" the operator that transforms a matrix into a vector.

Despite their straightforward architectures, the MLP has demonstrated remarkable effectiveness in modeling tabular data and is often used as reference in specific tasks such as the learning of physical solutions.

1.2.1.2 Neural Network Reminder: Training

Training a neural network consists in fitting its learnable parameters so as it can accurately map some input data to a desired output or prediction. It is therefore an optimization process. This training can be unsupervised or supervised. In unsupervised learning, a dataset of only inputs is provided and the objective is to discover hidden patterns or structures in the data. Therefore, the training algorithm has no access to predefined labels. Examples of unsupervised learning are k-means clustering, principal component analysis and Physics Informed Neural Networks. We detail the Physics Informed Neural Network approach in the next section. In supervised learning, a dataset of inputs and their corresponding outputs or labels is provided. It is to note that it is possible to mix supervised and unsupervised approaches leading to semi-supervised learning.

In practice, the supervised or unsupervised training of neural networks, such as a ℓ-layer MLP denoted as f_θ and parametrized by θ, are realised through the following steps. First, the neural architecture is initialized i.e input layer, hidden layers and an output layer. Their respective weights and biases θ are also initialized. Then, given some input data $x \in \mathbb{R}^{d_0}$, a Forward pass is realised, as indicated in (1.2), mapping x to the output $z = f_\theta(x)$. This Forward pass is realised for each data in a dataset or collection of data $\{(|x_0|x_1|\cdots|x_N|), x \in \mathbb{R}_{d_0}\}$ with N the number of data.

Then, we evaluate the misfit of the prediction through a loss or cost function. In supervised learning for regression problems, the mean squared error (MSE) is a popular choice as loss function:

$$MSE = \frac{1}{N}\sum_{i=0}^{N} |f_\theta(x_i) - f(x_i)|^2 \tag{1.3}$$

with $f(x_i)$ the true label. For physical problems, it is more common to use the relative mean squared error or relative L_2 error with a small regularization

term ϵ:

$$MSE = \frac{1}{N}\sum_{i=0}^{N}\frac{|f_\theta(x_i) - f(x_i)|^2}{|f(x_i)|^2 + \epsilon} \tag{1.4}$$

Moreover, in unsupervised learning for physical problems, an example of loss function is based on the residual of a differential equation

Then, we wish to minimize the loss by updating the parameter θ. The most popular optimization algorithms are based on gradient descent which update the parameters by propagating the error backward through the network (i.e back-propagation). In practice, for each layer, we calculate the gradient of the loss L w.r.t the parameter θ using the chain rule. Then, we iteratively repeat the forward pass, the Loss calculation and the parameter update through back-propagation until the loss converges to a satisfactory level.

For differential equation problems, an idiomatic approach using neural networks is the 'Physics Informed Neural Network'.

1.2.1.3 Physics Informed Neural Network

The main idea behind Physics Informed Neural Network (PINN) is to create an artificial neural network for solving differential equations by minimizing a loss function representing the equation residual. This can be seen as unsupervised or semi-supervised learning with a soft constraint. Given the residual of a general differential equation G:

$$G(\vec{x}, \Psi(\vec{x})) = 0, \quad \vec{x} \in D \tag{1.5}$$

and subject to boundary conditions, where $D \subset \mathbb{R}^n$ is the domain of definition and $\Psi(\vec{x})$ is the solution to be computed.

For example, given a partial differential equation (PDE) such as Burgers' equation:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2},$$

with u the velocity and ν the viscosity. In this case, the PINN solution of Burgers' equation is obtained by minimizing a loss function corresponding to the equation residual (i.e G):

$$G(\vec{x}, u(\vec{x})) = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2},$$

with $\vec{x}$ a vector containing x and t.

To solve equation (1.5), we use a Neural Network (NN) to predict $\hat{\Psi}(\vec{x}, \theta)$ with θ the adjustable parameters of the NN (i.e weights and biases). The loss function used to fit $\hat{\Psi}(\vec{x}, \theta)$ to $\Psi(\vec{x})$ is the L_2 norm of G. Hence, $\hat{\Psi}(\vec{x}, \theta)$ used together with the loss function is a *physics informed neural network*. The quadrature points to approximate the loss function are generated by sampling inside and at the border of the domain (see figure 1.5).

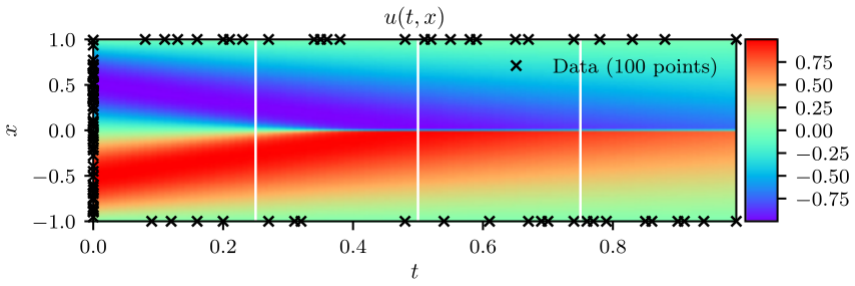


Figure 1.5: Example of initial and boundary points sampled on a domain using the Latin Hypercube Sampling strategy from [46]

Generally speaking, a PINN has three building blocks, an approximation block which is a neural network usually, an approximation corrector block i.e a loss function and a minimization block which minimizes the loss function.

Each of these blocks brings an error, an approximation error from the approximation block which depends on the neural network architecture, an estimation error brought by the approximation corrector block and the optimization error from the minimization block. Regarding the approximation error, there are results such as the 'Universal approximation theorem' showing that a single layer neural network with a sufficiently large width can uniformly approximate a function and it's derivative [18][31] [44]. Then the

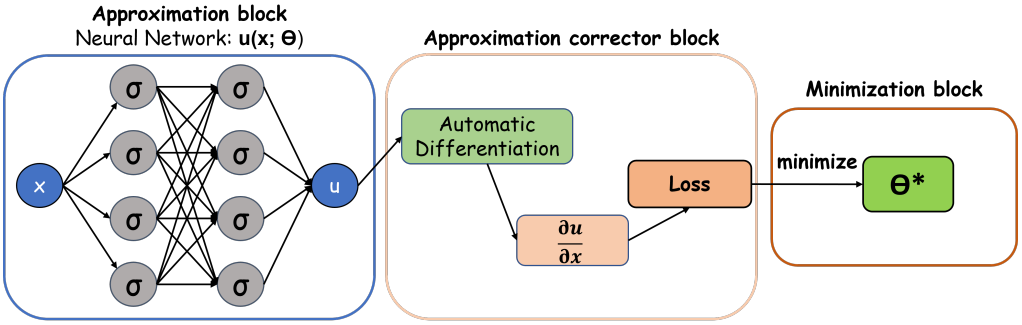


Figure 1.6: Schematic of a physics-informed neural network (PINN) composed of 3 blocks. An approximation block composed of a neural network parameterized by theta. The approximation corrector block which computes the loss function through automatic differentiation. Finally, an minimization block that minimizes the loss with respect to the neural network parameters leads to an approximation $u(x, \theta*)$ of the solution of a PDE.

estimation error, comes from the use of finite data. Finally, the optimization is the harder to tackle as the objective function is highly non convex. The optimization is often realised using Gradient-based optimization methods or quasi-Newton methods. In particular, stochastic gradient descent (SGD) and all its variants [49] such as Adam [25] are popular for PINN optimization. Additionally, L-BFGS [34] is a popular quasi-Newton method for PINN optimization.

To construct a PINN, there are two seminal approaches, the first one from Lagaris et al. [28] and the second one from Raissi et al. [46]. Their main differences are in the approximation and estimation block.

Overview of Lagaris' method

In 1997, Lagaris et al. [28] proposed to construct the trial solution $\hat{\Psi}(\vec{x}, \theta)$ such that it satisfies the boundary conditions. Therefore, this can be seen as unsupervised or semi-supervised learning with hard constraints.

$$\hat{\Psi}(\vec{x}, \theta) = A(\vec{x}) + F(\vec{x}, N(\vec{x}, \theta)) \tag{1.6}$$

where $N(\vec{x}, \theta)$ is a single output feedforward neural network with adjustable parameters θ and fed with the input vector $\vec{x}$, $A(\vec{x})$ satisfies the boundary

conditions and F is constructed so as not to contribute to the boundary conditions. It is to note that the notation $\vec{x}$ may encompass multiple variables such as space and time (i.e can represent $\vec{x} = (x, y, t)$) and therefore $A(\vec{x})$ satisfies the boundary and initial conditions. According to these definitions, $\hat{\Psi}(\vec{x}, \theta)$ also satisfies the boundary conditions.

The adjustable parameters θ can be learned by minimizing the mean square error loss:

$$MSE = \frac{1}{N_G} \sum_{i=1}^{N_G} |G(x_i^{\hat{D}}, \hat{\Psi}(x_i^{\hat{D}}, \theta))|^2 \tag{1.7}$$

with $(x_i^{\hat{D}})_{1 \leq i \leq N_G}$ the sampled points on the domain and G the residual function.

Overview of Raissi's method

Raissi et al. [46] propose a different approach. Instead of creating a trial function that satisfies the boundary conditions by construction such as Lagaris proposes, Raissi includes the boundary conditions inside the loss function.

If $\hat{\Psi}(\vec{x}, \theta)$ denotes the solution with adjustable parameters θ of problem (1.5), G the residual, then the parameters θ can be learned by minimizing the mean squared error loss

$$MSE = \frac{1}{N_G} \sum_{i=1}^{N_G} |G(x_i^{\hat{D}}, \hat{\Psi}(x_i^{\hat{D}}, \theta))|^2 + \frac{1}{N_u} \sum_{i=1}^{N_u} |\hat{\Psi}(x_i^{\hat{S}}, \theta) - \Psi(x_i^{\hat{S}})|^2 \tag{1.8}$$

here $(x_i^{\hat{S}})_{1 \leq i \leq N_u}$ denote the sampled points where Dirichlet boundary conditions are imposed.

The nature of PINN can be understood as unsupervised learning when it relies exclusively on physical equations and boundary conditions to solve forward problems.

1.2.1.4 Limitations of Physics Informed Neural Networks

Physics-Informed Neural Networks (PINN) strive to integrate domain-specific physical knowledge as flexible constraints within an empirical loss function. These constraints are subsequently optimized using established machine learning training techniques. Nevertheless, it is crucial to acknowledge certain intricate aspects and potential pitfalls.

Optimization

The training of a Physics Informed Neural Network can be viewed as an optimization problem where we try to minimize a loss function L. This last can be formulated as

$$\min_{\theta} L(\vec{x}, \hat{\Psi}(\vec{x}, \theta)) \tag{1.9}$$

such that

$$G(\vec{x}, \hat{\Psi}(\vec{x}, \theta)) \simeq 0, \quad \vec{x} \in D\,.$$

More generally, the loss function contains multiple terms corresponding to the residual of the differential equation, the initial conditions and the boundary conditions:

$$L = L_{ic} + L_{bc} + \lambda L_G\,. \tag{1.10}$$

Where L_{ic} and L_{bc} measure the misfit of the neural network prediction with respect to the initial and boundary conditions and L_G measures the misfit of the neural network prediction with respect to the residual of the differential equation. Here, λ is a hyperparameter that manages the balance between the initial and boundary conditions and the residual.

This loss function is most of the time highly non-convex. Therefore, it is generally admitted that the optimization process aims to converge at a critical point, but this critical point has no guarantee to be a global minimum. Moreover, the optimization process often involves engineering tricks and fine-tuning of parameters by trial and error. Those problems are common in most of deep learning fields, however PINN's have shown some special issues in the optimization process. The analyze of PINN models on simple

physical models show that the balance between the residual contribution and the initial/boundary conditions contributions in the optimization process is not straightforward to manage. Indeed, [61] shows that, during training using gradient descent, the back-propagated gradients of the residual and initial/boundary conditions show unstable imbalance in magnitudes and proposed a solution using an adaptive balance coefficient λ.

Generalization

The generalization can be first seen through its error which is the sum of the estimation error and the approximation error. References [40] and [39] provide upper bounds on this error for PINN's under certain conditions. Basically, the training error should be low, i.e the optimization process has been correctly done. Then, the number of collocations points must be sufficiently large and the solution of the underlying PDE is stable with respect to perturbations of inputs. For nonlinear PDEs, this might imply that the PDE solution is sufficiently regular. Moreover, PINN may have issues learning solutions of specific PDE's. Indeed, [27] shows that PINNs struggle to capture essential physics in PDEs with large coefficients in opposite with small coefficients where PINNs show satisfying performances. Moreover, [13] shows that given a specific non linear hyperbolic partial differential equation (PDE) and considering different choices for the flux function (concave, convex, non-convex), it is observed that PINN models face difficulties in accurately learning non-convex flux functions. This holds true even in cases where the solution can be adequately represented by a piece-wise continuous function. Finally, for PDE problems with low-regularity solutions, defining the PINN formulation requires a battle-hardened expertise [5].

Utilization

Suppose that one succeeds at training a Physics Informed Neural Network model. The obtained model will be useful as a relatively cheap surrogate model to solve the exact same problem with the same initial and boundary conditions. Therefore, in its original formulation, the model can only be used on one specific case. Changing the initial and/or boundary conditions requires to train a new model. There is a lack of generalization in the methodology itself.

Instead of learning one specific solution given a PDE with boundary/initial conditions, a more general idea is to learn operators that map functions to

functions and therefore handle a whole family of solutions corresponding to various boundary/initial conditions.

1.2.2 Neural Operators

Neural operators are a class of deep learning models that aim to learn and approximate mathematical operators directly from data. These operators can be differential operators, integral operators, or other types of mathematical operations encountered in various fields of science and engineering. The concept of neural operators originates from the field of scientific computing, where traditional numerical methods are commonly used to solve complex mathematical problems involving differential equations. However, these traditional methods often require explicit knowledge of the underlying equations and may become computationally expensive for high-dimensional problems or complex domains. Neural operators provide an alternative approach by leveraging the representational power of deep neural networks to approximate the operators directly from input-output data pairs. The underlying idea is to train a neural network to learn the mapping between input data and the desired operator output, without explicitly specifying the mathematical equations or operations involved. The development of neural operators has been greatly influenced by the success of deep learning in various domains, particularly in computer vision and natural language processing. By extending deep learning techniques to mathematical operators, researchers have sought to apply neural networks to problems in scientific computing, physics, and other areas where the accurate approximation of operators is crucial.

Recall that the universal approximation theorem states that a single hidden layer neural network can be used to approximate any continuous function to arbitrary accuracy [31]. It has been extended to continuous operators. The universal approximation theorem of operators [8] states that a shallow neural network can approximate accurately any nonlinear continuous functional or operator. This last result shows the interest in neural networks to learn nonlinear operators. However, in practice, deep neural networks and specific architectures outperform shallow neural networks for many tasks such as the classification of images.

The architecture and design of neural operators can vary depending on the specific problem and type of operator being approximated. Techniques

such as convolutional neural networks (CNNs) [29], recurrent neural networks (RNNs) [22][50], attention mechanisms [57], and graph neural networks (GNNs) [51] have been adapted and tailored to capture the inherent structure and properties of the operators being learned.

Overall, neural operators provide a promising avenue for enhancing computational methods in scientific and engineering applications by enabling the automatic learning of complex mathematical operators directly from data, leading to improved accuracy, efficiency, and generalization capabilities.

We present in the next sections, two popular types of neural networks that perform well at learning operators: DeepONet and Fourier Neural Operator.

1.2.2.1 Deep Operator Network (DeepONet)

DeepONet idea is based on an extension of the first operator network proposed by Chen and Chen [8], where they considered only shallow neural network. In the most used way, DeepONet is composed of two neural networks: a branch net and a trunk net. In [8], the following theorem is proved:

Theorem 1 (Universal approximation theorem for operators [8])**.** *Suppose that σ is a continuous non polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$, are two compact sets, V is a compact set in $C(K_1)$, G is a (linear or non-linear) continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integer n, p, m constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, ..., n$, $k = 1, ..., p$, $j = 1, ..., m$ such that,*

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk} \right| < \epsilon \qquad (1.11)$$

holds for all $u \in V$ and $y \in K_2$

This theorem shows that it is possible to learn non-linear operators through a neural network trained in a supervised manner and is at the basis of the DeepONet architecture [36]. In practice, this architecture has two inputs, the u function evaluated at m distinct points $[u(x_1), u(x_2), ..., u(x_m)]$ and y

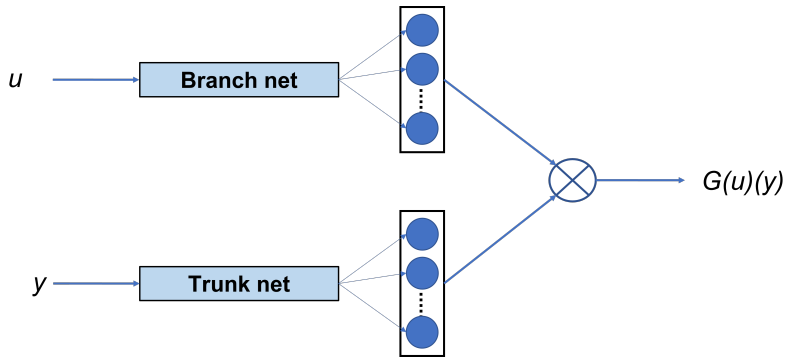


Figure 1.7: DeepONet architecture is composed of two networks, a Branch net and a Trunk net. The branch takes the evaluation of the function u at different points and returns p basis. The trunk takes any point y on the domain of $G(u)$ and returns p coefficients. Finally, the two outputs are multiplied and summed to reconstruct the desired evaluation of the operator G.

any point on the domain of $G(u)$. The output of a DeepONet is a linear combination of p basis functions. The basis functions b_k are the output of the branch net while the coefficients t_k of the linear combination are given by the trunk net:

$$G(u)(y) \approx \sum_{k=1}^{p} b_k t_k.$$

It is to note that the branch and trunk can be any neural network architecture.

As a practical example, we detail what would be DeepONet input and outputs for an elliptic equation:

$$\begin{cases} -\nabla \cdot (D(x)\,\nabla G) = f, & x \in (0,1), \\ G(0) = a, & G(1) = b. \end{cases} \tag{1.12}$$

We want to use DeepONet to represent the parameter-to-solution operator $(D, f) \mapsto G$, with D a function within a certain distribution and f the right-hand side. Using the terminology of [36], the domain is discretized

	NNs	DeepONets	Interpolation	Neural Operators
Discretization Invariance	✗	✗	✓	✓
Is the output a function?	✗	✓	✓	✓
Can the output be evaluated at any point?	✗	✓	✓	✓
Input at any point	✗	✗	✓	✓
Universal approximation	✗	✓	✗	✓

Table 1.1: Comparison of different properties of deep learning models taken from [26].

in m distinct sensor points $[x_1, x_2, \cdots, x_m]$ for D and in m' sensor points $[x'_1, x'_2, \cdots, x'_m]$ for f. The input of the branch net is the function D evaluated at each sensor point $[D(x_1), D(x_2), \cdots, D(x_m)]$ and the function f is evaluated at each sensor point $[f(x'_1), f(x'_2), \cdots, f(x'_{m'})]$. Then the input y of the trunk net is a new point to evaluate which does not necessarily coincide with the sensor points. Therefore, the output of the DeepONet is $G(D, f)(y)$.

1.2.2.2 Fourier Neural Operator

Neural operators are especially useful to learn solution operators associated with PDEs. In their paper "Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs" [26], Kovachki et al. propose a new definition of neural operators as a class of models that guarantee both discretization-invariance and universal approximation. A Discretization-invariant model can: handle any set of points in the input domain, be estimated at any point of the output domain and, as the discretization is refined, converge to a continuum operator . In opposition with other models such as DeepONet where modifying grid points makes the model no longer applicable. The main differences are sum up in the table 1.1.

We present the neural operator framework. The main parts consist in a lifting operation of the inputs to a higher-dimensional space, apply multiple Kernel Operators which can be seen as hidden layers, and finally project back to the output function. In a more formal way, let A and U be spaces of functions. Given input functions $a \in A = A(D; \mathbb{R}^{d_a})$ with values in $\mathbb{R}^{d_a}$ defined on a bounded domain $D \subset \mathbb{R}^d$, and output functions $u \in U = U(D'; \mathbb{R}^{d_u})$ with values in $\mathbb{R}^{d_u}$ and defined on the bounded domain $D' \subset \mathbb{R}^{d'}$. Neural operator

$G_\theta : A \mapsto U$ has the following structure:

1. **Lifting**: map the input $\{a : D \to \mathbb{R}^{d_a}\} \mapsto \{v_0 : D \to \mathbb{R}^{d_{v_0}}\}$ with $d_{v_0} > d_a$ to the first hidden representation.
2. **Iterative Kernel Integration**: map each hidden representation to the next $\{v_t : D_t \to \mathbb{R}^{d_{v_t}}\} \mapsto \{v_{t+1} : D_{t+1} \to \mathbb{R}^{d_{v_{t+1}}}\}$ for $t \in [0, 1, .., T-1]$ (with t the layer index) by combining a local linear operator, a linear integral kernel operator, and a bias function, and then applying a fixed, point-wise non linearity i.e an activation function. It is to note that $D_0 = D$ and $D_T = D'$.
3. **Projection**: map the output of the last layer $\{v_T : D \to \mathbb{R}^{d_{v_T}}\} \mapsto \{u : D' \to \mathbb{R}^{d_u}\}$ with $d_{v_T} > d_u$ to the output function.

This can be written as:

$$G_\theta := Q \circ \sigma_T(W_{T-1} + \mathcal{K}_{T-1}) \circ \cdots \circ \sigma_1(W_0 + \mathcal{K}_0) \circ P\,, \tag{1.13}$$

where $P : \mathbb{R}^{d_a} \mapsto \mathbb{R}^{d_{v_0}}$ is the lifting mapping, $Q : \mathbb{R}^{d_{v_T}} \mapsto \mathbb{R}^{d_u}$ is the projection mapping, $W_t : \mathbb{R}^{d_{v_{t+1}} \times d_{v_t}}$ are local linear operators and $\mathcal{K}_t : \{v_t : D_t \mapsto \mathbb{R}^{d_{v_t}}\} \mapsto \{v_{t+1} \mapsto \mathbb{R}^{d_{v_{t+1}}}\}$ are the linear kernel integral operators. Finally, σ is a non-linear activation function. The kernel integral operator mapping can be defined by:

$$\left(\mathcal{K}_\theta v_t\right)(x) := \int_D \kappa_\theta(x, y) v_t(y) dy, \forall x \in D \tag{1.14}$$

where κ_θ is a neural network parameterized by θ and represents a kernel function which is to learn from data. Therefore, the overall structure mimics the one of a finite dimensional neural network.

When it comes to the numerical computation of the kernel integral operator, the Monte Carlo approximation of (1.14) on a J-points discretization of D requires $\mathcal{O}(J^2)$ matrix-vector multiplications. Therefore, it is not fit for use in case of a supervised learning as it will be too computationally intense. Multiple solutions to reduce the cost have been proposed and are exposed in [26] but the solution that showed the most promising performances is the Fourier Neural Operator (FNO) [33].

The main idea behind FNO is to substitute the kernel integral operator in equation (1.14) with a convolution operator that is defined within Fourier space. This enables the use of the Fast Fourier Transform (which requires an uniform discretization) to efficiently compute (1.14) in quasi-linear complexity. To do so, $\kappa_\theta(x, y) = \kappa_\theta(x - y)$ is enforced, making (1.14) a convolution operator.

The Fourier integral operator $\mathcal{K}$ is then defined as:

$$\left(\mathcal{K}_\theta v_t\right)(x) = \mathcal{F}^{-1}\left(R_\theta \cdot \left(\mathcal{F} v_t\right)\right)(x) \quad \forall x \in D \tag{1.15}$$

with $\mathcal{F}$ the Fourier transform, $\mathcal{F}^{-1}$ the inverse Fourier transform and $R_\theta = \mathcal{F}(\kappa_\theta)$. This notation assumes that κ is periodic in order for it to admit a Fourier series expansion. Moreover, we can select a finite-dimensional parameterization by truncating the Fourier series at a maximal number of modes $k_{max} = |\{k \in \mathbb{Z}^d : |k_j| \leq k_{max,j}, j = 1, \cdots, d\}|$. This truncation can be interpreted as a low-pass filter of Fourier high-frequency modes induced by the non-linearity of the activation functions. Therefore, R_θ is the complex-valued tensor of parameters to learn.

This defines a Fourier layer which is exposed on figure 1.8. Finally, we can define the whole Fourier Neural Operator as presented in figure 1.9 which consists in a lifting layer, which can be any transformation including a neural network, multiple Fourier layers corresponding to the iterative kernel integrator and at the end a projection layer usually chosen as a neural network.

1.2.3 Physics Informed Deep Learning for CO2 storage

Now that we have presented the main neural architectures used in Physics Informed Deep Learning, let's delve into their specific applications to geological CO2 storage.

Physics Informed Deep Learning has been applied to geological sequestration of CO_2 using a wide range of machine learning techniques. One important aspect to assess when injecting CO_2 in the underground is the potential for leakage. Therefore, numerous numerical simulations are required to validate an injection scenario. These simulations are costly, and machine learning has

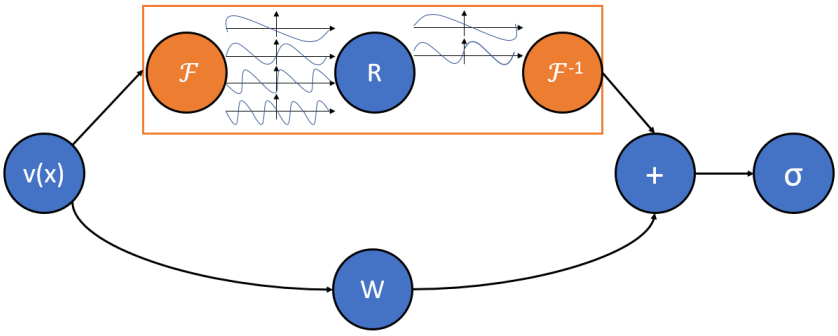


Figure 1.8: Architecture of a Fourier layer. The Fourier layer starts with an input vector v, applies the Fourier transform $\mathcal{F}$ to it, then performs a linear transformation R on the lower Fourier modes while filtering out the higher modes. The inverse Fourier transform $\mathcal{F}^{-1}$ is then applied. A local linear transformation W is applied to the original input vector v. The output of the top and bottom operations are then added together and an activation function is applied.

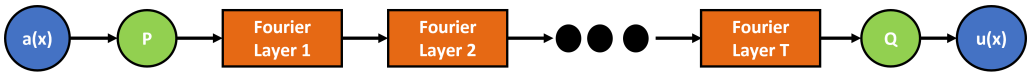


Figure 1.9: Architecture of the Fourier Neural Operator. Given an input a, lift to a higher dimension channel space by a neural network P, then apply T layers of integral operator i.e Fourier layers. Finally, project back to the desired dimension using a neural network Q and get the output u.

the potential to accelerate them, facilitating decision-making and enabling a faster and scalable deployment of CCS. For example, in [9], simulations were accelerated by a factor of 8000 compared to the traditional numerical solvers, while maintaining reasonable accuracy. This was achieved using a novel architecture based on FNO. In [65], yet another innovative architecture, also based on the FNO achieves a remarkable speedup compared to existing methods in the numerical simulation of CO_2 storage. Several other notable articles also demonstrate a substantial reduction in the computational cost of CO_2 storage numerical simulations [67] [66] [54] [21] [71] [52] [68] [64]. These performances are essential to evaluate plume migration and consequently, predicting potential leakage. In [55], a deep-learning based surrogate model is developed to manage operations for coupled flow and geomechanic simulations in a deep saline aquifer. The surrogate's goal is to reduce uncertainty aquifer's pressure buildup. Furthermore, since the mineralization of the CO_2 is an essential trapping mechanism and modeling it involves coupled reactive transport models that are computationally expensive, machine learning can offer cost-effective and accurate surrogate models, as shown in [56]. Additionally, machine learning models serve as a predictive tools for assessing the trapping performances of CO_2, as demonstrated in [59].

Another crucial aspect is reservoir monitoring over time. Effective monitoring ensures the safe and permanent storage of CO_2. Geophysical surveys serve as important tools for achieving high-resolution reservoir monitoring through historical data. However, the inversion of this data can be costly. Machine learning techniques have the potential to reduce the cost of this process, as demonstrated in [35]. Moreover, there are multi-objectives monitoring approaches such as those in [53], where reinforcement learning is used to optimally manage CO_2 storage in deep saline aquifer while extracting brine, or in [72], where machine learning is employed to optimize CO_2 storage in an oil reservoir while maximizing the oil production through enhanced oil recovery. Additionally, in [7], a machine-learning-based approach is proposed that accounts for uncertainties in reservoir monitoring.

As injection scenarios span over long periods, lasting at least 30 years for injection and potentially thousands of years for sequestration, assessing well degradation rates becomes essential. Machine learning approaches have shown promising results in addressing this issue, as demonstrated in [32] [60].

1.2.4 Hybrid approaches and hybrid Newton's method

In most of the cases, the resulting machine learning model is used as a surrogate model that supplant the traditional solver. This allows for faster evaluation of new input parameters in contrast to the original numerical solver. This becomes especially valuable when the numerical solver is computationally expensive and time-consuming, as surrogate models can provide quick approximations. **However, this 'black-box' approach lacks a guarantee of predicting solutions accurately, in contrast to the reliability of a numerical solver.** Indeed, these machine-learning models may induce errors even greater as the new data to evaluate are far from the training data, but we can not assess these errors without employing the numerical solver. Consequently, hybrid approaches that complement the numerical solver with a machine learning model while preserving the numerical guarantees are worth considering for evaluation. Before delving into the hybrid Newton method, which is the primary focus of this thesis, we wish to highlight some other hybrid methods.

Numerical schemes such as the theta-scheme or the backward difference formulas [17], necessitate the assignment of coefficients typically set to constant values based on the specific problem at hand. However, there is no guarantee that utilizing these constant coefficients will result in the optimal solution, i.e the one with the least error. In [38], machine-learning models are used to predict the optimal coefficients of numerical schemes. Furthermore, machine-learning models may enhance the discretization of dynamical systems [15], facilitating the handling of the dynamics at various scales [4].

When dealing with problems involving conservation laws, discontinuities may arise, necessitating specific attention to avoid numerical issues. The management of these discontinuities typically involves two steps: i) the identification of 'troubled-cells', i.e cells in the mesh impacted by the discontinuities, and ii) the application of a correction to these troubled-cells. This two-step method can be numerically costly. Machine-learning models used as troubled-cells detection functions have demonstrated significant performance improvements [47] [58].

1.2.4.1 Hybrid Newton's method

Solving non-linear system of equations resulting from the discretization of PDEs often requires a linearization method. In practice, Newton's method is often used and requires at each iteration to solve a linear system. Moreover, Newton's method shows a quadratic rate of convergence when the iterates are in the neighbourhood of the solution. Therefore, Newton's methods is very sensitive to initialization. Indeed, an initial guess that is far from the solution may require a lot of iterations to reach convergence. This leads to an interesting hybrid approach which is the hybrid Newton's method [42] [19] [70] [6]. It consists in predicting an initialization of Newton's method directly in the zone of quadratic convergence. The hybrid initialization is obtained through the prediction of a machine learning model. In [19], machine learning models are used to predict initialization of either Newton's method for solving non-linear PDEs or the shifted power method for eigenvalue problem. Then in [70], the hybrid Newton's method is used to accelerate the convergence of Newton's method for the kinematics of cable-driven parallel robots with sagging cables. Indeed, for this kind of problems, a random initial guess often leads to Newton's method failure to converge i.e a number of Newton iterations larger than a fixed threshold. In [42] addresses the numerical simulation of hyper-elasticity problems which also require to solve a non-linear PDE through Newton's method. In the standard Newton's method, 46% of their simulations would converge, this number increases to 71% using the Hybrid Newton's method. Finally in [6], a neural network is used to accelerate a non-linear iterative solver (which one is not mentioned) for some single gas-phase reservoir problems. Machine learning models are used to learn first a pressure guess, then this pressure guess is used to predict the gas properties. Finally, both pressure and gas predictions are used as initialization. Regarding the methodology, they first run a small part of the total reservoir simulation, then train the pressure and gas machine learning models on it. Thereafter, they start applying hybrid Newton's method and update the model at each time-step. This approach is interesting as it does not necessitate to construct a dataset in an offline phase, everything is constructed 'on the fly'.

Overall, the hybrid Newton's method is promising for cases where the standard Newton's initialization is far from the solution and we can not use information from the past solutions to interpolate for example. Therefore,

this method combines the predicting capability of neural networks and the numerical guarantees of the traditional solvers.

1.3 Outline and results summary of the thesis

This thesis aims to combine recent developments in physics informed deep learning with Newton's hybrid method in order to mitigate the impact of well events in the numerical simulation of multiphase flow in porous media.

In our case, an initialization by the current solution is most of the time far from the solution since a well opening or closing has a dramatic impact on the reservoir. As a well event, can be described by few parameters, we think that is is possible to develop a general methodology based on the hybrid Newton's method that ideally, may take into account any well parameters, reservoir parameters and as many wells as needed at the same time.

In practice, we developed three main points: i) machine learning model as a direct solver, ii) machine learning model as a global initializer of the iterative non-linear solver, iii) machine learning model as a local initializer of the iterative non-linear solver.

Before diving into these main points, chapter 2 presents the continuous mathematical model that we consider together with a reservoir use case. Then, chapter 3 exposes the numerical resolution of the continuous model and the numerical behaviour of well events.

In chapter 4, we assess the potential of machine learning models as 'black-box' direct solver. We start with a benchmark study on different machine-learning models as parameter-to-solution surrogate models for the steady-state heat equation with a source term. Our results indicate that Neural Operators such a Fourier Neural Operator (FNO) or DeepONet are well-suited neural architectures to learn PDE parameter-to-solution operators in a supervised manner. Nevertheless, the resulting prediction model has no guarantee to perform well at predicting data that are out of the training distribution. We then train in a supervised manner the Fourier Neural Operator to learn a parameter-to-solution operator for the 1D incompressible two-phase flow equation. Our results show that it is possible to predict accurately, given a

data snapshot, the solution after a step of time. However, when replacing the solver by the machine learning model and iterating through time by subsequently applying the model on its outputs, we observe that a small error in the prediction may lead to a significant error after a few iterations. Therefore, we show this 'black-box' approach lacks a guarantee of predicting solutions accurately, in contrast to the reliability of a numerical solver. Consequently, we turn in the next chapters to hybrid approaches that complement the numerical solver with a machine learning model while preserving the numerical guarantees.

We complement the traditional numerical solver in chapter 5 by using the hybrid Newton's method. It consists in predicting an initialization of Newton's method closer to the solution than the standard initialization, i.e. the solution at the previous time-step. Newton's method is an iterative method utilized to solve non-linear system of equations, and is sensible to initialization. Well events lead to numerical difficulties in Newton's method as they have an instant impact on the reservoir state, resulting in a standard initial guess far from the solution. In practice, we train in a supervised manner a Fourier Neural Operator to predict the saturation over the whole reservoir as the saturation contains the main non-linearities. Regarding the pressure, we use an implicit pressure solver that catches the main variations of pressure and requires to solve only a linear system as an initial guess. We generate two databases using the SHPCO_2 benchmark reservoir geometry and a wide range of well injection scenarios. We compare the required number of Newton iterations to achieve convergence between the Hybrid and the Standard Newton's methods using a global initialization. Our results show a significant decrease in the required number of Newton iterations, by around 54% for the first test case, and by around 39% for test case 2. Moreover, we observe that the Hybrid Newton's method seems to scale with the difficulty, meaning that the more challenging the problem is using the standard methodology, the greater the potential benefit from employing the hybrid methodology. Even though the global hybrid Newton's methodology shows significant performances at handling a wide a range of well scenarios, it only works for a constant well location and reservoir geometry.

Encouraged by these results, we wish to obtain a more general approach that handles any well location and reservoir geometry. Well events induce local variations of saturation, in the near-well region, but we trained a machine-learning model to predict over the whole domain. Therefore, in chapter 6,

we propose to train a model to predict the saturation in the near-well region. First, we apply this local approach on a 1D test case to assess the impact of the well extension and the well location on the hybrid Newton's method. Our results show a decrease of around 51% of the total number of Newton iterations using a small local domain, in contrast to 81% for a larger local domain. Moreover, when using the model trained on the small local domain to predict the saturation at any well location, we observe a reduction of around 83% of the total number of Newton iterations. Then, we apply the local hybrid Newton's method to the first test case of the global approach. We propose two hybrid initial guesses, the first one using a Fourier Neural Operator to predict the saturation and the second one using a fully implicit solver on the local domain which can be seen as a domain decomposition method. We apply those two approaches to three test cases. Overall, our results show that the domain decomposition initial guess performs equal or better for all test case compared to the machine learning initial guess but still less than the global approach initial guess. This is an important result as the domain decomposition local guess approach handles any well location and injection scenario by construction. Nevertheless, when moving to more difficult cases (3D geometry, complex physics, etc.), we expect the machine learning initial guess to infer significantly faster and hence provide an improvement with respect to the domain decomposition initial guess.

To conclude, the global hybrid Newton's method shows better performances than the local hybrid Newton's method for our test cases and is well-suited when the goal is to optimize well injection scenarios with constant well location. In opposite with the local hybrid Newton's method is well-suited when the goal is to optimize the well location for a wide range of injection scenarios. A straightforward perspective is to develop a 2D local model for any well locations and compare its performances with the domain decomposition approach.

Bibliography

[1] *Climate Change 2022 - Mitigation of Climate Change: Working Group III Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2023.

[2] International Energy Agency. *Energy Technology Perspectives 2017*. OECD/IEA, 2017.

[3] International Energy Agency. *Energy Technology Perspectives 2020: Special report on Carbon Capture Utilisation and Storage*. OECD/IEA, 2020.

[4] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, July 2019.

[5] Beguinet, Adrien, Ehrlacher, Virginie, Flenghi, Roberta, Fuente, Maria, Mula, Olga, and Somacal, Agustin. Deep learning-based schemes for singularly perturbed convection-diffusion problems*. *ESAIM: ProcS*, 73:48–67, 2023.

[6] Subhrajyoti Bhattacharyya and Aditya Vyas. A novel methodology for fast reservoir simulation of single-phase gas reservoirs using machine learning. *Heliyon*, 8(12):e12067, 2022.

[7] Bailian Chen, Dylan R. Harp, Youzuo Lin, Elizabeth H. Keating, and Rajesh J. Pawar. Geologic CO2 sequestration monitoring design: A machine learning and uncertainty quantification based approach. *Applied Energy*, 225(C):332–345, 2018.

[8] Tianping Chen and Hong Chen. Universal approximation to nonlinear

operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[9] Andrew K. Chu, Sally M. Benson, and Gege Wen. Deep-learning-based flow prediction for co2 storage in shale–sandstone formations. *Energies*, 16(1), 2023.

[10] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841, 2018.

[11] earth science society. Would co2 storage in deep saline aquifers carry an environnemental risk for shallow aquifers, June 2014.

[12] Svenja Ehlers, Marco Klein, Alexander Heinlein, Mathies Wedler, Nicolas Desmars, Norbert Hoffmann, and Merten Stender. Machine learning for phase-resolved reconstruction of nonlinear ocean wave surface elevations from sparse remote sensing data. *Ocean Engineering*, 288:116059, November 2023.

[13] Olga Fuks and Hamdi A. Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. 2020.

[14] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.

[15] R. González-García, R. Rico-Martínez, and I.G. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engineering*, 22:S965–S968, 1998. European Symposium on Computer Aided Process Engineering-8.

[16] Jules Guillot, Guillaume Koenig, Kadi Minbashian, Emmanuel Frénod, Hélène Flourent, and Julien Brajard. Partial differential equations for oceanic artificial intelligence. *ESAIM: Proceedings and Surveys*, 70:137–146, 01 2021.

[17] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, volume 14. 01 1996.

[18] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[19] Jianguo Huang, Haoqin Wang, and Haizhao Yang. Int-deep: A deep learning initialized iterative method for nonlinear problems. *Journal of Computational Physics*, 419:109675, 2020.

[20] Scott C. James, Yushan Zhang, and Fearghal O'Donncha. A machine learning framework to forecast wave conditions. *Coastal Engineering*, 137:1–10, 2018.

[21] Zhongyi Jiang, Min Zhu, Dongzhuo Li, Qiuzi Li, Yanhua O. Yuan, and Lu Lu. Fourier-mionet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration, 2023.

[22] M I Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. 5 1986.

[23] Daekyum Kim, Sang-Hun Kim, Taekyoung Kim, Brian Byunghyun Kang, Minhyuk Lee, Wookeun Park, Subyeong Ku, DongWook Kim, Junghan Kwon, Hochang Lee, Joonbum Bae, Yong-Lae Park, Kyu-Jin Cho, and Sungho Jo. Review of machine learning methods in soft robotics. *PLOS ONE*, 16(2):1–24, 02 2021.

[24] Sung Wook Kim, Iljeok Kim, Jonghwa Lee, and Seungchul Lee. Knowledge integration into deep learning in dynamical systems: an overview and taxonomy. *Journal of Mechanical Science and Technology*, 35:1331 – 1342, 2021.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[26] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces, 2023.

[27] Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. Characterizing possible failure modes in physics-informed neural networks, 2021.

[28] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[29] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images,

speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[30] Jean-Michel Lemieux. Review: The potential impact of underground geological storage of carbon dioxide in deep saline aquifers on shallow groundwater resources. *Hydrogeol J*, 19:757–778, 06 2011.

[31] Moshe Leshno, V. Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a non-polynomial activation function can approximate any function. *New York University Stern School of Business Research Paper Series*, 1991.

[32] Ben Li, Fujian Zhou, Hui Li, Andrew Duguid, Liyong Que, Yanpeng Xue, and Yanxin Tan. Prediction of co2 leakage risk for wells in carbon sequestration fields with an optimal artificial neural network. *International Journal of Greenhouse Gas Control*, 68:276–286, 2018.

[33] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020.

[34] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1–3):503–528, aug 1989.

[35] Mingliang Liu, Divakar Vashisth, Dario Grana, and Tapan Mukerji. Joint inversion of geophysical data for geologic carbon sequestration monitoring: A differentiable physics-informed neural network model. *Journal of Geophysical Research: Solid Earth*, 128, 03 2023.

[36] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *CoRR*, abs/1910.03193, 2019.

[37] F. Martínez-Álvarez, J. Reyes, A. Morales-Esteban, and C. Rubio-Escudero. Determining the best set of seismicity indicators to predict earthquakes. two case studies: Chile and the iberian peninsula. *Knowledge-Based Systems*, 50:198–210, 2013.

[38] Siddhartha Mishra. A machine learning framework for data driven acceleration of computations of differential equations, 2018.

[39] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. *IMA Journal of Numerical Analysis*, 42(2):981–1022, 06 2021.

[40] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA Journal of Numerical Analysis*, 43(1):1–43, 01 2022.

[41] Sridhar Narayan. The generalized sigmoid activation function: Competitive supervised learning. *Information Sciences*, 99(1):69–82, 1997.

[42] Alban Odot, Ryadh Haferssas, and Stéphane Cotin. Deepphysics: a physics aware deep learning framework for real-time simulation. *CoRR*, abs/2109.09491, 2021.

[43] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, 2022.

[44] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[45] Jean Rabault, Feng Ren, Wei Zhang, Hui Tang, and Hui Xu. Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization. *Journal of Hydrodynamics*, 32:234–246, 2020.

[46] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations. *CoRR*, abs/1711.10561, 2017.

[47] Deep Ray and Jan S. Hesthaven. An artificial neural network as a troubled-cell indicator. *Journal of Computational Physics*, 367:166–191, 2018.

[48] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[49] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[50] David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

[51] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[52] Parisa Shokouhi, Vikas Kumar, Sumedha Prathipati, Seyyed A. Hosseini, Clyde Lee Giles, and Daniel Kifer. Physics-informed deep learning for prediction of co2 storage site response. *Journal of Contaminant Hydrology*, 241:103835, 2021.

[53] Alexander Y. Sun. Optimal carbon storage reservoir management through deep reinforcement learning. *Applied Energy*, 278:115660, 2020.

[54] Hewei Tang, Qingkai Kong, and Joseph P. Morris. Multi-fidelity fourier neural operator for fast modeling of large-scale geological carbon storage, 2023.

[55] Meng Tang, Xin Ju, and Louis J. Durlofsky. Deep-learning-based coupled flow-geomechanics surrogate model for co2 sequestration. *International Journal of Greenhouse Gas Control*, 118:103692, 2022.

[56] Zeeshan Tariq, Ertugrul Umut Yildirim, Manojkumar Gudala, Bicheng Yan, Shuyu Sun, and Hussein Hoteit. Spatial–temporal prediction of minerals dissolution and precipitation using deep learning techniques: An implication to geological carbon sequestration. *Fuel*, 341:127677, 2023.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[58] Maria Han Veiga and Rémi Abgrall. Neural network based limiter with transfer learning, 2019.

[59] Hung Vo Thanh and Kang-Kun Lee. Application of machine learning to predict co2 trapping performance in deep saline aquifers. *Energy*, 239:122457, 2022.

[60] Hung Vo Thanh, Qamar Yasin, Watheq J. Al-Mudhafar, and Kang-Kun Lee. Knowledge-based machine learning techniques for accurate prediction of co2 storage performance in underground saline aquifers. *Applied Energy*, 314:118985, 2022.

[61] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks, 2020.

[62] Weiyu Wang and Keng Siau. Artificial intelligence, machine learning, automation, robotics, future of work and future of humanity: A review and research agenda. *Journal of Database Management (JDM)*, 30(1):61–79, 2019.

[63] Jing Wei, Xuan Chu, Xiang-Yu Sun, Kun Xu, Hui-Xiong Deng, Jigen Chen, Zhongming Wei, and Ming Lei. Machine learning in materials science. *InfoMat*, 1(3):338–358, 2019.

[64] Gege Wen, Catherine Hay, and Sally M. Benson. Ccsnet: A deep learning modeling suite for co2 storage. *Advances in Water Resources*, 155:104009, 2021.

[65] Gege Wen, Zongyi Li, Qirui Long, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. Real-time high-resolution co2 geological storage prediction using nested fourier neural operators. *Energy Environ. Sci.*, 16:1732–1741, 2023.

[66] Philipp A Witte, Russell Hewett, and Ranveer Chandra. Industry-scale co2 flow simulations with model-parallel fourier neural operators. In *NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning*, 2022.

[67] Philipp A. Witte, Tugrul Konuk, Erik Skjetne, and Ranveer Chandra. Fast co2 saturation simulations on large-scale geomodels with artificial intelligence-based wavelet neural operators. *International Journal of Greenhouse Gas Control*, 126:103880, 2023.

[68] Hao Wu, Nicholas Lubbers, Hari S. Viswanathan, and Ryan M. Pollyea. A multi-dimensional parametric study of variability in multi-phase flow dynamics during geologic co2 sequestration accelerated with machine learning. *Applied Energy*, 287:116580, 2021.

[69] Jinyu Wu and Jun Zhang. Physics-informed deep learning for predicting

flow around a circular cylinder. *Journal of Fluid Mechanics*, 892:A24, 2020.

[70] Ichrak Ben Yahia, Jean-Pierre Merlet, and Yves Papegay. Mixing neural networks and the Newton method for the kinematics of simple cable-driven parallel robots with sagging cables. In *ICAR 2021 - 20th International Conference on advanced robotics*, Ljulbjana, Slovenia, December 2021.

[71] Bicheng Yan, Bailian Chen, Dylan Robert Harp, Wei Jia, and Rajesh J. Pawar. A robust deep learning workflow to predict multiphase flow behavior during geological co2 sequestration injection and post-injection periods. *Journal of Hydrology*, 607:127542, 2022.

[72] Junyu You, William Ampomah, and Qian Sun. Development and application of a machine learning based multi-objective optimization workflow for co2-eor projects. *Fuel*, 264:116758, 2020.

Chapter 2

Physical problem formulation and use case

Contents

2.1 Incompressible two-phase flow problem **54**
2.1.1 Standard well model 57
2.2 SHPCO2 benchmark **57**
2.2.1 Physical configuration 59

Figures

2.1 1D from SHPCO2 case geometry [2] 59
2.2 2D SHPCO2 case geometry from [2] 60

This chapter aims at introducing the physical model problem considered in this manuscript. We then define in this chapter a reservoir example.

2.1 Incompressible two-phase flow problem

Let us consider a two-phase fluid composed of an aqueous phase noted w and a gaseous phase noted g flowing in a porous medium. The classical approach of modeling a two-phase flow in a porous medium is based on the application

of Darcy's law [3] for each phase $\alpha \in \{w, g\}$ with dimensionless factors kr_α in front of the tensor of permeability $\overline{\mathrm{K}}$, called relative permeabilities of the phases. If two fluids are flowing simultaneously, there is a reduction in the permeability of each fluid. The two fluids interfere with each other. This interference is modeled by the relative permeabilities which are functions depending on the saturation.

We consider the medium as isotropic. Therefore, the tensor of permeability $\overline{\mathrm{K}}$, which represents the ability of a rock to allow fluids to flow through its pores, can be considered as a scalar field K.

We consider reservoir geometries such that gravity can be neglected. For incompressible fluids, the equations describing the conservation of pore volume and phase volume, the conservation of momentum, the capillary forces between the two phases are written:

$$\phi \frac{\partial}{\partial t}(S_w) + div(v_w) = 0, \quad \phi \frac{\partial}{\partial t}(S_g) + div(v_g) = 0, \tag{2.1}$$

$$v_w = -\frac{K kr_w(S_w)}{\mu_w} \nabla P_w, \quad v_g = -\frac{K kr_g(S_g)}{\mu_g} \nabla P_g, \tag{2.2}$$

$$S_g + S_w = 1, \tag{2.3}$$

$$P_g - P_w = P_{c_{w,g}}(S_w). \tag{2.4}$$

Where S_α is the phase saturation, v_α the Darcy flow velocity of the phase in the porous medium, μ_α the phase viscosity, P_α the phase pressure, $P_{c_{g,w}}$ the capillary pressure and ϕ the medium porosity. The relative permeabilities kr_α are increasing functions of the saturation only.

We suppose that the permeability K and the porosity ϕ vary in space but are constant through time. The viscosities μ_α are constant.

Consider Ω a bounded domain of $\mathbb{R}^{\mathrm{N}}$ for $\mathrm{N} = 1$ or 2 such can be a reservoir and $T > 0$ a given time. Using the previous definitions and assumptions

and inserting (2.2) into (2.1), the problem of flow (2.1), (2.2), (2.3), (2.4) considered in Ω can be reformulated as follows:

$$\begin{cases} v + K(\lambda_T(S)\nabla P + \lambda_w(1-S)\nabla P_{c_{g,w}}(1-S)) & = 0 \\ div(v) & = 0 \quad \text{in } \Omega \times (0,T) \\ \phi\dfrac{\partial S}{\partial t} + div(\dfrac{\lambda_g(S)}{\lambda_T(S)}(v - K\lambda_w(1-S)\nabla P_{c_{g,w}}(1-S))) & = 0 \end{cases} \tag{2.5}$$

where $S = S_g$, $P = P_g$ and v is the total velocity:

$$v = v_g + v_w = -K(\lambda_T \nabla P + \lambda_w \nabla P_{c_{g,w}}).$$

The mobility λ_α of the phase $\alpha \in \{w, g\}$ is defined as:

$$\lambda_\alpha(S_\alpha) = \frac{kr_\alpha(S_\alpha)}{\mu_\alpha},$$

and the total mobility as:

$$\lambda_T(S) = \lambda_g(S) + \lambda_w(1-S).$$

In the remainder of these manuscript, it is assumed that the capillary pressure $P_{c_{g,w}}$ is neglected.

The system (2.5) can be simplified as:

$$\begin{cases} v + K\lambda_T(S)\nabla P & = 0 \\ div(v) & = 0 \quad \text{in } \Omega \times (0,T) \\ \phi\dfrac{\partial S}{\partial t} + div(f_g(S)v) & = 0 \end{cases} \tag{2.6}$$

Where the function $f_g(S)$ is called the fractional flow. It is defined by:

$$f_g(S) = \frac{\lambda_g(S)}{\lambda_T(S)} = \frac{\lambda_g(S)}{\lambda_w(1-S) + \lambda_g(S)}.$$

Let $\Gamma_D \subset \partial\Omega$ a part of the boundary of Ω and ${\Gamma_D}^- = \{x \in \Gamma_D / v(x).n(x) < 0\}$. The problem (2.6) closure is ensured by imposing the following boundary conditions:

$$\begin{cases} P = P_b(x) & \forall x \in \Gamma_D, \\ S = S_b(x) & \forall x \in {\Gamma_D}^-, \\ v.n = 0 & \text{on } \partial\Omega \backslash \Gamma_D. \end{cases} \tag{2.7}$$

We consider in this thesis that the boundary conditions does not vary in time, therefore we remove the dependency in time in the equation (2.7).

Also, the initial conditions at $t = 0$ are defined as $P(x, t = 0) = P_0(x)$ and $S(x, t = 0) = S_0(x), x \in \Omega$.

2.1.1 Standard well model

A well can be modeled by a source term in the conservation equation. The equations (2.1) become:

$$\phi \frac{\partial}{\partial t}(S_w) + div(v_w) = q_w, \quad \phi \frac{\partial}{\partial t}(S_g) + div(v_g) = q_g, \tag{2.8}$$

and the system 2.6:

$$\begin{cases} v + K\lambda_T(S)\nabla P & = 0, \\ div(v) & = q_T, \\ \phi \dfrac{\partial S}{\partial t} + div(f_g(S)v) & = q_g, \end{cases} \quad \text{in } \Omega \times (0, T), \tag{2.9}$$

q_w, q_g and $q_T = q_w + q_g$ are respectively the water flow, gas flow and the total injected or produced flow in the wells.

2.2 SHPCO2 benchmark

As a practical use case, we use the SHPCO_2 [2] benchmark to test our methods. This benchmark was created for modelling reactive transport for CO_2 geological storage. The SHPCO_2 geological configuration is inspired from

Sleipner, the world's first commercial CO_2 storage project. Sleipner is an area located in the North-Sea, belonging to Norway and exploited for its natural gas field since the mid-1990s.

As the natural gas produced contains up to 9% CO_2 , the Sleipner CO_2 gas processing and capture unit is built to evade the expensive 1991 Norwegian CO_2 tax. The captured CO_2 is thus injected and stored in a deep saline formation one kilometer below the seabed.

One of the main risk of CO_2 storage is leakage which is a human risk associated to the escape of CO_2 from a deep reservoir to a shallow aquifer or to the surface [1]. Therefore, reservoir simulation is key to confidently predict and understand CO_2 behavior inside the geological formation.

Simulation context

We aim to reproduce the SHPCO_2 synthetic test case but with simplified physics for simulation. We consider an incompressible and immiscible two-phase flow problem without chemical reaction. The targeted deep saline aquifer is heterogeneous and located one kilometer below the seabed. The area of interest is 4,750 meters in East-West direction and 3,000 meters in north-south direction. The geological formation has a thickness of 100 meters.

We assume that the geological layers limiting the aquifer in its upper and lower parts are impermeable. The aquifer primarily consists of high-permeability sandstone, with several low-permeability barriers. Flow is allowed on three lateral surfaces: '*Injector1*', '*Injector2*', and '*Productor*'. Different uniform pressures can be imposed on these surface boundaries. The overall flow in the aquifer is directed from west to east but is significantly influenced by the presence of the barriers.

1D and 2D Geometries

We consider the one and two dimensional cases, their geometries are described in the figures below. The mesh sizes (XS, S, M, L) have different purposes. The Extra Small (XS) is for setting up the code, Small (S) and Medium (M) meshes are used for testing new methods and Large (L) is challenging and used for robustness tests.

1D Geometry: see figure 2.1 and table 2.1

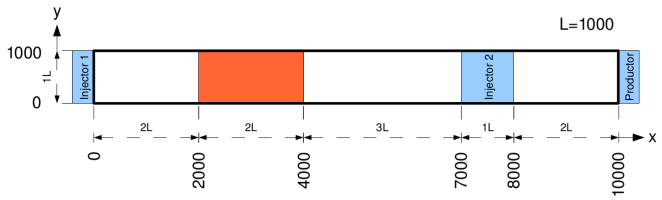


Figure 2.1: 1D from SHPCO2 case geometry [2]

Mesh	XS	S	M	L
Dx	250	50	10	5
Dy	1000	1000	1000	1000
Dz	100	100	100	100
Nx	40	200	1000	2000
Ny	1	1	1	1
Nz	1	1	1	1
NCell	40	200	1000	2000

Table 2.1: 1D mesh parameters

2D Geometry: see figure 2.2 and table 2.2

Modifications in gas zone

We want to study well events, however in the SHPCO_2 configuration, there are no well openings or closings. We realise a modification to fit more to our case. The gas zone in orange on figures 2.1 and 2.2 is replaced by a well that injects CO_2 at specific times. Therefore, there is no gas in the initial configuration, all gas comes from this injector well that we denote *well CO2*.

2.2.1 Physical configuration

Petrophysical properties

The domain after modification is separated in two zones, the first zone called "Barrier zone" is coloured in green on the 2D figure and the second zone called "Drain zone" is formed of the rest of the domain. These two domains have different petrophysical properties.

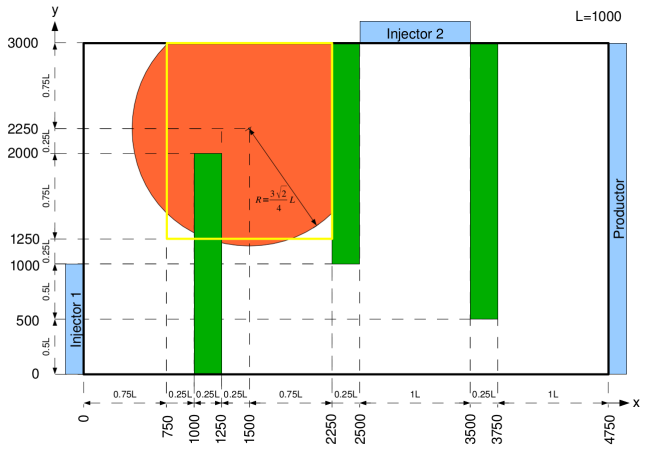


Figure 2.2: 2D SHPCO2 case geometry from [2]

Mesh	XS	S	M	L
Dx	250	50	10	5
Dy	250	50	10	5
Dz	100	100	100	100
Nx	19	95	475	950
Ny	12	60	300	600
Nz	1	1	1	1
NCell	228	5700	142 500	570 000

Table 2.2: 2D mesh parameters

	Barrier zone	Drain zone
Porosity [-]	0.2	0.2
Permeability[m^2]	1.e-15	100.e-15

Table 2.3: Petrophysical Parameters

Phase properties

We consider two models of relative permeabilities:

	Gas phase	Water phase
Viscosity [Pa.s]	0.0285e-03	0.571e-03

Table 2.4: Physical properties of fluids

$$\text{Cross kr: } kr_g(S_g) = S_g \quad \text{and} \quad kr_w(S_w) = S_w.$$
$$\text{Quadratic kr: } kr_g(S_g) = S_g^2 \quad \text{and} \quad kr_w(S_w) = S_w^2.$$

Boundary Conditions

	Pressure[Pa]	Composition
Injector 1	110.e+05	Water
Injector 2	105.e+05	Water
Productor	100.e+05	-
CO2 injector	-	Gas

Table 2.5: Limit condition parameters

The pressure for the CO_2 Injector (*well CO2*) is not given here because it may change during each experiment, we will precise its value every time it is needed. The composition of the Productor well is not given, it produces the fluid composition of its location in the reservoir.

Bibliography

[1] Adam L. Atchley, Reed M. Maxwell, and Alexis K. Navarre-Sitchler. Human health risk assessment of co2 leakage into overlying aquifers using a stochastic, geochemical reactive transport approach. *Environmental Science & Technology*, 47(11):5954–5962, 2013. PMID: 23618095.

[2] Florian Haeberlein. *Time Space Domain Decomposition Methods for Reactive Transport — Application to CO2 Geological Storage*. Theses, Université Paris-Nord - Paris XIII, October 2011.

[3] M. King Hubbert. Darcy's Law and the Field Equations of the Flow of Underground Fluids. *Transactions of the AIME*, 207(01):222–239, 12 1956.

Chapter 3

Numerical resolution

Contents

3.1 Mesh definition and notations **64**
3.2 IMPES scheme **66**
3.2.1 Solving the pressure problem 66
3.2.2 Solving the saturation problem 68
3.2.3 Taking wells into account 69
3.3 IMPIMS scheme **71**
3.4 Fully Implicit scheme **72**
3.5 Time-step management **74**
3.5.1 Time stepping for each scheme 74
3.6 YADS reservoir simulation library **75**
3.6.1 YADS feature blocks 77
3.7 Numerical behaviour during well event **82**

In this chapter we present three classical finite volume methods to numerically solve the problem (2.6) then we expose how the time step is managed through simulations. Thereafter, we present the python library developed to run the simulations. Finally, we expose the usual numerical behaviour during well events.

3.1 Mesh definition and notations

Let $\mathcal{K}_h$ a finite set of non empty, open and disjoint polyhedra forming a partition of Ω such that $h = \underset{k\in\mathcal{K}_h}{max}\ h_k$ where h_k is the diameter of element $k \in \mathcal{K}_h$. The set $\mathcal{K}_h$ thus forms a mesh of Ω whose elements $k \in \mathcal{K}_h$ are the cells. We define N_h as the number of cells in the mesh $\mathcal{K}_h$. Similarly, we denote $\mathcal{F}_h$ the set of faces in the mesh. Moreover, we define $\mathcal{F}_h^i$ the set of internal faces of the mesh

$$\mathcal{F}_h^i = \{\sigma \in \mathcal{F}_h | \sigma \not\subset \partial\Omega.\}$$

We denote $\mathcal{F}_h^k$ the set of boundary faces

$$\mathcal{F}_h^k = \{\sigma \in \mathcal{F}_h | \sigma \subset \partial\Omega\}.$$

The set of faces $\mathcal{F}_h$ of the mesh $\mathcal{K}_h$ then verifies $\mathcal{F}_h = \mathcal{F}_h^i \cup \mathcal{F}_h^k$. We denote $\mathcal{F}_{h,k}$ the set of faces of a cell $k \in \mathcal{K}_h$:

$$\mathcal{F}_{h,k} = \{\sigma \in \mathcal{F}_h | \sigma \subset \partial k\}.$$

Moreover, we denote $\mathcal{F}_{h,k}^i$ the set of internal faces of a cell $k \in \mathcal{K}_h$:

$$\mathcal{F}_{h,k}^i = \mathcal{F}_{h,k} \cap \mathcal{F}_h^i,$$

and $\mathcal{F}_{h,k}^b$, the set of faces of a cell k that are located on the border of the domain Ω

$$\mathcal{F}_{h,k}^b = \mathcal{F}_{h,k} \cap \mathcal{F}_h^k.$$

$\mathcal{F}_{h,k}^b = \emptyset$ if the cell k does not share faces with $\partial\Omega$.

In the problem (2.9), boundary conditions (2.7) are imposed on a border $\Gamma_D \subset \partial\Omega$. We then denote $\mathcal{F}_h^D$ the set of border faces included in Γ_D:

$$\mathcal{F}_h^D = \{\sigma \in \mathcal{F}_h^k | \sigma \subset \Gamma_D\},$$

and $\mathcal{F}_{h,k}^D$ the set of faces of the cell $k \in \mathcal{K}_h$, that are located on Γ_D

$$\mathcal{F}_{h,k}^{D} = \mathcal{F}_{h,k} \cap \mathcal{F}_{h}^{D}.$$

We denote by Π a set of wells, $\mathcal{K}_i^{\Pi}$ the set of cells perforated by a well i and $\mathcal{K}_{k,i}^{\Pi}$, the cell k perforated by a well $i \in \Pi$

$$\mathcal{K}_{k,i}^{\Pi} = \{k \in \mathcal{K}_i^{\Pi} | i \in \Pi\}.$$

In a similar way to $\mathcal{F}_{h,k}$, we note $\mathcal{K}_{h,\sigma}$ the set of cells $k \in \mathcal{K}_h$ located on both side of a face $\sigma \in \mathcal{F}_h$:

$$\mathcal{K}_{h,\sigma} = \{k, l \in \mathcal{K}_h | \sigma = \partial k \cap \partial l\}.$$

In particular, the set $\mathcal{K}_{h,\sigma}$ contains exactly two elements if $\sigma \in \mathcal{F}_h^i$ and one element if $\sigma \in \mathcal{F}_h^k$.

For each interface $\sigma \in \mathcal{F}_h^i$, we define an unitary normal n_σ arbitrarily oriented. For a mesh $k \in \mathcal{K}_{h,\sigma}$, let $n_{\sigma,k}$ the unitary σ-normal outgoing from k.

For a cell $k \in \mathcal{K}_h$, we identify its center as x_k . Similarly, for any face $\sigma \in \mathcal{F}_h$, x_σ is the center of the face. For any mesh $k \in \mathcal{K}_h$, and for any face $\sigma \in \mathcal{F}_{h,k}$, we note:

$$d_{k,\sigma} = dist(x_k, x_\sigma).$$

Similarly, for any $k_1, k_2 \in \mathcal{K}_h$:

$$d_{k_1,k_2} = dist(x_{k_1}, x_{k_2}).$$

For a face $\sigma \in \mathcal{F}_h$, $|\sigma|$ is the face area. In the same way, for a cell $k \in \mathcal{K}_h$, $|k|$ is the cell volume.

In this work, we consider admissible meshes [4]. It consists in the following formulation: $\forall \sigma \in \mathcal{F}_h^i$ such that $\sigma = \partial k_1 \cap \partial k_2$, the segment $[x_{k_1}, x_{k_2}]$ is orthogonal to σ. Similarly, if $\sigma \in \mathcal{F}_h^b$ such that $\sigma \in \partial k$, the segment $[x_k, x_\sigma]$ is orthogonal to σ.

The main admissible meshes are the cartesian meshes, the Voronoï meshes and the triangular meshes if all angles of triangles are smaller than $\frac{\Pi}{2}$.

Concerning the time discretization, we introduce a sequence of discrete times $\{t^n\}_{0 \leq n \leq N}$

$$0 = t^0 < ... < t^N = \tau,$$

which divide the time interval $(0, \tau)$ into N time steps $\delta t = t^{n+1} - t^n$. Thereafter, we denote $P^n = P(x, t^n)$, $S^n = S(x, t^n)$ and P_k^n, respectively S_k^n approximations of $P(x_k, t^n)$ and $S(x_k, t^n)$.

We also introduce the notation $(u)^+ = max(u, 0)$ and $(u)^- = min(u, 0)$.

3.2 IMPES scheme

The IMPES (Implicit in Pressure and Explicit in Saturation) [8] [2] [5] numerical scheme allows to decouple the resolution in pressure and saturation. This scheme is conditionally stable (Courant-Friedrichs-Lewy (CFL) condition). Considering the problem (2.6), we solve the semi-discrete system:

$$div(v(P^{n+1}, S^n)) = 0, \tag{3.1}$$

$$\phi \frac{S^{n+1} - S^n}{\delta t} + div(f_g(S^n)v(P^{n+1}, S^n) = 0. \tag{3.2}$$

We solve the pressure equation using 3.1 and the saturation problem with 3.2.

3.2.1 Solving the pressure problem

In the first place, we are interested in the pressure P^{n+1}. To compute P^{n+1}, equation (3.1) is integrated over all cells $k \in \mathcal{K}_h$. We then use Green's formula:

$$\int_k div(v)\mathrm{d}x = \sum_{\sigma \in \mathcal{F}_{h,k}} \int_\sigma v.n_{k,\sigma}\mathrm{d}s = \sum_{\sigma \in \mathcal{F}_{h,k}^i} \int_\sigma v.n_{k,\sigma}\mathrm{d}s + \sum_{\sigma \in \mathcal{F}_{h,k}^b} \int_\sigma v.n_{k,\sigma}\mathrm{d}s.$$

Diffusive fluxes on inner faces
For $\sigma \in \mathcal{F}_h^i$ such that $\sigma = \partial k \cap \partial l$

$$\int_\sigma v(P^{n+1}, S^n).n_{k,\sigma} \mathrm{d}s \simeq \Phi_{k,\sigma}^i,$$

with:

$$\Phi_{k,\sigma}^i = \lambda_{T,\sigma}^{i,n} T_\sigma^i (P_k^{n+1} - P_l^{n+1}),$$

where T_σ^i the face transmissivity is calculated using a harmonic average:

$$T_\sigma^i = |\sigma| \frac{K_k K_l}{K_l d_{k,\sigma} + K_k d_{l,\sigma}},$$

and the total mobility $\lambda_{T,\sigma}^{i,n}$ is computed through an up-winding from the pressure gradient:

$$\lambda_{T,\sigma}^{i,n} = \begin{cases} \lambda_T(S_k^n) & \text{if } P_k^n \geq P_l^n, \\ \lambda_T(S_l^n) & \text{else.} \end{cases}$$

Diffusive fluxes on boundary faces
For $\sigma \in \mathcal{F}_h^b$ such that $\sigma \in \partial k$

$$\int_\sigma v(P^{n+1}, S^n).n_{k,\sigma} \mathrm{d}s \simeq \Phi_{k,\sigma}^D,$$

with:

$$\Phi_{k,\sigma}^D = \lambda_{T,\sigma}^{D,n} T_\sigma^D (P_k^{n+1} - P_{b,\sigma}),$$

where T_σ^D is given by:

$$T_\sigma^D = |\sigma| \frac{K_k}{d_{k,\sigma}},$$

and the total mobility $\lambda_{T,\sigma}^{D,n}$ is computed through an up-winding from the pressure gradient:

$$\lambda_{T,\sigma}^{D,n} = \begin{cases} \lambda_T(S_k^n) & \text{if } P_k^n \geq P_{b,\sigma}, \\ \lambda_T(S_{b,\sigma}) & \text{else.} \end{cases}$$

$P_{b,\sigma}$ and $S_{b,\sigma}$ are respectively approximations of P_b and S_b on $\sigma \in \mathcal{F}_h^b$.

Solving the linear system

The discrete equation associated to a cell k for the pressure problem is:

$$\sum_{\sigma \in \mathcal{F}_{h,k}^i} \Phi_{k,\sigma}^i + \sum_{\sigma \in \mathcal{F}_{h,k}^D} \Phi_{k,\sigma}^D = 0. \tag{3.3}$$

After assembly, we obtain a linear system of size N_h (number of cells), that can be written in the generic form:

$$Ax = b,$$

where $x = P^{n+1}$.

3.2.2 Solving the saturation problem

Once the pressure P^{n+1} is computed, the saturation is explicitly solved for a given cell k:

$$\int_k \left(\phi \frac{S^{n+1} - S^n}{\delta t} + div(f_g(S^n) v(P^{n+1}, S^n)) \right) \mathrm{d}x = 0.$$

Using the divergence formula:

$$\int_k \left(\phi \frac{S^{n+1} - S^n}{\delta t} \right) \mathrm{d}x + \sum_\sigma \int_\sigma f_g(S^n) v(P^{n+1}, S^n).n_{k,\sigma} \mathrm{d}\sigma = 0.$$

Accumulation term
We approximate the accumulation term:

$$\int_k \left(\phi \frac{S^{n+1} - S^n}{\delta t} \right) \mathrm{d}x \simeq |k| \phi_k \frac{S_k^{n+1} - S_k^n}{\delta t}.$$

Flux term
For a face $\sigma \in \mathcal{F}_{h,k}^i$ such that $\sigma = \partial k \cap \partial l$, we write:

$$\int_\sigma f_g(S^n) v(P^{n+1}, S^n).n_{k,\sigma} \mathrm{d}\sigma \simeq F_{g,\sigma}^i(S_k^n, S_l^n),$$

with

$$F_{g,\sigma}^i(S_k^n, S_l^n) = f_g(S_k^n)(\Phi_{k,\sigma}^i)^+ + f_g(S_l^n)(\Phi_{k,\sigma}^i)^-,$$

for faces in $\mathcal{F}^b_{h,k} \cap \mathcal{F}^D_{h,\sigma}$

$$\int_\sigma f_g(S^n) v(P^{n+1}, S^n).n_{k,\sigma} \mathrm{d}\sigma \simeq F^D_{g,\sigma}(S^n_k, S_{b,\sigma}),$$

with

$$F^D_{g,\sigma}(S^n_k, S_{b,\sigma}) = f_g(S^n_k)(\Phi^D_{k,\sigma})^+ + f_g(S_{b,\sigma})(\Phi^D_{k,\sigma})^-.$$

Solving
The discrete equation associated to cell k becomes:

$$S^{n+1}_k = S^n_k - \frac{\delta t}{|k|\phi_k} \left(\sum_{\sigma \in \mathcal{F}^i_{h,k}} F^i_{g,\sigma}(S^n_k, S^n_l) + \sum_{\sigma \in \mathcal{F}^D_{h,k}} F^D_{g,\sigma}(S^n_k, S_{b,\sigma}) \right)$$

Due to the explicit resolution of the saturation, the time-step δt must satisfy a CFL condition.

$$\delta t \leq \frac{\underset{k \in \mathcal{K}_h}{inf} \left(|k| \phi_k \right)}{\underset{k \in \mathcal{K}_h}{sup} \left(- \sum_{\sigma \in \mathcal{F}^i_{h,k}} (F^i_{g,\sigma})^- - \sum_{\sigma \in \mathcal{F}^D_{h,k}} (F^D_{g,\sigma})^- \right) \underset{S \in [0,1]}{sup} \left(f'_g(S) \right)}.$$

3.2.3 Taking wells into account

On a discrete level, wells can be represented by source terms, equation (3.1) and (3.2) become:

$$div(v(P^{n+1}, S^n)) = q_T, \tag{3.4}$$

$$\phi \frac{S^{n+1} - S^n}{\delta t} + div(f_g(S^n) v(P^{n+1}, S^n) = q_g. \tag{3.5}$$

Solving the pressure problem

The discrete pressure problem presented in section 3.2.1 and associated to a cell k then becomes:

$$\sum_{\sigma \in \mathcal{F}^i_{h,k}} \Phi^i_{k,\sigma} + \sum_{\sigma \in \mathcal{F}^D_{h,k}} \Phi^D_{k,\sigma} - \sum_{i \in \Pi | \mathcal{K}(i) = k} q^\Pi_{T_{k,i}} = 0. \tag{3.6}$$

with $q^{\Pi}_{T_{k,i}}$ the total flux at the well i in the perforated cell k between t^n and t^{n+1}, Π the set of wells, $\mathcal{K}^{\Pi}_i$ the set of perforated cells by a well $i \in \Pi$. If $q^{\Pi}_{T_{k,i}} > 0$ the well is injector, else if $q^{\Pi}_{T_{k,i}} < 0$ the well is producer.

Diffusive fluxes at the wells

There are two well operation modes. Either a constant total flow rate q_T or a constant pressure P^k_i. In this last case, the total flux at the wells are approximated by:

$$q^{\Pi}_{T_{k,i}} = -IP_k \lambda^{\Pi}_{T_{k,i}} (P^{n+1}_k - P^{\Pi}_i),$$

where the total mobility $\lambda^{\Pi}_{T_{k,i}}$ is computed through an up-winding from the pressure gradient:

$$\lambda^{\Pi}_{T_{k,i}} = \begin{cases} \lambda_T(S^n_k) & \text{if } P^n_k \geq P^{\Pi}_i, \\ \lambda_T(S^{\Pi}_i) & \text{else,} \end{cases}$$

and where S^{Π}_i is the gas saturation imposed at the well i and the production index of the cell k IP_k is given by Peaceman's formula [6]:

$$IP_k = 2\pi \frac{K_k}{\log(r_e/r_w)},$$

with $r_e \simeq 0.14\sqrt{dx^2 + dy^2}$, dx and dy are the respective perforated cell length and width of a cartesian mesh and r_w is the well radius.

Solving the linear system

After assembly of 3.6, we obtain a linear system of size N_h (number of cells), that can be written in the generic form:

$$Ax = b,$$

where $x = P^{n+1}$.

Solving the saturation problem with wells

The discrete equation associated to cell k is:

$$S_k^{n+1} = S_k^n - \frac{\delta t}{|k|\phi_k}\left(\sum_{\sigma\in\mathcal{F}_{h,k}^i} F_{g,\sigma}^i(S_k^n, S_l^n) + \sum_{\sigma\in\mathcal{F}_{h,k}^D} F_{g,\sigma}^D(S_k^n, S_{b,\sigma}) + \sum_{i\in\Pi|k\in\mathcal{K}_i^\Pi} F_{g,i}^\Pi(S_k^n, S_i^\Pi)\right),$$

where S_i^Π is the injected gas saturation of well $i \in \Pi$ and:

$$F_{g,i}^\Pi(S_k^n, S_i^\Pi) = f_g(S_k^n)(-q_{T_{k,i}}^\Pi)^+ + f_g(S_i^\Pi)(-q_{T_{k,i}}^\Pi)^-.$$

Due to the explicit resolution of the saturation, the time step δt must satisfy a CFL condition (Courant-Friedrichs-Lewy).

$$\delta t \leq \frac{\inf\limits_{k\in\mathcal{K}_h}(|k|\phi_k)}{\sup\limits_{k\in\mathcal{K}_h}\left(-\sum\limits_{\sigma\in\mathcal{F}_{h,k}^i}(F_{g,\sigma}^i)^- - \sum\limits_{\sigma\in\mathcal{F}_{h,k}^D}(F_{g,\sigma}^D)^- - \sum\limits_{i\in\Pi|k\in\mathcal{K}_i^\Pi}(F_{g,i}^\Pi)^-\right)\sup\limits_{S\in[0,1]}\left(f_g'(S)\right)}.$$

3.3 IMPIMS scheme

As for the IMPES scheme, the pressure and saturation resolutions are decoupled. In the IMPIMS (Implicit in Pressure and Implicit in Saturation) scheme [3][5], the time integration is also of Euler type, the pressure is always implicit but the saturation is implied to calculate the fractional flow. We solve the system

$$div(v(P^{n+1}, S^n)) = q_T, \tag{3.7}$$

$$\phi\frac{S^{n+1} - S^n}{\delta t} + div(f_g(S^{n+1})v(P^{n+1}, S^n) = q_g. \tag{3.8}$$

Therefore, the pressure is computed as in section 3.2.3 and allows us to obtain the total velocities. Then to compute the saturation at time t^{n+1}, we realise a finite volume discretization of equation (3.8). For any cell $k \in \mathcal{K}_h$

$$S_k^{n+1} = S_k^n - \frac{\delta t}{|k|\phi_k}\left(\sum_{\sigma\in\mathcal{F}_{h,k}^i} F_{g,\sigma}^{i,n+1} + \sum_{\sigma\in\mathcal{F}_{h,k}^D} F_{g,\sigma}^{D,n+1} - \sum_{i\in\Pi|k\in\mathcal{K}_i^\Pi} F_{g,i}^{\Pi,n+1}\right), \tag{3.9}$$

with the discrete diffusive flux defined using an up-winding such in paragraph 3.2.3 and 3.2.2.

$$\begin{aligned} F_{g,i}^{\Pi,n+1} &= f_g(S_k^{n+1})(-q_{T_{k,i}}^{\Pi})^+ + f_g(S_i^{\Pi})(q_{T_{k,i}}^{\Pi})^-, \\ F_{g,\sigma}^{i,n+1} &= f_g(S_k^{n+1})(\Phi_{k,\sigma}^{i})^+ + f_g(S_l^{n+1})(\Phi_{k,\sigma}^{i})^-, \\ F_{g,\sigma}^{D,n+1} &= f_g(S_k^{n+1})(\Phi_{k,\sigma}^{D})^+ + f_g(S_{b,\sigma})(\Phi_{k,\sigma}^{D})^-. \end{aligned}$$

The calculation of the saturations S_k^{n+1} requires the resolution of a nonlinear system because of the expression of the fractional flow function f_g. This resolution is done using Newton's method. By evaluating the flows implicitly, the obtained solution respects the maximum principle whatever the chosen time step.

3.4 Fully Implicit scheme

For the fully implicit scheme, the pressure and saturation are solved simultaneously and the time integration is still of Euler type.

We consider the system (2.8) written in the following form:

$$\begin{cases} \phi \dfrac{\partial}{\partial t}(S_w) + div(v_w) = q_w, \\ \phi \dfrac{\partial}{\partial t}(S_g) + div(v_g) = q_g, \end{cases} \tag{3.10}$$

with $S_w + S_g = 1$, and where v_w, v_g are given by (2.2) and q_g, q_w are the water flow and gas flow in the wells.

The fully implicit discretization of (3.10) gives $\forall k \in \mathcal{K}_h$ and for $\alpha \in \{w, g\}$:

$$|k|\phi_k \frac{S_{\alpha,k}^{n+1} - S_{\alpha,k}^{n}}{\delta t} + \left(\sum_{\sigma \in \mathcal{F}_{h,k}^{i}} F_{\alpha,\sigma}^{i}(P^{n+1}, S_\alpha^{n+1}) + \sum_{\sigma \in \mathcal{F}_{h,k}^{D}} F_{\alpha,\sigma}^{D}(P^{n+1}, S_\alpha^{n+1}) \right) + \sum_{i \in \Pi | k \in \mathcal{K}_i^{\Pi}} F_{\alpha,k}^{\Pi,n+1}(P^{n+1}, S_\alpha^{n+1}) = 0. \tag{3.11}$$

The phase fluxes are approximated as follows:

For a face $\sigma \in \mathcal{F}^i_{h,k}$ such that $\sigma = \partial k \cap \partial l$:

$$F^i_{\alpha,\sigma}(P^{n+1}, S^{n+1}) = \lambda^{i,n+1}_{\alpha,\sigma} T^i_\sigma (P^{n+1}_k - P^{n+1}_l),$$

where:

$$\lambda^{i,n+1}_{\alpha,\sigma} = \begin{cases} \lambda_\alpha(S^{n+1}_{\alpha,k}) & \text{if } T^i_\sigma(P^{n+1}_k - P^{n+1}_l) \geq 0, \\ \lambda_\alpha(S^{n+1}_{\alpha,l}) & \text{if } T^i_\sigma(P^{n+1}_k - P^{n+1}_l) < 0, \end{cases}$$

for faces in $\mathcal{F}^b_{h,k} \cap \mathcal{F}^D_{h,\sigma}$

$$F^D_{\alpha,\sigma}(P^{n+1}, S^{n+1}) = \lambda^{D,n+1}_{\alpha,\sigma} T^D_\sigma (P^{n+1}_k - P_\sigma),$$

where

$$\lambda^{D,n+1}_{\alpha,\sigma} = \begin{cases} \lambda_\alpha(S^{n+1}_{\alpha,k}) & \text{if } T^D_\sigma(P^{n+1}_k - P^\sigma) \geq 0, \\ \lambda_\alpha(S_{b,\sigma}) = & \text{if } T^D_\sigma(P^{n+1}_k - P_\sigma) < 0. \end{cases}$$

For cells $k \in \mathcal{K}^\Pi_i$

$$F^{\Pi,n+1}_{\alpha,k}(P^{n+1}, S^{n+1}) = \lambda^{\Pi,n+1}_{\alpha,i} IP_k (P^{n+1}_k - P^\Pi_i),$$

where

$$\lambda^{\Pi,n+1}_{\alpha,i} = \begin{cases} \lambda_\alpha(S^{n+1}_{\alpha,k}) & \text{if } IP_k(P^{n+1}_k - P^\Pi_i) \geq 0, \\ \lambda_\alpha(S^\Pi_{\alpha,i}) = & \text{if } IP_k(P^{n+1}_k - P^\Pi_i) < 0. \end{cases}$$

We still choose P and $S = S_g$ as primary unknowns and $S_w = 1 - S$ as secondary unknown.

For each cell k, the system (3.11) can be written as:

$$\begin{cases} R_{g_k}(P^{n+1}, S^{n+1}) & = 0, \\ R_{w_k}(P^{n+1}, 1 - S^{n+1}) & = 0. \end{cases} \tag{3.12}$$

This non linear system of 2N equations is solved using Newton's method.

The fully implicit scheme is unconditionally stable but taking large time steps can make Newton's method diverge. The size of the linear system to be solved at each iteration is more important than for other explicit schemes (all unknowns are coupled).

Newton's method stopping criterion

During the numerical resolution, we use a stopping criterion for Newton's method based on the residual norm.

At the Newton iteration number I, the instantaneous balance for a phase α in a cell $k \in \mathcal{K}_h$ is:

$$B^I_{\alpha,k} = |R^I_{\alpha,k}| \frac{\delta t}{|k|}, \tag{3.13}$$

where $|k|$ aims to put into perspective the error between the large and small cells. Then, the stopping criterion is:

$$\max_{\alpha \in \{w,g\}} \max_{k=1,N} B^I_{\alpha,k} < \epsilon, \tag{3.14}$$

with epsilon the stopping criterion value. This criterion can be seen as a test on the residual infinity norm.

3.5 Time-step management

In this section, we expose specific numerical aspects related to the reservoir simulation.

3.5.1 Time stepping for each scheme

IMPES

For this scheme, the time step is driven by the CFL condition.

IMPIMS and Fully implicit

For those two schemes, the time step is driven by Newton's method and by a test on variable variations. In this thesis, we do not use the test on variable variations as it relies on several hyperparameters that are chosen empirically.

Newton time stepping

The time-step management driven by Newton's method is described by algorithm 1. We use relaxation technique which restrains the variable variations and clipping which ensures that the saturation remains physical (i.e $0 \leq S \leq 1$).

As presented in the algorithm 1, the time step is reduced if the Newton's resolution does not converge in a certain number of iterations. This number of iterations highly depends on the initialization. If we assume that there are no big variations in pressure and saturation between t^n and t^{n+1}; in this case the time step is not lowered.

Time-step management through events

Starting from a time t_0 and with an event such as a well opening or closing at a time t_1, we want to completely capture the event by choosing wisely the time step dt that reaches it. For now, we use the following method:

Given t_0 the actual time, t_1 the event time with $t_1 > t_0$ and a time step dt_0. If $t_0 + dt_0 > t_1$ then we miss the event by keeping this time step. We so chose $dt_1 = t_1 - t_0$ in order to synchronize with the exact beginning of the event.

3.6 YADS reservoir simulation library

This work involves running reservoir simulations using the numerical scheme presented above, and artificial intelligence models can easily integrated into the process. To the best of our knowledge, we haven't found any available library that meets these criteria. Therefore, we have developed our own: YADS which stands for 'Yet another Darcy Solver'. YADS is fully written in python and will be released as open-source at the end of this thesis at the following link. We will start by presenting the main building blocks and their key features. Then, we will provide a concrete script example that demonstrates the use of some key features.

Algorithm 1 Newton algorithm for Fully Implicit scheme

Initialization: $P^{I=0} = P^n$, $S^{I=0} = S_g^n$ (and $S_w^{I=0} = S_w^n$), δS_{max}
Iteration
for $I \leftarrow 1$ to **NMAXITER do**
Solving the linear system:

$$\begin{pmatrix} \frac{\partial R_g^I}{\partial P} & \frac{\partial R_g^I}{\partial S} \\ \frac{\partial R_w^I}{\partial P} & \frac{\partial R_w^I}{\partial S} \end{pmatrix} \begin{pmatrix} \delta P \\ \delta S \end{pmatrix} = - \begin{pmatrix} R_g(P^I, S^I) \\ R_w(P^I, 1 - S^I) \end{pmatrix} \tag{3.15}$$

Update unknowns

$\alpha = min(1, \delta S_{max}/max(\delta S))$ ▷ Compute relaxation coefficient

$P^{I+1} = P^I + \alpha\, \delta P, \quad S^{I+1} = S^I + \alpha\, \delta S \quad$ (and $S_w^{I+1} = 1 - S^{I+1}$)

$S^{I+1} = clipping(S^{I+1}, min = 0, max = 1)$ ▷ clip the saturation to $[0, 1]$

if convergence **then** ▷ Convergence test on the residual L_∞ norm

$P^{n+1} = P^{I+1}, \quad S^{n+1} = S^{I+1} \quad$ (and $S_w^{n+1} = 1 - S^{I+1}$)

break
else if I == **NMAXITER then** ▷ Max number of iterations reached
restart algorithm with $\Delta t^n = \Delta t^n/2$
end if
end for
End of Newton's iterations

3.6.1 YADS feature blocks

To build a reservoir simulation library, we require several fundamental building blocks. First, there's the 'Mesh' block, which handles tasks such as mesh creation, loading, and modifications. Next, we have the 'Physics' block, responsible for applying physical laws to the grid, including tasks like implementing relative permeability laws. Another crucial component is the 'Well' block, which manages well-related operations from creation and connection to the mesh to their management throughout the numerical simulation. Finally, once the mesh is defined, the wells are connected, and the physical laws are established, we can execute the reservoir simulation using the 'Numeric' block. In the following sections, we will delve into each of these blocks in greater detail.

3.6.1.1 Mesh block

The 'Mesh' block is built using the '*Mesh*' class, which serves as a representation of a mesh object. It encapsulates all of the mesh's properties and provides methods to explore the relationships between its components. Importantly, these relationships are dimensionless, meaning that the methods for querying them remain consistent regardless of the mesh's dimensionality (1D and 2D are supported). Let's now dive into the key features of this block.

Mesh creation

When create a Mesh object, there are several options available. One way is to use a construction method. In this case, the library supports the creation of one and two-dimensional Cartesian meshes using the method '*create_2D_cartesian*'. This method takes as input arguments the length of the mesh and the number of cells per dimension.

```
# create 1D cartesian mesh
# 10 000m x 1000m with 200 cells in x-direction and 1 in y-direction
grid = yads.mesh.two_D.create_2d_cartesian(
            Lx=10000, Ly=1000, Nx=200, Ny=1
            )
```

This code creates a 1D Mesh object called grid with 200 cells in the x-direction and 1 cell in the y-direction. The total length in the x-dimension

is $10,000$ meters and $1,000$ meters in the y-dimension.

Another way to create a mesh, it to load it from an external file. As this functionality has been developed to handle meshes coming from 'FreeFEM' software [7] (see FreeFEM.org). Therefore, it handles some of FreeFEM meshes export extensions such as '.mesh' and '.msh' and the resulting meshes are not necessarily cartesian. In this case, the associated method is '*load_meshfile* and only requires as argument the path to the mesh file in the correct extension. It is also possible to load Mesh using *load_json* method, this meshfile in JSON (JavaScript Object Notation) extension is obtained through a specific export method.

Mesh interface

With the mesh loaded, the next step is to interact with it, such as defining groups that will have specific properties or obtaining relational information between cells or faces. we will introduce some essential methods and provide an example of their utilisation within the library.

- *Mesh.measures*: get faces or cells lengths.
  Used to calculate transmissivities.
- *Mesh.face_to_cell*: given a face index, return one (boundary face) or two cells (inner face) indexes.
  Used to calculate fluxes
- *Mesh.add_cell_group_by_square*: Create a cell group composed of all cells within a defined square.
  Used to create permeability barriers
- *Mesh.connect_well*: connect a Well object to the Mesh object through perforated cell(s).

Mesh export

Exporting Mesh objects is essential, as it enables loading them into other software for visualization, for example. It also helps avoid the need for mesh creation during numerical experiments, which can be costly depending on the mesh size. To this extent, there are two methods: *Mesh.to_json* and *export_vtk*. *Mesh.to_json* allows the conversion of a Mesh object to a JSON file which is really useful as it can be seen as a dictionary that is supported in most of the coding languages. Moreover, it can be loaded back using the

load_json metod. Then the *export_vtk* method allows the export of the Mesh object to the VTK extension. This extension allows the Mesh to be visualized in some visualization software such as Paraview [1] (see Paraview.org).

3.6.1.2 Physics block

The physics block facilitates the computation of essential physical properties for numerical schemes. It comprises three primary methods: fractional flow computation, mobility computation, and relative permeability computation. Additionally, there are auxiliary methods that assist in calculating derivatives of these physical properties with respect to saturation, for example.

3.6.1.3 Well block

Wells are managed through the 'Well' class. The definition of a Well object necessitates the following properties:

- name
- cell or group of cell: position(s) or cell index(es)
- mode: injector or productor
- control: Pressure (Dirichlet) or Flow rate control (Neumann)
- Injection gas saturation: only works if the well is injector
- well radius
- schedule: opening and closing times

It is to note that the name is useful when connecting the well to the grid through the *Mesh.connect_well* method, as we will be able to work directly with the Mesh cell group associated to the well.

3.6.1.4 Numeric block

Now that the mesh is defined using a Mesh object, the physical laws are established, and the wells are defined and connected to the mesh, the 'Numeric' block encompasses several features for running the numerical simulation. It consists of two main packages and two auxiliary packages.

Among the auxiliary packages, there is a physics module available for computing numerical physical properties, such as transmissivity or well physical models like the Peaceman formula. The second auxiliary package is related to numerical tests associated with the numerical schemes. As we use an explicit saturation solver, which is conditionally stable, we need to compute a CFL condition to ensure the numerical stability of the scheme.

The two main packages are the Solver and the Scheme packages. The Solver package includes four primary features: an implicit pressure solver (IMP), explicit and implicit saturation solvers (ES/IMS), and a fully implicit solver. On the other hand, the Scheme package allows for running full numerical simulations using specific schemes, including IMPES, IMPIMS, and Fully Implicit. It manages the scheme and the succession of time steps.

Moreover, you can save simulations in JSON format. These saves can be performed either at the time step scale, meaning you save the reservoir state after each time step, or at the scheme scale. Saving Newton iterates provides valuable insights into the simulation behavior. Additionally, you have the option to save in VTK format at the time step scale for visualization in Paraview.

We now provide a synthetic code example that loads the 2D SHPCO_2 S size mesh from a JSON file, create some properties and groups from it, run a simulation using the fully implicit solver and save.

```
# Imports
import numpy as np
from yads.wells import Well
from yads.mesh.utils import load_json
from yads.numerics.schemes.solss import solss

# Load mesh
grid = load_json("SHP_CO2_2D_S.json")

# Boundary face groups creation
grid.add_face_group_by_line("injector_one",
(0., 0.), (0., 1000.))

# Permeability barrier zone creation
barrier_1 = grid.find_cells_inside_square((1000., 2000.),
   (1250., 0.))

# Diffusion coefficient (i.e Permeability)
```

```
K = np.full(grid.nb_cells, 100.0e-15)
permeability_barrier = 1.0e-15
K[barrier_1] = permeability_barrier
# Porosity
phi = np.full(grid.nb_cells, 0.2)
# gaz saturation initialization
S = np.full(grid.nb_cells, 0.0)
# Pressure initialization
P = np.full(grid.nb_cells, 100.0e5)
# viscosity
mu_w = 0.571e-3
mu_g = 0.0285e-3

# relative permeability law
kr_model = "quadratic"
# BOUNDARY CONDITIONS #
# Pressure
Pb = {"injector_one": 110.0e5, "right": 100.0e5}
# Saturation
Sb_d = {"injector_one": 0.0,  "right": 0.0}
Sb_n = {"injector_one": None, "right": None}
Sb_dict = {"Dirichlet": Sb_d, "Neumann": Sb_n}

# time step
dt = 2 * (60 * 60 * 24 * 365.25)  # in years
# maximum number of Newton iterations
max_newton_iter = 200
# stopping criterion
eps = 1e-6

# Well creation
well_co2 = Well(
    name="well co2",
    cell_group=np.array([[1475.0, 2225]]),
    radius=0.1,
    control={"Neumann": -(10 ** -3.1)},
    s_inj=1.0,
    schedule=[[2 * dt, 12 * dt],],
    mode="injector",
)

solss(
    grid=grid,
    P=P, S=S,
    Pb=Pb, Sb_dict=Sb_dict,
```

```
    phi=phi, K=K, mu_g=mu_g, mu_w=mu_w,
    dt_init=dt, total_sim_time=40 * dt,
    kr_model=kr_model,
    wells=[well_co2],
    max_newton_iter=max_newton_iter, eps=eps,
    save=True, save_step=1, save_path="shp_teaser_",
)
```

3.7 Numerical behaviour during well event

Well events generate pressure and saturation discontinuities (in time) inside the reservoir and lead to nonlinear convergence problems as they act as singular point sources that are tightly coupled to the reservoir model. In this section, we depict some typical behaviour induced by a well event.

We aim to assess the impact of a well event in a reservoir simulation through a simple 1D test case. In this case, we employ a reservoir that is 10 km long in the x-direction and 1 km long in the y-direction, with 200 cells in the x-direction and 1 cell in the y-direction, effectively creating a 1D model. We utilize the same physical configuration as outlined in the SHPCO_2 physical configuration 2.2.1, employing a drain zone permeability and a quadratic relative permeability law. Regarding the boundary conditions and well injection parameters, we applied Dirichlet boundary conditions for pressure, setting it to 110×10^5 Pa at the left boundary and 100×10^5 Pa at the right boundary. Similarly, we use Dirichlet boundary conditions for the saturation, with only water at both the left and right boundaries.

Thereafter, the well is located in the middle of the domain and we control the gas injection in pressure with a constant injection pressure of 200×10^5 Pa. The simulation runs for 60 years with a time step of 2 years, we start injecting at 4 years and stop injecting at 24 years. Finally, we set the maximum number of Newton iterations at 200, and the stopping criterion at 1×10^{-6}.

In figure 3.1, we present the reservoir state just before the well opening and right after. We notice a sudden global pressure discontinuity caused by the well, this discontinuity is instantaneous and remains independent of the time step. Furthermore, we observe a saturation discontinuity in the vicinity of the well, and this latter discontinuity depends on the time step.

In figure 3.2, the evolution of the required number of Newton iterations to

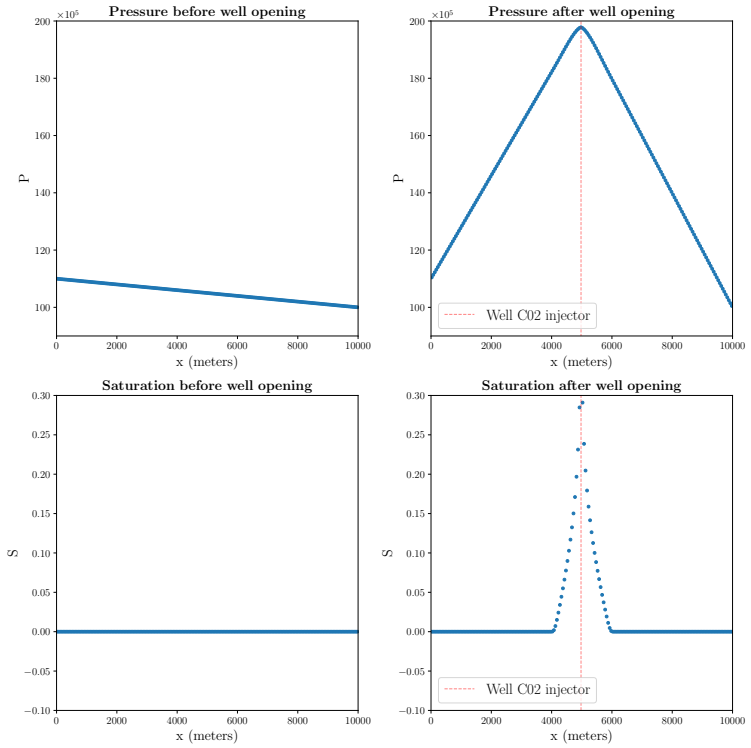


Figure 3.1: Example of well event impact on the pressure and saturation profile. The left up and down figures represent the pressure and saturation before a well opening, while the up and down right figures represent the pressure and saturation after a well opening.

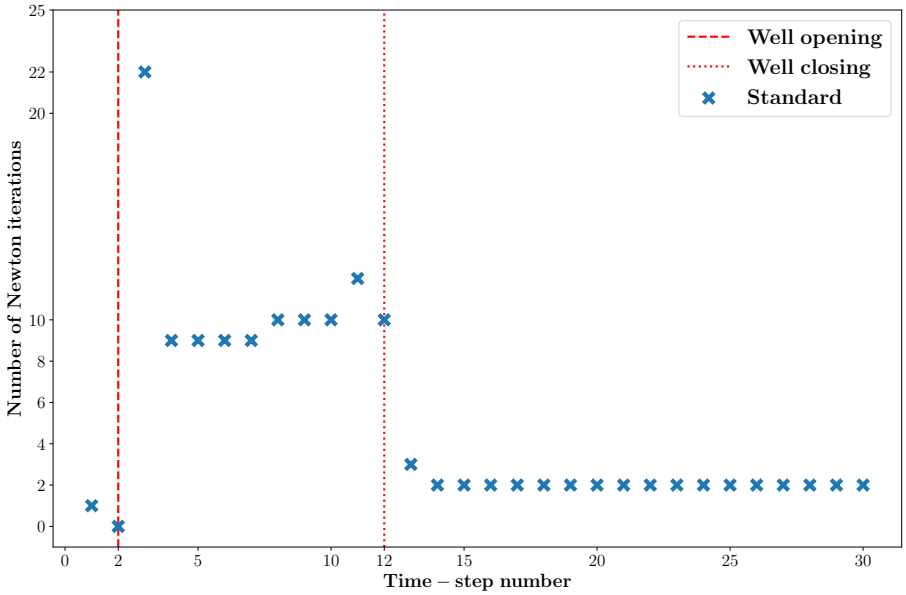


Figure 3.2: Example of scenario and it's impact on Newton's method. The scenario is composed of a well event opened after 2 time steps (red dashed line) and closed at 12 time steps (red dotted line).

achieve convergence at each time-step is depicted. We observe that immediately after the well opening, the number of required Newton iterations to reach convergence peaks at 22 and remains consistently above 9 throughout the entire well injection period. Conversely, when the well is closed, no more than 3 iterations are needed to achieve converge. Consequently, it is evident that the well event contributes significantly to the overall count of Newton iterations over the entire simulation duration. This behavior can be attributed to the sensitivity of Newton's method to initialization. Since the solution from the previous time step serves as the initial guess for the next time step, a substantial deviation between the new and previous solutions may necessitate a larger number of iterations.

Bibliography

[1] James Paul Ahrens, Berk Geveci, and C. Charles Law. Paraview: An end-user tool for large-data visualization. In *The Visualization Handbook*, 2005.

[2] Khalid Aziz and Antonín Settari. Petroleum reservoir simulation. 1979.

[3] K.H. Coats. A note on impes and some impes-based simulation models. *SPE Journal - SPE J*, 5:245–251, 09 2000.

[4] Robert Eymard, Gallouet Thierry, and Raphaèle Herbin. Finite volume methods. 7, 12 2000.

[5] Franck Ouaki. *Etude de schémas multi-échelles pour la simulation de réservoir*. PhD thesis, Ecole Polytechnique X, 2013. NNT: . pastel-00922783.

[6] D.W. Peaceman. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation(includes associated paper 6988). *Society of Petroleum Engineers Journal*, 18(03):183–194, 06 1978.

[7] Georges Sadaka. Freefem++, a tool to solve pdes numerically, 2012.

[8] John W. Sheldon and W. T. Jr. Cardwell. One-dimensional, incompressible, noncapillary, two-phase fluid flow in a porous medium. *Transactions of the AIME*, 216:290–296, 1959.

Chapter 4

Black-Box machine learning PDE solver

Contents

4.1 Problems considered **88**
4.1.1 Poisson equation 88
4.1.2 Incompressible two-phase flow problem 90
4.2 Conclusion **98**

In this section, we evaluate whether the learning models have the potential to serve as direct substitutes for solvers, or if they are more suited to augmenting traditional solver approaches. We consider parameter to solution problems, all adapted to some geosciences applications. This work results from the CEMRACS 2023 on Scientific Machine Learning. During this event, we developed more test cases (that are not presented in this thesis) where Physics Informed Neural Network are also part of the benchmark. For the first problem, we propose to compare the performances of different methods between the following:

- Proper orthogonal decomposition (POD)
- Multi-layer perceptron (MLP)
- Deep Operator Network (DeepONet)

- Fourier Neural Operator (FNO)

MLP, FNO and DeepONet have been presented in the state of the art. We expose a short introduction here on POD.

Proper orthogonal decomposition

Proper Orthogonal Decomposition (POD) [1] [3] is a mathematical technique widely used in the field of fluid dynamics and computational mechanics to analyze and extract information from complex data sets. It is composed of two parts: the Truncated Singular Value Decomposition (SVD) which serves as a tool for reducing the dimensionality of large datasets, enabling efficient data representation and analysis through the creation of POD basis.

Mathematically, SVD operates by decomposing a dataset into a set of orthogonal basis functions or modes, often represented as eigenfunctions. These modes are determined by capturing the most significant variations within the data. They are organized in order of importance, with the first few modes explaining the majority of the variance in the dataset. This ordered set of modes allows for a concise representation of the data, shedding light on its underlying structure and dynamics.

Given a collection of snapshot as a data matrix, the singular value decomposition (SVD) factorizes the data matrix into a product of three matrices: the left singular vectors, the singular values, and the right singular vectors. In the context of fluid dynamics, the leading singular vectors correspond to the primary POD modes, which effectively capture the dominant flow patterns or structures present in the data.

After obtaining the POD basis through the SVD, a Galerkin projection is applied to infer new data. Galerkin projection is a technique that allows us to represent the dynamics of a high-dimensional system using a reduced set of modes or basis functions. In the context of POD, Galerkin projection involves approximating the governing PDEs or system dynamics using the POD basis functions. The Galerkin projection provides a computationally efficient way to study and simulate the system's dynamics while significantly reducing the computational cost associated with high-dimensional problems.

4.1 Problems considered

4.1.1 Poisson equation

4.1.1.1 Problem presentation

For the first case, we consider the following elliptic equation.

$$\begin{cases} -\nabla \cdot (D(x)\,\nabla u) = \delta(x,y), \qquad x \in (0,1), \\ \qquad\qquad u(0) = u(1) = 0. \end{cases} \tag{4.1}$$

Where D is the diffusion coefficient function and $\delta(x,y)$ can be interpreted as a localised well injection flow rate defined by:

$$\delta(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

We use a 1D cartesian grid composed of 101 points. To obtain non-zero values for the source term delta, y must coincide with the grid points, we therefore will have to train at a low data regime i.e with few data. We consider a mapping from the parameters (D, y) to the solution u. We uniformly sample 61 points over the domain $[0.2, 0.8]$, for each point y we generate a D map over the whole domain using the following function:

$$D(x) = 0.1 + exp(-\frac{(x-y)^2}{0.02}). \tag{4.3}$$

We therefore obtain Gaussian D maps centered on y, a standard deviation of 0.1 and with minimum values of 0.1. It is to note that D is thus defined by only one parameter y. Also we chose the well location at the maximum diffusion location, this corresponds to the fact that the injection is realised in a high permeability field corresponding to a reservoir for example. The main issue is that we reduce the input variability by doing so. We use an implicit finite difference solver to generate the solutions and show on figure 4.1 some examples of inputs parameters and their corresponding solutions.

We then randomly split in train/validation sets with a 80/20 ratio. Finally, we fit four models: POD, MLP, FNO and DeepONet.

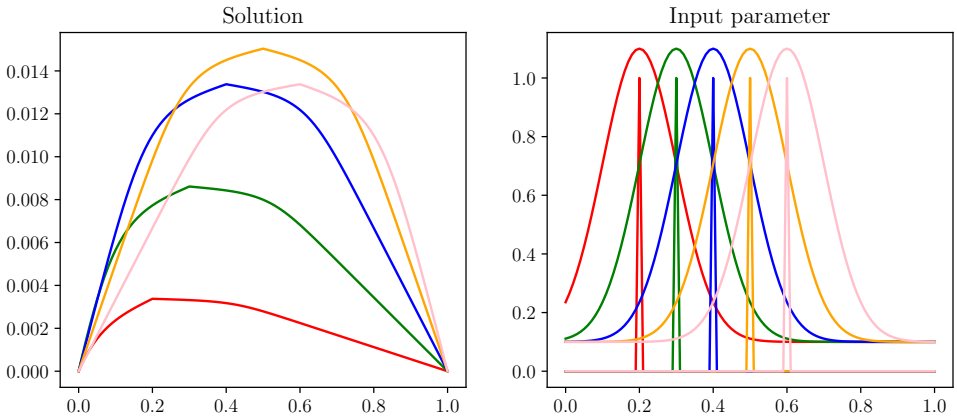


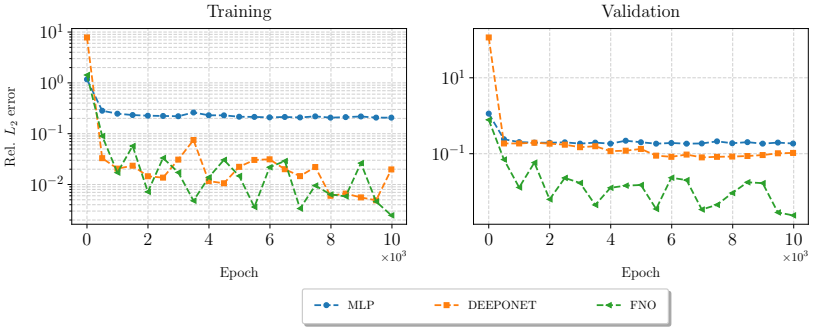
Figure 4.1: Right figure: input parameters D maps and source δ maps for different positions and their corresponding solution on the left figure.

4.1.1.2 Results

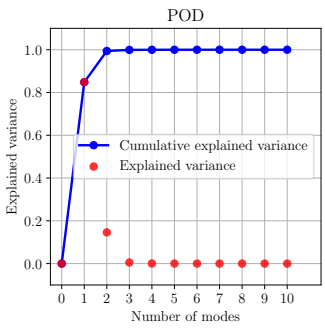
We show on figure 4.2 the results of the fitting process for each method. We use Optuna [2] (see Optuna.org) for the hyperparameter optimization. We observe on 4.2b that two modes is enough to reach 99% of cumulative explained variance. Moreover, we observe on 4.2a that the neural operators outperform the MLP for training and validation by reaching a lower minimum relative L_2 error. Indeed, neural operators are constructed by design to learn mappings from functions to functions. However, we observe that DeepONet has a higher validation relative L_2 error than the FNO meaning that it overfits, this is mainly due to the low data regime (40 for training, 21 for validation).

To assess the potential of each model as a direct substitute to the solver, we use parity plots on in distribution (ID) and out of distribution (OOD) data. Parity plots consist in plotting the norm of the prediction $\hat{u}_D$ versus the norm of the solution u_D for each sample of a dataset. This helps us visualize the performances of the predictive model over a whole dataset. An accurate predictive model results visually in a parity plot corresponding to the identity. The ID data is the concatenation of train and validation sets while the OOD dataset is composed of grid values in $[0.1, 0.2[\cup]0.8, 0.9]$.

Figure 4.3 shows that all models except MLP performs well in distribution according to the L_2 norm which is in agreement with the fitting process



(a) Evolution of relative L_2 error for training and validation through epochs for MLP, DeepONet and FNO.



(b) Scree Plot: Eigenvalue Distribution for Component Analysis.

Figure 4.2: Results of the different fitting process for each method. Left figure shows the evolution of relative $L2$ error through the fitting process i.e training and right figure is the scree plot of the POD.

curves. Indeed, a correct prediction should match the solution on the $x = y$ red dotted line. Moreover, the out of distribution prediction norms do not match the solution norms for FNO, DeepONet and MLP models meaning that the models do not generalize.

Considering the replacement of the solver with one of these models, it's crucial to acknowledge this significant factor. Substantial effort is necessary to accurately sample the input data. This ensures that when the model takes over the solver's role, there won't be any out-of-distribution points since we are aware of the models' limited generalization capabilities.

4.1.2 Incompressible two-phase flow problem

We now consider a 'piston' reservoir problem. We replace the traditional numerical solver by a black-box machine learning model. However, compared to the previous problem, there is now a time-dependency. Therefore, we will iterate through time using the machine learning model.

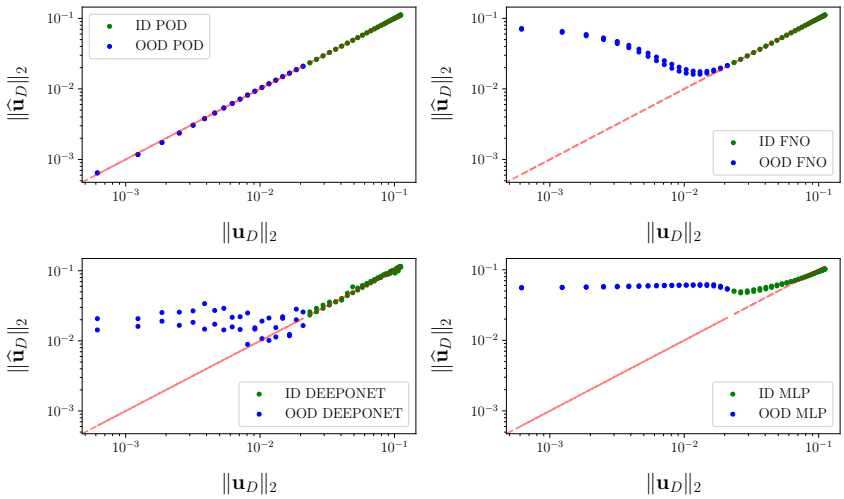


Figure 4.3: In and out of distribution parity plots using L_2 norm for each method between POD (upper left), FNO (upper right), DeepONet (lower left) and MLP (lower right). The red dotted line corresponds to the identity. The x-axis corresponds to the finite difference solution norm while the y-axis corresponds to the prediction norm.

4.1.2.1 Problem presentation

We consider a 1D reservoir with only water inside. It is composed of 100 cells in the x-direction representing 10 kilometers and 1 kilometer in the y-direction. The porosity and the permeability are constant over the whole reservoir and equal to respectively 0.2 and 300×10^{-15}. The water and gas viscosity are as mentioned in the 2.2 section. We use a quadratic relative permeability model. Regarding the boundary conditions, we inject gas at the left boundary with a certain pressure, at the right boundary, the pressure is equal to 10 MPa and and the initial conditions correspond to a reservoir with only water.

Data generation

We generate 10000 data snapshots using a fully implicit solver with a constant time-step of 1 year. We start at $t = 0$ with an initial pressure equal to 10 MPa and we sample a left boundary pressure between 10.5 MPa and 20 MPa. Then we iterate in time for a total of 101 iterations. At each iteration, we save the reservoir state as pressure and saturation snapshots. Then, we remove the

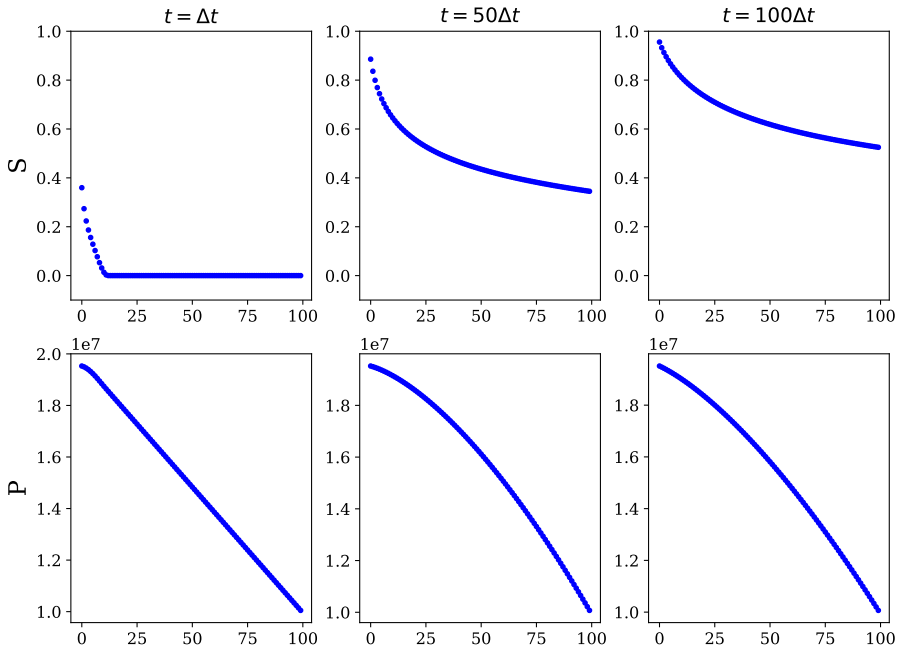


Figure 4.4: Example of saturation and pressure solutions at different times using a left boundary pressure of 19.5×10^6 Pa.

first snapshot as it has been initialized with a pressure that is not usable for training. We repeat this process 100 times. This results in 10 000 snapshots, each composed of a pressure and a saturation map. Finally, we split the whole dataset in three sets: train, test and validation with a 60/20/20 ratio by batch coming from the same simulation (i.e 100 per 100).

Model training

We use the Fourier Neural Operator architecture with 12 parameterized modes and 128 channels. The input features are the pressure and saturation snapshots, together with a constant map of boundary pressure, therefore a single input data shape is $(1, 100, 3)$. The predicted features are the pressure and saturation after a time-step, therefore the output shape for a single sample is $(1, 100, 2)$. We train on a A100 GPU using Adam optimizer, a learning rate of 1×10^{-4}, a momentum of 0.9 during 2500 epochs and keep the model parameters corresponding the the lowest test value. We show on the figure 4.5 the loss curve. The minimum loss value is 1.3×10^{-3} and is reached at

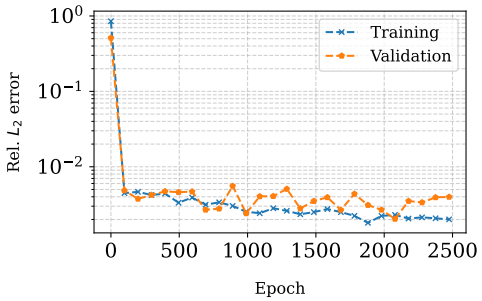


Figure 4.5: Evolution of relative L_2 error for training and validation through epochs.

epoch 2427. It's corresponding train loss value is 2.2×10^{-3}.

4.1.2.2 Results

We present the results obtained by the machine learning model, first when used for single sample predictions, then as a black-box simulator which iterates through time.

Single sample

We show on the figure 4.6 three snapshots from the training dataset of pressure and saturation at different times: at the very beginning of the simulation ($t = \Delta t$), at the mid-time of the simulation ($t = 50\Delta t$) and at the end ($t = 100\Delta t$). The left boundary pressure is 14.1×10^6 Pa. In blue dots we show the exact solutions for each snapshot while the red dashed line shows the corresponding predictions using the machine learning model. We observe that the predictions fit qualitatively well the exact solutions.

Parity plot

We show on figure 4.7 the parity plots of the predicted versus exact saturation (left figure) and pressure (right figure) using the L_2 norm for the three datasets. We observe that, according to the L_2 norm, the predictions for all data sets are accurate as each point is on the identity line.

Therefore, given a snapshot coming from a simulator, it seems possible to predict accurately the solution after a step of time.

Time iteration

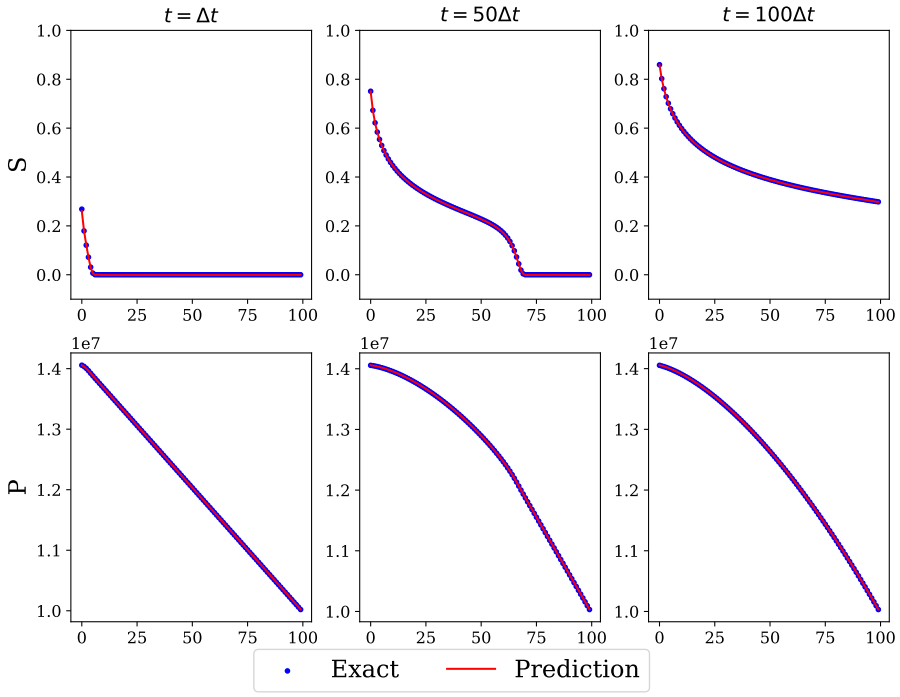


Figure 4.6: Example of exact and predicted saturation and pressure at different times with a left boundary pressure of 14.1×10^6 Pa.

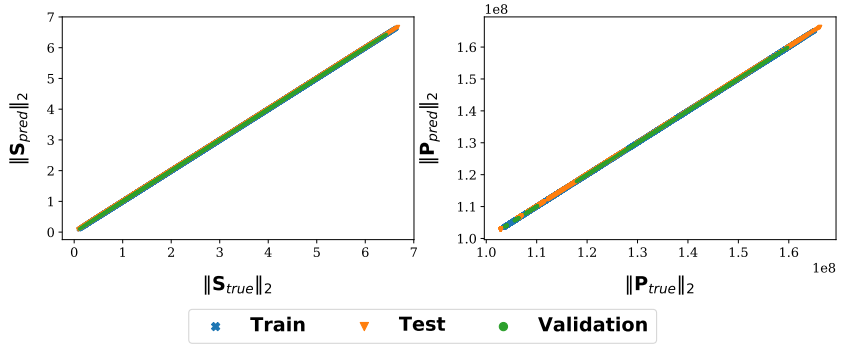


Figure 4.7: Parity plot of the predicted versus exact saturation (left figure) and pressure (right figure) using the L_2 norm for the three datasets.

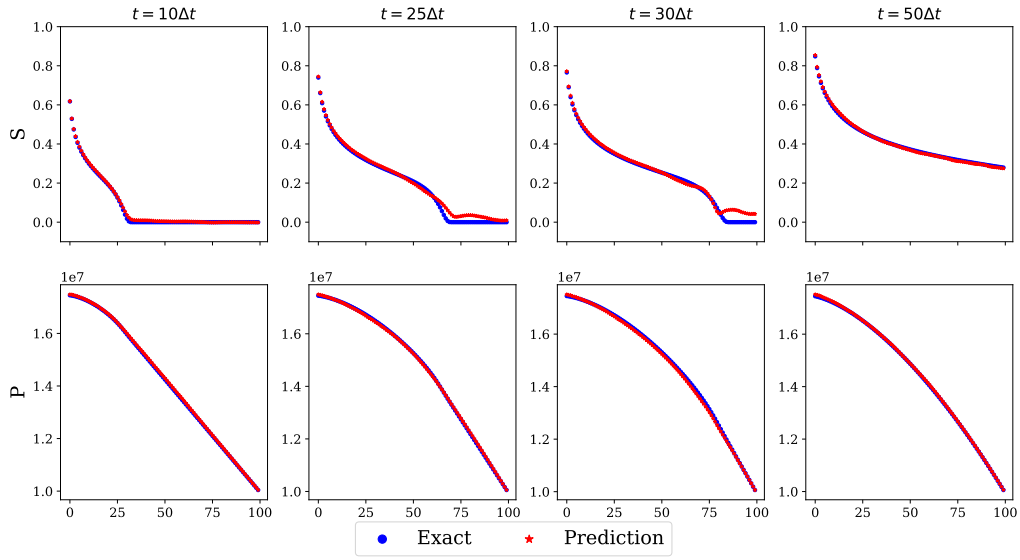


Figure 4.8: Example of saturation and pressure snapshots obtained using the machine-learning model to iterate through time starting from a time $t = 0$ and with a left boundary pressure of 17.5×10^6 Pa.

Now, given a snapshot, we iterate through time using only the machine-learning model i.e, we consecutively apply the model on it's own predictions. The figure 4.8 exposes different snapshots obtained starting from $t = 0$ and a left boundary pressure of 17.5×10^6 Pa. We observe that pressure remains close the solution for all snapshots while the saturation remains accurate everywhere except at the front of saturation. The figure 4.9 exposes different snapshots obtained starting from $t = 0$ and a left boundary pressure of 14.1×10^6 Pa. We observe that after 10 time-steps, the predicted saturation starts to drift away from the solution at the bottom of the saturation front. After a few more iterations, the predicted pressure and saturations have drifted away from the solution on the whole domain. Therefore, a small error at a specific location may lead to strongly non-physical solutions after only a few iterations.

A straightforward engineering bootstrap of the case presented on the figure 4.9 is to incorporate, such as in Newton's method, some clipping. Indeed, we observe that the predicted saturation is negative at the bottom of the saturation front after $t = 10\Delta t$. To prevent it, we set all negative values to 0

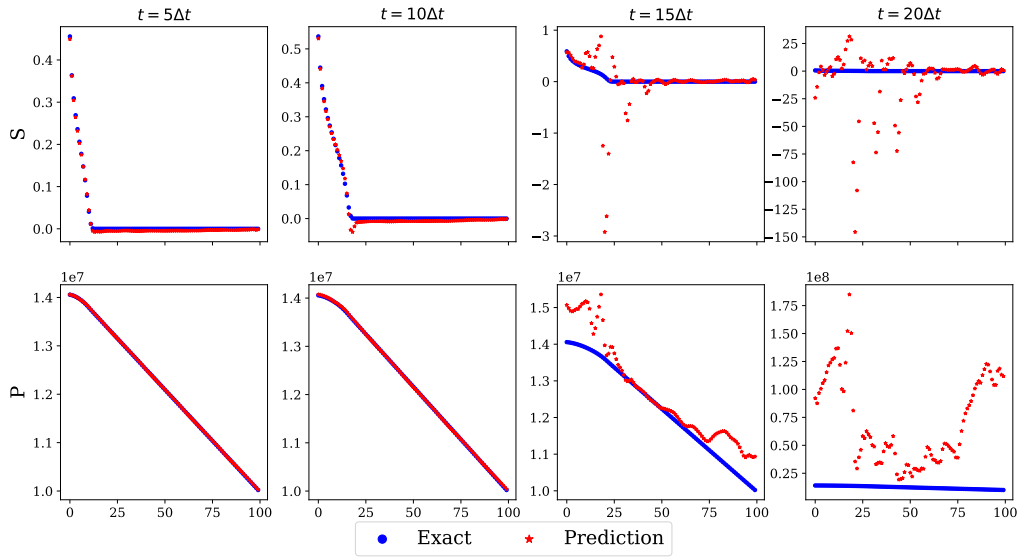


Figure 4.9: Example of saturation and pressure snapshots obtained using the machine-learning model to iterate through time starting from a time $t = 0$ and with a left boundary pressure of 14.1×10^6 Pa.

and all values superior to 1 to 1. We show the same test case on the figure 4.10 and we observe that the clipping results in a much better quality prediction even after multiple step of times. It is to note that this trick would not work for the case presented on figure 4.8 as the saturation respects the maximum principles at all times. Moreover, it is also possible to apply clipping on the pressure if needed.

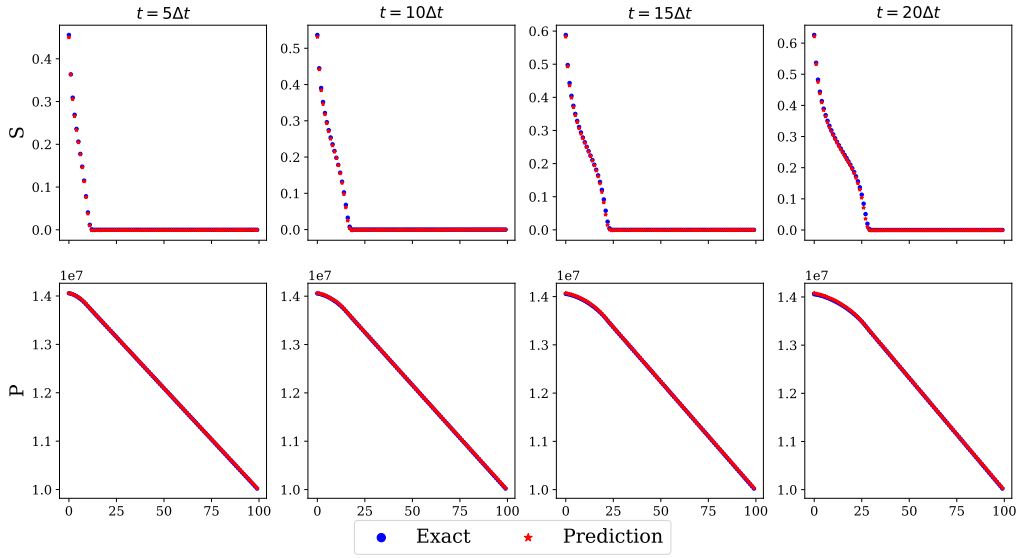


Figure 4.10: Example of saturation and pressure snapshots obtained using the machine-learning model and saturation clipping to iterate through time starting from a time $t = 0$ and with a left boundary pressure of 14.1×10^6 Pa. The predictions are represented in red dots while the solutions are in blue dots.

4.2 Conclusion

This study highlights the potential of machine-learning models to accurately predict solutions of PDE problems. We observed that given snapshots generated by a traditional numerical solver, it is possible to learn the solution after a certain step of time with different boundary conditions. However, it is not straightforward to replace entirely the solver by the machine-learning model. Indeed, a small error can make the predicted solution drift away from the true solution in a few time iterations. This highlights the fact that it is important to take into account numerical guarantees that the traditional solver ensures. Therefore, we propose in this thesis to use machine-learning models as a complement to accelerate the traditional numerical solver while preserving it's guarantees.

Bibliography

[1] *Volume 1 System- and Data-Driven Methods and Algorithms*. De Gruyter, Berlin, Boston, 2021.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.

[3] Peter Benner, Mario Ohlberger, Albert Cohen, and Karen Willcox. *Model reduction and approximation: theory and algorithms*. SIAM, 2017.

Chapter 5

Hybrid Newton Method - Global approach

Contents

5.1 Introduction **101**
5.2 Methodology **103**
5.2.1 Hybrid Newton algorithm 103
5.3 Initial guess construction **103**
5.3.1 Neural Network architecture 105
5.4 Test case and Database Generation **106**
5.4.1 SHPCO2 benchmark 106
5.4.2 Test cases 108
5.5 Results and discussion **111**
5.5.1 Neural Network training 111
5.5.2 Results 113
5.5.3 Discussion 116
5.6 Conclusion **117**

5.1 Introduction

In this chapter, we complement the traditional numerical solver by predicting a global initialization of Newton's method. The continuous model (2.9) is discretized using a two-points finite volume scheme and a Euler type time integration. The pressure P and saturation S are solved simultaneously through a fully implicit scheme (3.11). Finally, the resulting non-linear system of equations is solved using Newton's method. This last is initialized using the solution obtained at the previous step X^n as an initial guess and returns the solution at the next step X^{n+1}.

Time-step management

In theory, the time-step size of a fully implicit reservoir simulator is not limited by stability (i.e unconditionally stable). However, in practice, when using the standard Newton's method, convergence may fail for larger time-step sizes. This necessitates multiple time-step reductions to achieve convergence, resulting in a significant number of superfluous iterations.

At the scale of a single time-step (in opposition with the global simulation scale), the time-step is driven by Newton's method. One can allow a maximum number of Newton iterations N_{max} under which the method must converge. Above this number, we start over with a smaller time-step, usually by dividing by a factor two. There are other time-step management mechanisms at the global simulation scale such as the management through the solution variations. This mechanism presents multiple empirical parameters to set without much justifying.

But these global simulation time-step management mechanisms do not concern us in this study since we focus on the acceleration of the Newton method, we work at the single time step scale and not at the global simulation scale.

Impact of well events during the numerical resolution

Well events generate pressure and saturation discontinuities (in time) inside the reservoir and lead to nonlinear convergence problems as they act as singular point sources that are tightly coupled to the reservoir model [1]. These discontinuities often prevent Newton's method from converging while attempting to solve the system as the initial guess may be far from the solution. In such situations, we attempt to solve the system again with

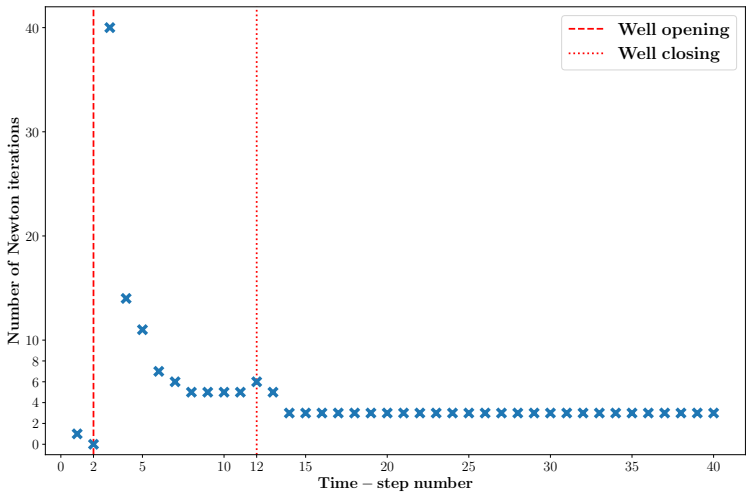


Figure 5.1: Example of scenario and it's impact on Newton's method. The scenario is composed of a well event opened after 2 time steps (red dashed line) and closed at 12 time steps (red dotted line).

a reduced time step and continue this process until convergence is achieved. Well events can thus account for a significant portion of the actual simulation time. An illustrative scenario, inspired by the test case 2 discussed in Section 5.4, is depicted in Figure 5.1. This scenario involves a simulation of 40 time steps, each spanning 2 years. At 4 years, an injection well is opened using an well injection flow rate of $10^{-3.1}$ and closed at 24 years. We observe that immediately following the well opening, Newton's method necessitates 40 iterations to reach convergence, with a minimum of 5 iterations required at each subsequent step throughout the entire well opening process. In contrast, in the absence of well events, the maximum number of Newton iterations is 5 immediately after well closure and 3 otherwise. Consequently, the well event-induced Newton iterations constitute the majority of the total Newton iterations throughout the entire scenario, accounting for 54%. Furthermore, if the maximum allowable iteration count were less than 40, the time step would be halved.

5.2 Methodology

We propose a methodology based on the Hybrid Newton algorithm [8] [5] [11] [2] to alleviate the impact of well events. It consists in predicting via machine learning an initialization closer to the solution than the standard initialization, hence accelerating the Newton's method convergence while still guaranteeing the correct solver solution. We use a fixed reservoir geometry and configuration where a well event occurs at a specific location. We aim to alleviate the impact of the well opening for a wide range of scenarios.

5.2.1 Hybrid Newton algorithm

The hybrid Newton algorithm is a modification of the standard Newton algorithm with respect to the initialization. Indeed, the hybrid method proposes to initialize with a more accurate solution instead of using the standard initialization (i.e. the previous time step solution). By using an initialization closer to the solution, we aim to converge in fewer iterations.

In this article, our primary focus is on the hybrid guess construction X_{ML}^{n+1} and its performance compared to the standard initial guess X_{init}^{n+1}. We will compare them through their impact on the number of Newton iterations.

5.3 Initial guess construction

In the hybrid Newton algorithm, a prediction guess $X_{ML}^{n+1} := (P_{pred}, S_{pred})$ is required. We propose the following construction for the pressure and saturation:

1. P_{pred}: Implicit Pressure solver
2. S_{pred}: Prediction using supervised learning

It is to note that, in this context, it is not possible to use an interpolation from previous time-step solutions to construct an initial guess.

5.3.0.1 Pressure

During a well event, the pressure discontinuity is global in the reservoir and instantaneous in time. We propose to use the solution of an implicit pressure solver P_{imp} (IMP) [9] at the step $n+1$ as a prediction guess P_{pred}. The implicit

pressure solver solves the linear elliptic equation (5.1) and catches the main global variations of pressure but does not catch the small local variations of pressure due to saturation variations. Moreover, the implicit pressure solver only requires to solve a linear system. Therefore, we consider that it is not worth it to train a machine learning model as we have a good approximation. The implicit pressure solver solves the following equation:

$$div(v(P_{imp}, S^n)) = q_T, \tag{5.1}$$

with $v = v_g + v_w$, v, v_g and v_w are respectively the total velocity, the gas velocity and the water velocity inside the reservoir. $q_T = q_g + q_w$, q_T, q_g q_w are respectively the total, gas and water well injection flow rates. In practice, only gas is injected, therefore $q_T = q_g$.

5.3.0.2 Saturation

During a well event, the saturation discontinuity inside the reservoir is local and near to the well. One could use an implicit saturation solver (IMS) which would require a Newton's method to solve a non linear system. Therefore, we propose to predict the saturation using a neural network considering that the inference of a neural network is rather fast compared to the multiples Newton iterations of the IMS solver which each require to solve a linear system. The predicted saturation is denoted S_{ML}.

The standard methodology uses the initial guess $X_{init}^{n+1} = X_{sol}^n = (P^n, S^n)$ and the hybrid methodology uses the initial guess $X_{ML}^{n+1} = (P_{imp}, S_{ML})$. Given that we aim to conduct a fair assessment of the impact of a saturation predictive model, we compare the hybrid initialization with a reference initialization. Therefore we use in this article the following initial guess for the reference methodology $X_{init}^{n+1} = (P_{imp}, S^n)$.

In the end, we will compare in terms of Newton iterations the two following initialization:

1. Reference initialization: $X_{init}^{n+1} = (P_{imp}, S^n)$
2. Hybrid initialization: $X_{ML}^{n+1} = (P_{imp}, S_{ML})$

5.3.1 Neural Network architecture

The objective is to predict using reservoir and well information the global saturation state reached after a well event. A well event at a specific location and a specific well geometry can be described with few parameters: an injection well flow q_g and a time-step dt. A neural network architecture with good predictive capability for different physics-based processes is required.

Fourier Neural Operator

Neural networks are commonly used to learn relationships between finite-dimensional spaces, but they can struggle to adapt to changes in governing equations or conditions [3]. The Fourier Neural Operator (FNO) [7] addresses this issue by learning relationships between infinite-dimensional spaces using data-driven methods. This allows the FNO to understand the rules governing an entire family of partial differential equations. Additionally, the FNO improves computational efficiency by converting convolution operations in neural networks to multiplication through the use of discrete Fourier transforms.

Selected Architecture

We use the architecture presented on figure 5.2 based on an uplifting dense layer, four Fourier Layers (see figure 5.3) and two dense layers. We use the Gaussian Error Linear Units (GELU) [4] as activation function for each layer. We use two different versions of this general architecture for the training on the two test cases. **Nc** and **Ni** are parameters respectively representing the number of channels and the number of inputs.

The architecture of a Fourier layer fig. 5.3 is composed of two parts, one that apply Fourier transform, a linear transform on the lower Fourier modes and a filter on the higher mode, then it applies the inverse Fourier transform. The other part is composed of a local linear transformation applied to the original input. Finally, the output of the two parts are added together and an activation function is applied.

Input Features

There are multiple features that can be considered as input features for the selected neural architecture. We select four possible input features, the time-step dt, the injection flow rate q_g, the initial saturation S^n and the implicit

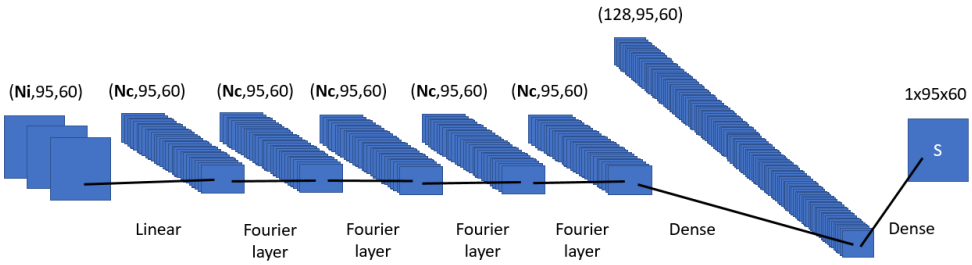


Figure 5.2: Selected neural network architecture composed by an uplifting dense layer, four Fourier layers, a dense layer and a projection dense layer. **Nc** and **Ni** are parameters respectively representing the number of channels and the number of inputs.

pressure P_{imp}. As S^n and P_{imp} have the same shape as the output feature, they can be used straightforward. However, as q_g and dt are scalars, they need to be reshaped. We propose to reshape dt into a constant map of value dt everywhere. For the injection flow rate q_g, we reshape it to a map of value zero everywhere except at the well location where it takes the value q_g.

5.4 Test case and Database Generation

5.4.1 SHPCO2 benchmark

As a practical use case, we use the SHPCO_2 benchmark presented in 2.2 to test our methods.

Reservoir configuration

We adapt the original 2D SHPCO_2 benchmark by removing the gas zone and replacing it by a well at its center.

The domain after modification is separated in two zones, the first zone called "Barrier zone" is coloured in green on the figure 5.5 and the second zone called "Drain zone" is formed of the rest of the domain. These two domains have different petrophysical properties.

Petrophysical properties

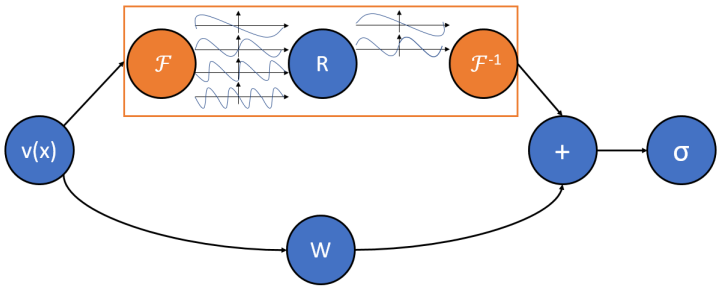


Figure 5.3: The Fourier layer starts with an input vector v, applies the Fourier transform $\mathcal{F}$ to it, then performs a linear transformation R on the lower Fourier modes while filtering out the higher modes. The inverse Fourier transform $\mathcal{F}^{-1}$ is then applied. A local linear transformation W is applied to the original input vector v. The output of the top and bottom operations are then added together and an activation function is applied.

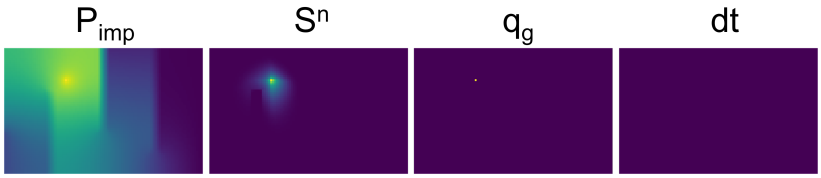


Figure 5.4: Qualitative view of Neural network input feature possible assembly.

Phase properties

Relative permeability model

We use a quadratic relative permeability model:

$$kr(S) = S^2 \qquad \text{and} \qquad kr(S_w) = (1 - S)^2.$$

Boundary Conditions

The boundary conditions of the adapted SHPCO_2 (figure 5.5) are presented in the following table 5.3:

It is to note that the Productor has the composition of what it produces. We consider the following mesh geometry which corresponds to the small (S) mesh size of the SHPCO_2 benchmark.

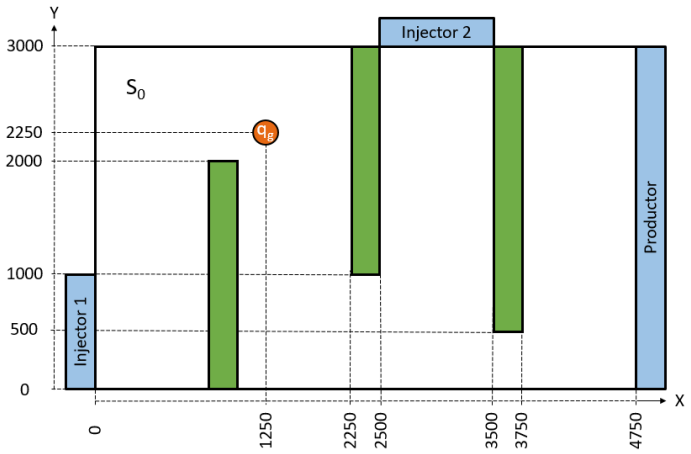


Figure 5.5: Adapted 2D SHPCO_2 case geometry

	Barrier zone	Drain zone
Porosity [-]	0.2	0.2
Permeability[m^2]	1.e-15	100.e-15

Table 5.1: Petrophysical Parameters

Well conditions

We detail the well conditions in the following table 5.5

5.4.2 Test cases

We propose two different test cases based on the SHPCO_2 benchmark and it's reservoir configuration presented in 5.4.1. The test case 1 has constant initial saturation $S0$ while the test case 2 has more realistic initial saturations.

5.4.2.1 Test case 1

Database Generation

We launch simulations with a constant reservoir configuration except for three parameters, $S0$ the initial saturation, q_g the well injection flow rate and dt the time-step. We allow a maximum of 200 Newton iterations to converge. The convergence criterion is based on the residual norm and we

	Gas phase	Water phase
Viscosity [Pa.s]	0.0285e-03	0.571e-03

Table 5.2: Physical properties of fluids

	Pressure[Pa]	Composition
Injector 1	110.e+05	Water
Injector 2	105.e+05	Water
Productor	100.e+05	-

Table 5.3: Boundary condition parameters

iterate till $dt\|R\|_\infty/V \leq \epsilon$ with $\epsilon = 1e{-}6$, V the cell volume (constant in our case) and R the residual of the physical system. The residual R is detailed in 3.4

We generate 5004 parameter combinations through a Latin Hypercube Sampling strategy [10] within the following ranges: $S0 \in [0, 0.6]$, $|q_g| \in [10^{-5}, 10^{-3}]m^2/s$ which corresponds to a well pressure $\in]10, 20]$ MPa and $dt \in [0.1, 10]$ years. Note that P_{imp} is obtained using the Implicit Pressure solver (IMP).

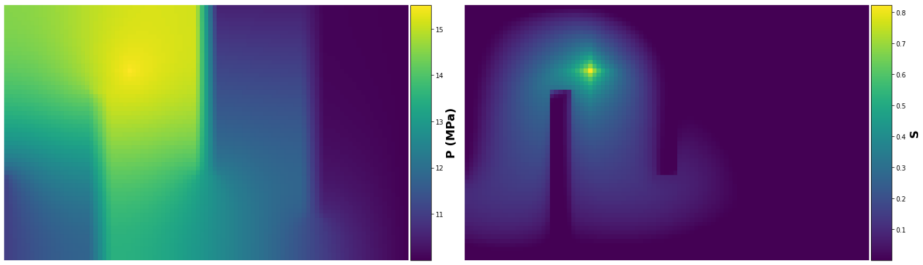


Figure 5.6: Test case 1 example of reservoir Pressure (left) and Saturation (right) obtained after a time-step with $S0 = 3.81{\times}10^{-4}$, $q_g = 7.61{\times}10^{-4}m^2/s$ and $dt = 2.4 \times 10^{+8}s$.

5.4.2.2 Test case 2

In the test case 1, we use constant reservoir saturation maps as initial saturation S0 and realised one well opening with a particular well injection flow rate q_g and time-step dt. In the test case 2, we use more realistic initial

Mesh	Dx [m]	Dy [m]	Nx	Ny	NCell
S	50	50	95	60	5700

Table 5.4: 2D mesh parameters

Injection parameters	Flow rate $[m^2/s]$	Composition	well radius $[m]$	Opening period $[s]$
CO_2 injector	q_g	Gas (S = 1.)	0.1	dt

Table 5.5: CO_2 injection well conditions

saturation maps. To do so, we realise N consecutive well opening and closure events (see figure 5.7. We use Latin Hypercube Sampling strategy to generate parameter combinations. The initial parameters are q_g, dt and S^n. After the first simulation, we close the well (i.e $q_g = 0m^2/s$) and launch another simulation using the previous reservoir state obtained and a new time-step dt (sampled through Latin Hypercube Sampling). Finally we open the well and launch a simulation with q_g and dt as parameters. The opening and closure step are repeated as many times as needed (see figure 5.7). The reservoir state is saved at every well opening or closure.

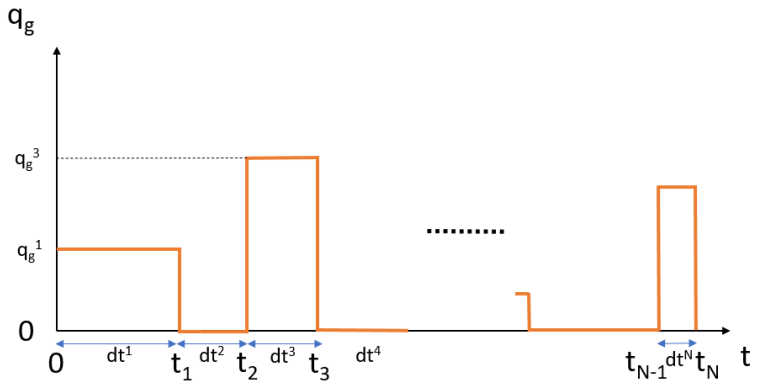


Figure 5.7: Test case 2 workflow with multiple well openings and closures. A step of time with a null well flow is realised between each closure and opening.

We generate data with $N = 9$ (i.e 5 well openings and 4 closures) and 3600 parameter combinations for each step. The parameters are sampled using a

Latin Hypercube Sampling strategy within the following ranges: $S0 \in [0, 0.6]$, $|q_g| \in [1 \times 10^{-5}, 1 \times 10^{-3}]$ m^2/s and $dt \in [1, 10]$ years in seconds.

We perform simulations for each parameter combination and we only consider data where the well injected flow rate q_g is not null. For $N = 9$, we have 5 wells openings (i.e $q_g > 0$) and 3600 parameter combinations for each one. Therefore, there is a total of 18000 samples. When splitting the samples in train and test sets, data coming from a same scenario are sent together in a set (i.e 5 per 5 for $N = 9$).

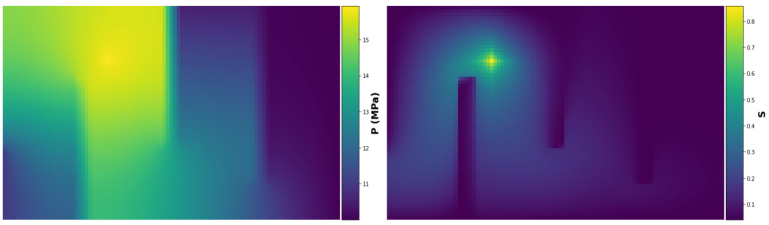


Figure 5.8: Example of reservoir Pressure (left) and Saturation (right) obtained after a time-step with $q_g = 9.8 \times 10^{-4}$ m^2/s and $dt = 3.1 \times 10^{+8} s$. The simulation required 12 Newton iterations to converge.

5.5 Results and discussion

5.5.1 Neural Network training

5.5.1.1 Test case 1

We use the Neural Network architecture presented in figure 5.2 with $Nc = 32$ channels in the Fourier layers and $\{q_g, dt, S^n\}$ as inputs (i.e Ni = 3). The input parameter dt is a scalar, therefore, we reshape it in a constant map of shape $(95, 60)$. Moreover, q_g is also a scalar. We reshape it in a (95,60) map which values are zeroes everywhere except at the well location where it takes q_g as value.

We split the data in a train and test sets with a 80/20 splitting ratio. We train the model on a NVIDIA V100 GPU during 27 hours using Adam optimizer, a batch size of 10, a learning rate of 5×10^{-5}, a momentum of 0.9, a weight decay of $1. \times 10^{-4}$ and keep the model parameters corresponding to the lowest

test loss value. We show the relative $L2$ loss evolution on the figure 5.9. The lowest test loss value is 1.9×10^{-3} reached at epoch 17285. The corresponding train loss value is 2.0×10^{-3}.

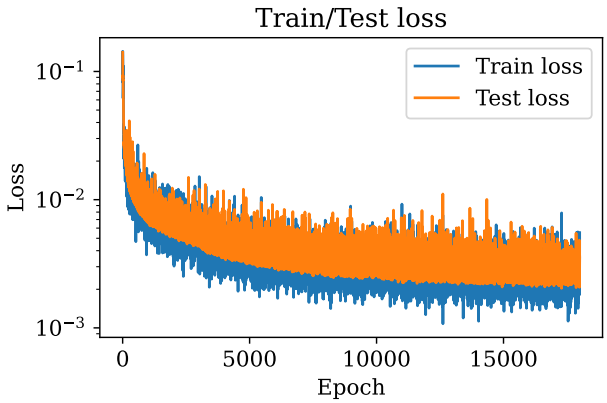


Figure 5.9: L2 loss evolution through epochs for test case 1. The lowest test loss value is 1.9×10^{-3} reached at epoch 17285. The corresponding train loss value is 2.0×10^{-3}.

5.5.1.2 Test case 2

We use the Neural Network architecture presented in figure 5.2 with $Nc = 64$ channels in the Fourier layers. As the case is more complex than the test case 1, the neural network is harder to train. To alleviate this complexity, the implicit pressure is added to the input features . We therefore use $\{P_{imp}, q_g, dt, S^n\}$ as input features (i.e Ni $= 4$). The input parameter dt is a scalar, therefore, we reshape it in a constant map of shape $(95, 60)$. Moreover, q_g is also a scalar. We reshape it in a (95,60) map which values are zeroes everywhere except at the well location where it takes q_g as value. S^n and P_{imp} can be used straightforward.

We split the data in a train and test set with a 80/20 splitting ratio. We train the model on a NVIDIA V100 GPU during 132 hours using Adam optimizer, a batch size of 128, a momentum of 0.9, a weight decay of $1. \times 10^{-4}$ and keep the model parameters corresponding to the lowest test loss value. We start with a learning rate of $1. \times 10^{-4}$, at iteration number 1000, we decrease it to $5. \times 10^{-5}$, then $1. \times 10^{-5}$ at iteration number 1600 and finally we set the learning rate to $5. \times 10^{-7}$ at iteration number 3600 and until the end.

We show the $L2$ loss evolution in the figure 5.10. The lowest test loss value is 8.7×10^{-4} reached at epoch 7295. The corresponding train loss value is 9.3×10^{-4}.

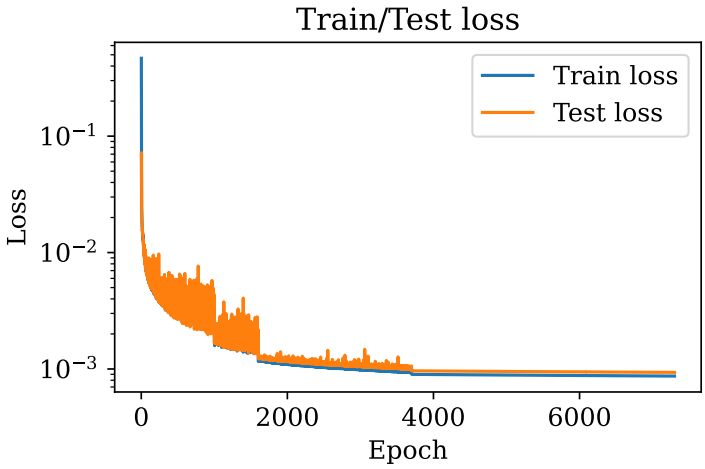


Figure 5.10: L2 loss evolution through epochs for the test case 2. The lowest test loss value is 8.7×10^{-4} reached at epoch 7295. The corresponding train loss value is 9.3×10^{-4}. We start with a learning rate of $1. \times 10^{-4}$, at iteration number 1000, we change it to $5. \times 10^{-5}$, then $1. \times 10^{-5}$ at iteration number 1600 and finally we set the learning rate to $5. \times 10^{-7}$ at iteration number 3600 and until the end.

5.5.2 Results

In this section, we present the results obtain using the hybrid methodology, first on the scenario presented in the figure 5.1 and then on the two tests cases. We compare the performances in term of Newton iterations obtained between the standard and hybrid approaches.

5.5.2.1 Single prediction example

We apply the hybrid methodology on the example scenario presented in figure 5.1 only during the well event, this implies that every step outside of the well event requires the same number of Newton iterations for the standard and hybrid approaches.

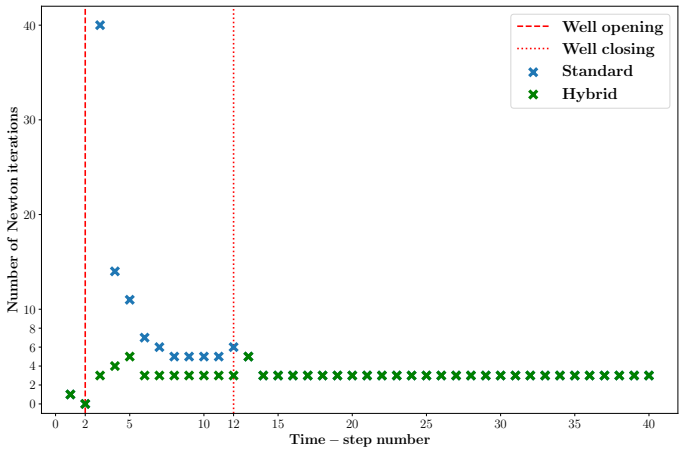


Figure 5.11: Hybrid Newton approach applied to the example scenario and it's impact on Newton's method.

We observe on the figure 5.11 a significant reduction in the number of Newton iterations during the well event. Immediately after the well opening, the standard approach (depicted by blue crosses) requires 40 iterations, whereas the hybrid approach (represented by green crosses) only necessitates 3 iterations. Furthermore, throughout the entire well event, the hybrid approach consistently requires fewer Newton iterations compared to the standard approach, resulting in a 68% reduction in Newton iterations during the well event and a 38% reduction across the entire scenario.

5.5.2.2 Test case 1

We launch simulations with the same parameters combinations of test case 1, $X_{init}^{n+1} = (P_{imp}, S^n)$ and $X_{ML}^{n+1} = (P_{imp}, S_{ML})$ as initial guesses. P_{imp} is calculated using the Implicit Pressure Solver and S_{pred} using the model obtained in the previous section. The results are presented in figure 5.12.

We observe that the hybrid methodology facilitates Newton's method initialization directly within the domain of quadratic convergence, resulting in a maximum of 5 Newton iterations for the training set and 4 for the test set. This is particularly intriguing as it appears to scale with problem complexity. In essence, the more challenging the problem using the standard methodology, the greater the potential benefit from employing the hybrid methodology. With the hybrid formulation, we achieve an average speedup

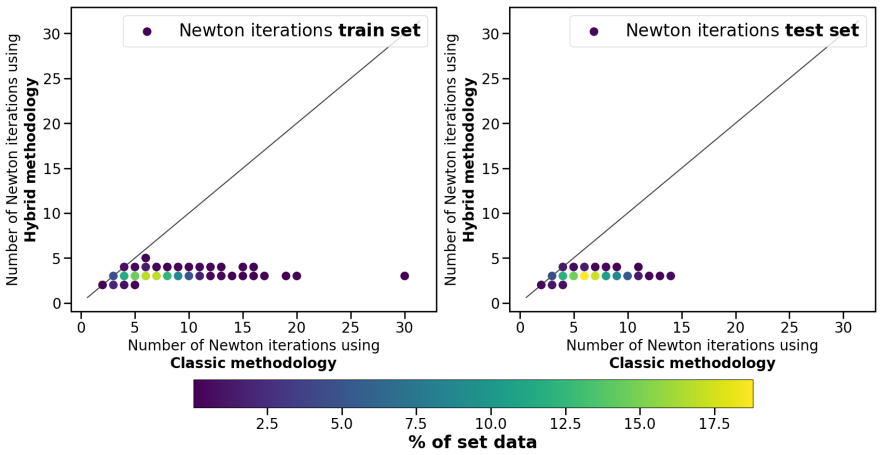


Figure 5.12: Test case 1 scatter plot of the number of Newton iterations needed to converge using reference (classic) methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using reference (classic) and Hybrid methodologies for the train and test set respectively.

of 54%, translating to 54% fewer Newton iterations compared to the reference methodology for the training set and a 53% reduction for the test set.

5.5.2.3 Test case 2

We launch simulations with the same parameters combinations of test case 2, $X_{init}^{n+1} = (P_{imp}, S^n)$ and $X_{ML}^{n+1} = (P_{imp}, S_{ML})$ as initial guesses. P_{imp} is calculated using the Implicit Pressure Solver and S_{ML} using the model obtained in the previous section.

Considering every simulations and using the hybrid methodology, we speed up by 39% , i.e 39% less Newton iterations than the standard methodology the computations for the training set and by 38.7% for the test set.

We observe in the figure 5.13 that for 17 simulations for the test set, the hybrid methodology perform very slightly worse than the standard methodology since it requires 1 more Newton iteration to converge. This can be explained by two main factors, the quality of the model and the position of the points in the parameters space. Indeed, we do not observe this issue in the test case 1 where the model has a test set loss of 2.02×10^{-2} while the

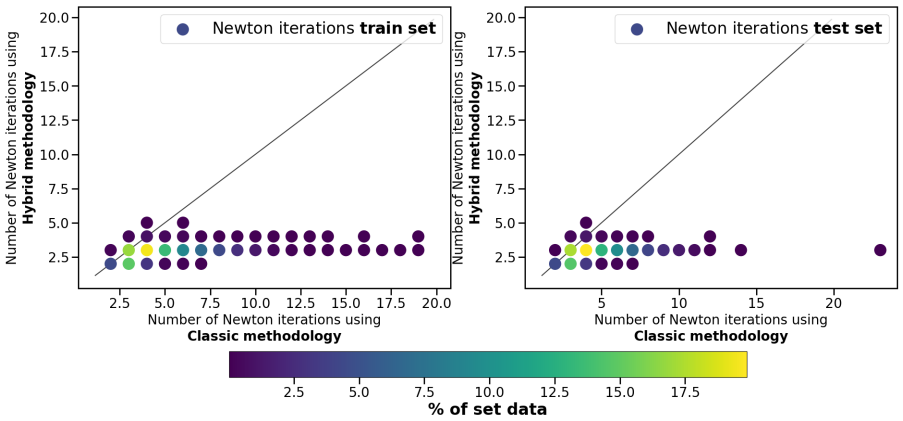


Figure 5.13: Test case 2 scatter plot of the number of Newton iterations needed to converge using reference (classic) methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using reference (classic) and Hybrid methodologies for the train and test set respectively.

test case 2 model has a test set loss of 1.16×10^{-1}. Moreover, we show on figure 5.14 that the points are close to the boundary of the space parameters. Models trained through supervised learning are sensible to data coming from the boundaries of the space parameters. In this case, we observe that data with low gas injection rate q_g and high step of time dt can present this issue.

5.5.3 Discussion

Considering a wide range of injection scenarios, we show that it is possible to learn the impact of a well event on a reservoir. We speed up by 53% the handling of well events for the test case 1 and by 38% for the test case 2. Moreover the hybrid Newton methodology is quite general and can be applied to any problem that requires an important number of costly iterations. Finally, we observe that the more challenging the problem is using the standard methodology, the greater the potential benefit from employing the hybrid methodology. However, there are some limiting issues that needs to be considered.

Constant discretization

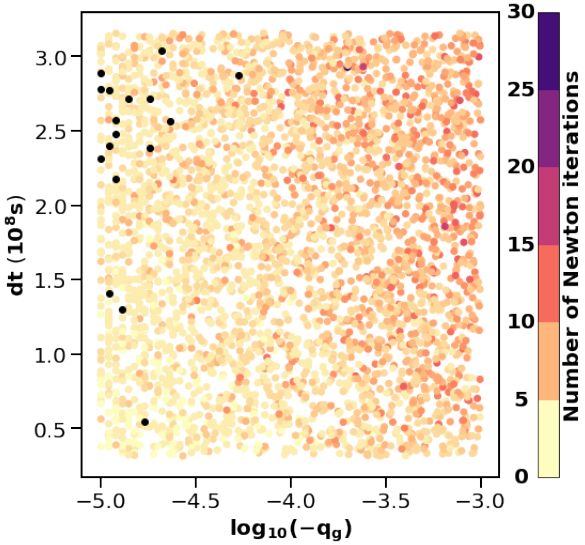


Figure 5.14: Distribution of Newton iterations considering well event parameters q_g the well injection flow and dt the time-step in seconds. The black points represent the simulations where the hybrid methodology requires more Newton iterations than the standard methodology.

We use a specific discretization (SHPCO_2 S mesh) for the data generation and we predict on the same discretization. This implies that the model would not work for different meshes. A new model has to be generated. However, the idea of a model invariant to discretization is developed through Neural Operators [6]. A model could be trained using data from different discretizations and predict the solution on multiple discretizations.

Constant well position

The methodology is applied on a constant grid with a constant well position. While the pressure variations are global during a well event, the saturation variations are local. Therefore, if we change the well position, the model prediction will not be accurate.

5.6 Conclusion

We proposed in this article a methodology to alleviate the impact of well events during the numerical simulation of CO_2 storage in the subsurface.

We complement the standard numerical algorithm by predicting an initialization of Newton's method directly in the domain of convergence using a supervised learning approach based on recently developed Fourier Neural Operators. Our results show a significant decrease in the number of Newton iterations required for convergence, while ensuring the convergence to the correct solution. Moreover the hybrid Newton methodology is quite general and can be applied to any problem that requires an important number of costly iterations. Finally, we observe that the more challenging the problem is using the standard methodology, the greater the potential benefit from employing the hybrid methodology.

In reservoir simulation, the well location is usually constant as it is numerically costly to optimize the well location and the reservoir parameters at the same time. Therefore, accelerating the numerical handling of well events for a wide range of positions is an interesting perspective. To do so, a local approach could be used, i.e create a model that predicts the saturation only in the near-well region.

Bibliography

[1] E. Ahmed, Ø. Klemetsdal, X. Raynaud, O. Møyner, and H. M. Nilsen. Adaptive Timestepping, Linearization, and A Posteriori Error Control for Multiphase Flow of Immiscible Fluids in Porous Media with Wells. *SPE Journal*, pages 1–21, 10 2022.

[2] Subhrajyoti Bhattacharyya and Aditya Vyas. A novel methodology for fast reservoir simulation of single-phase gas reservoirs using machine learning. *Heliyon*, 8(12):e12067, 2022.

[3] Olga Fuks and Hamdi Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. 07 2020.

[4] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2016.

[5] Jianguo Huang, Haoqin Wang, and Haizhao Yang. Int-deep: A deep learning initialized iterative method for nonlinear problems. *Journal of Computational Physics*, 419:109675, 2020.

[6] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces, 2021.

[7] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *CoRR*, abs/2010.08895, 2020.

[8] Alban Odot, Ryadh Haferssas, and Stéphane Cotin. Deepphysics: a physics aware deep learning framework for real-time simulation. *CoRR*, abs/2109.09491, 2021.

[9] J. W. Sheldon and W. T. Jr. Cardwell. One-dimensional, incompressible, noncapillary, two-phase fluid flow in a porous medium. *Transactions of the AIME*, 216:290–296, 1959.

[10] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[11] Ichrak Ben Yahia, Jean-Pierre Merlet, and Yves Papegay. Mixing neural networks and the Newton method for the kinematics of simple cable-driven parallel robots with sagging cables. In *ICAR 2021 - 20th International Conference on advanced robotics*, Ljulbjana, Slovenia, December 2021.

Chapter 6

Hybrid Newton Method - Local approach

Contents

6.1 1D problem **122**
6.1.1 Impact of local domain size 124
6.1.2 Impact of well location 130
6.2 2D SHPCO2 . . . . . **135**
6.2.1 Objectives and Workflow 135
6.2.2 Domain Decomposition 138
6.2.3 Data generation 140
6.2.4 Results 141

In the upcoming chapter, our focus remains on the hybrid Newton methodology for lessening the effects of well events. However, rather than using a global initialization as discussed in the previous chapter, we implement a local initialization in the near-well region. Indeed, as the variations of saturation are local and represent the main complexity in the numerical resolution, making predictions at the reservoir scale may be unnecessarily numerically costly and may lack flexibility for various reservoir scenarios. Then, the training and inference of a machine-learning predicting local saturation maps is numerically cheaper. Moreover, local initialization is the first step towards a

unique generalized well model which would mitigate the impact of well events for any possible well location.

We first work with a 1D test case to gain understanding on the local approach before moving to a modified 2D SHPCO_2 use case.

6.1 1D problem

We start with a simple 1D test case to set up the methodology. The reservoir is composed of 200 cells, with a left boundary pressure of 11 MPa and a right boundary pressure of 10 MPa. Moreover, the initial saturation S_0 is zero i.e there is only water is the reservoir. The porosity is constant equal to 0.2, the permeability constant equal to $1. \times 10^{-13}$ and we use a quadratic relative permeability law. Finally, an injection well is located at the middle of the reservoir.

Data generation

A well event is characterized by an injection flow rate q_g (only gas is injected) and an opening time dt. We generate 1000 injection scenarios at the center of the mesh using a Latin hypercube sampling strategy with $|q_g| \in]10^{-5}, 10^{-3}[\, m^2/s$ and $dt \in]0.5, 5[$ years and solve for each combination using a fully implicit solver. We then remove the scenarios that result in a production well instead of an injection well (lower imposed pressure at the well than the reservoir local pressure) which results in a dataset consisting of 731 data points.

Data exploratory analysis

We show on figure 6.1 an example of pressure and saturation solution obtained. We define the well impact extension (red dotted line) as the zone around the well that has been impacted by the well event. We observe on the figure that the left well impact extension is about 62 cells while the right cell extension is of 65 cells.

We present on figure 6.2 the evolution of the saturation for the same example at different Newton iterates (left figure) and the evolution of residual through all iterates (right figure) for the previous test case exposed in figure 6.1. The initial saturation is zero everywhere. We observe on the saturation plot that the first iteration (in blue) results in a small spike localised on the well

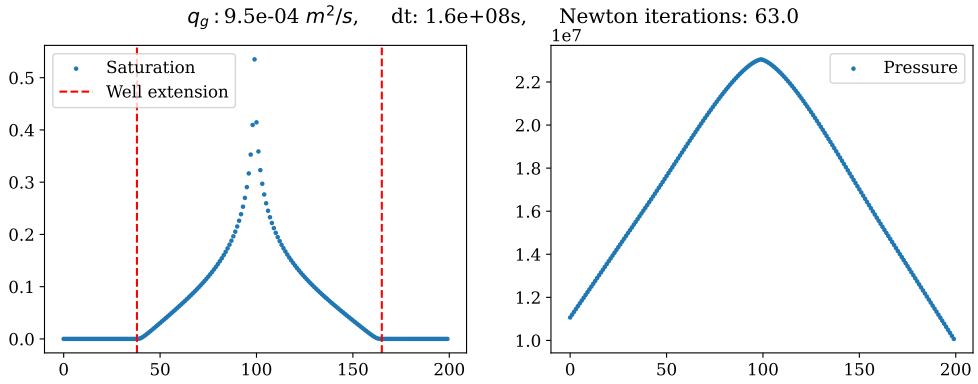


Figure 6.1: Example of saturation (left) and pressure (right) obtained through the numerical resolution using $|q_g| = 9.5e{-}4m^2/s$ and $dt = 1.6{\times}10^8 s$ (i.e ≈ 5 years).

location, then the next iterates results in larger and larger saturation "spikes" all centered on the well location. On the residual evolution plot, we observe a typical Newton's method convergence plot, with the residual decreasing slowly until the quadratic attraction zone is reached.

Therefore, we observe that there is a link between the well extension and the number of Newton iterations, meaning that ideally, a local initialization in the near well region only needs to cover the well impact extension. If this local prediction is accurate, then the initial guess should be in the quadratic convergence regime of Newton's method. Indeed, as the local domain size is a parameter defined by the user, there is a trade-off between choosing a large local domain or a small local domain. A large local domain is more likely to contain the well extension but the training and inference process may be more costly while a small local domain is faster to train and infer but may lack accuracy.

On the whole dataset, we show on figure 6.3 the numerical performances. We observe that the higher the well injection flow rate, the more Newton iterations required to converge.

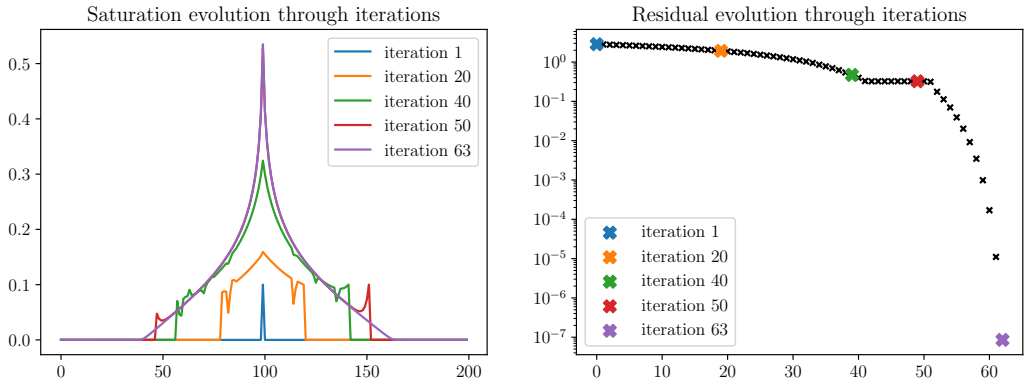


Figure 6.2: Saturation (left) and residual (right) evolution through Newton iterations. The final solution is presented in the figure 6.1.

6.1.1 Impact of local domain size

In this section, we will study the impact of the local domain size for the hybrid initialization. The expected behaviour is that if the local domain is larger or equal to the well extension impact, then we should obtain performances comparable to the global initialization. However, if the local domain is strictly smaller than the well extension impact, the local method should require more Newton iterations than the global method.

We show on the figure 6.4 the distribution of well extension impacts in order to determine the short and long range local domains.

We consider two local domain sizes: one short range consisting of 21 cells (well cell plus 10 cells in each direction) and one long range consisting of 81 cells (well cell plus 40 cells in each direction). We apply the hybrid Newton's methodology for each of these cases.

6.1.1.1 Results

Model training: short-range local domain

We train a FNO model on the short-range local domain using as inputs q_g, dt and P_{imp}. It is to note that, for a constant well location, the solution in saturation at the next time-step is entirely determined by q_g and dt. We use P_{imp} as an input feature as it eases this specific training process but it is not mandatory.

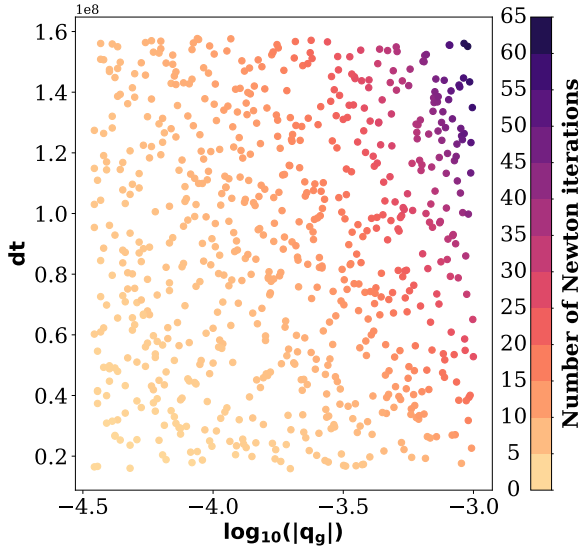


Figure 6.3: Local approach 1D test case distribution of Newton iterations considering well event parameters q_g the well injection flow rate and dt the time-step.

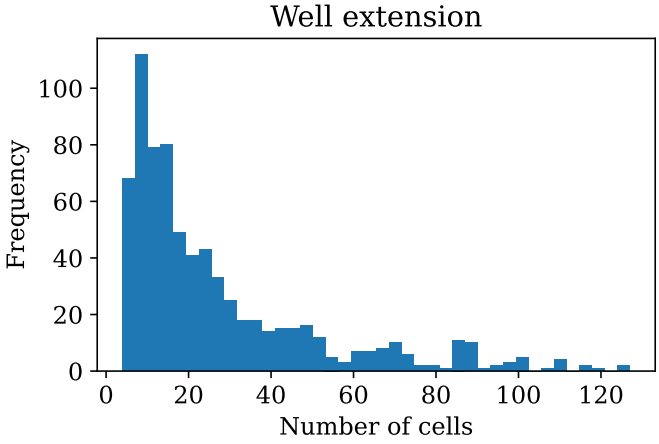


Figure 6.4: Distribution of well extension impacts in terms of number of cells.

The input features are reshaped as in the global approach chapter: q_g is a map full of zeros except at the well location where it takes q_g as value, dt a constant map and P_{imp} the pressure obtained through the implicit pressure

solver. Each input feature is of size the number of local cells i.e 21. After concatenation, the entire dataset has the shape $(731, 21, 3)$. We split it in train/test with a 80/20 ratio. The output or predicted feature is the saturation at time dt under the well injection flow rate q_g. We use a Fourier Neural Operator neural network architecture for the training, the loss function is the relative L_2 error and we keep the model parameters associated to the lowest test loss. The figure 6.5 shows the loss evolution through 1000 epochs. The best test loss is obtained at epoch 967 and is 3.56×10^{-3} and its corresponding train loss is 4.61×10^{-3}. The training process took around 5 minutes on CPU.

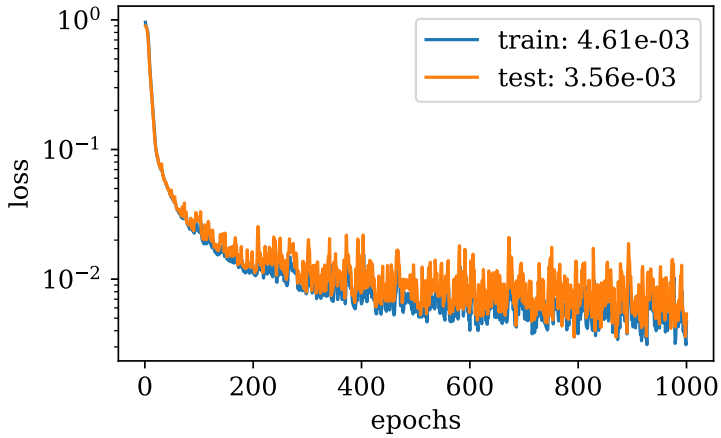


Figure 6.5: Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 3.56×10^{-3} reached at epoch 967. The corresponding train loss value is 4.61×10^{-3}.

Model training: long-range local domain

We are now interested in a long-range local domain, i.e a local domain composed of 81 cells centered on the well. We train using the exact same hyperparameter and the same dataset than the short-range local domain. We show on the figure 6.6 the evolution of the relative L_2 error through 1000 epochs. We keep the model parameters corresponding to the lower test loss value which is 5.29×10^{-3} and 6.88×10^{-3} its corresponding train loss value.

Hybrid Newton performances

The saturation prediction model is plugged into the solver as an initialization

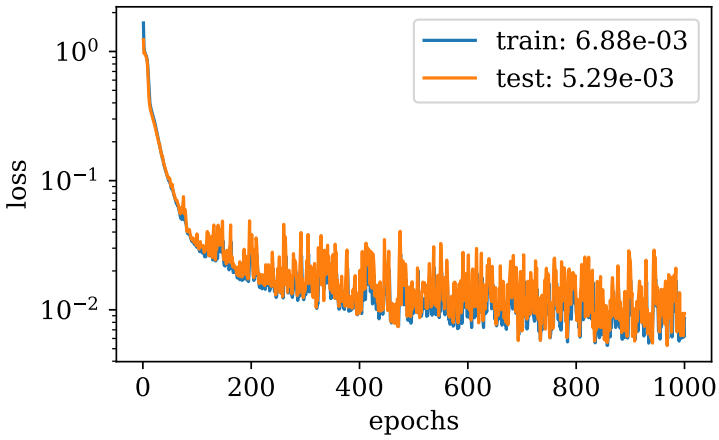


Figure 6.6: Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 5.29×10^{-3}. The corresponding train loss value is 6.88×10^{-3}.

of Newton's method with the implicit pressure. We highlight on the figure 6.7 the two main observed behaviours. The first one on the left figure corresponds to the local domain not reaching the well extension impact while the right figure shows a case where the size of the local domain is superior or equal to the well extension impact.

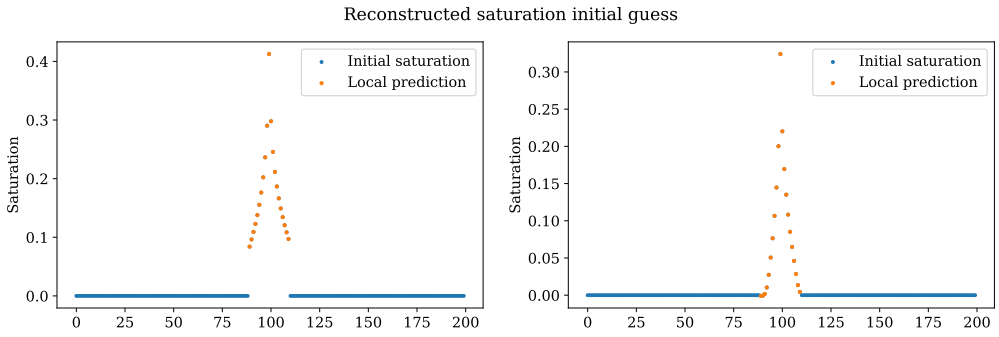


Figure 6.7: Example of reconstructed initial saturation guess composed of two parts, local neural network prediction and initial saturation in the reservoir. The left figure highlights a case where the local domain does not reach the well extension impact while the right figure shows a case where the size of the local domain is superior or equal to the well extension impact.

We apply the local hybrid initialization on the whole dataset for both short-range and long-range domains and show the results in terms of newton iterations per dataset on the figure 6.8. We observe two regimes separated by a red vertical line. The first regime corresponds to the cases where the number of Newton iterations is reduced to a constant amount through the hybrid method. The second one is a constant reduction of the number of Newton iterations. The explanation of these behaviours is straightforward, the first regime is composed of cases such as exposed in the right figure of 6.1 i.e the local domain size is larger than the well extension impact. This regime is identical to the global approach (since we pad the global initial guess saturation with zeros outside of the local prediction domain) which provides identical results. The second regime is composed of cases such as exposed in the left figure of 6.1 i.e the local domain size is smaller than the well extension impact. More specifically, in terms of Newton iterates, this case can be seen as initializing 'half-way' to the solution. Some iterations are still required to reach the well extension impact. Therefore, this initialization is not in the Newton's quadratic convergence zone.

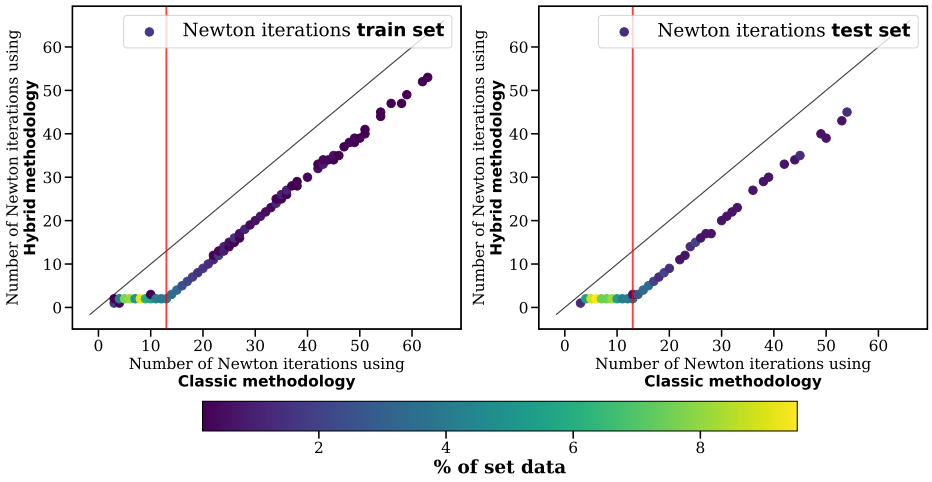
The table 6.1 sums up the result quantitatively. We observe that there is an important difference in terms of performances between each regime, it is therefore crucial to ensure that the local domain size meets at least the well impact extension even though we do not know in advance this last.

Approaches		**Hybrid**		
Regimes		**1**	**2**	**All**
Short-range	Train	74%	42%	**51**%
	Test	73%	42%	**52**%
Long-range	Train	84%	81%	**83**%
	Test	85%	78%	**84**%

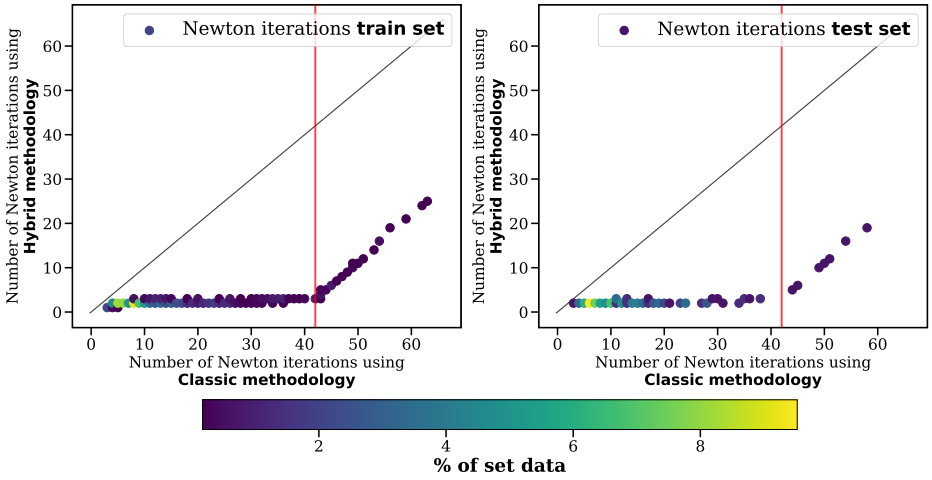
Table 6.1: Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and the long-range local approaches applied on the global test case 1 dataset. 'Regime 1' correspond to the case where the local domain extension is smaller than the well impact extension and 'regime 2' is the opposite. 'All' regime refers to both regime 1 and 2.

Linear interpolation

A straightforward extension that needs to be addressed is to complement



(a) Short-range local domain



(b) Long-range local domain

Figure 6.8: Short-range (figure a) and long-range (figure b) local approach 1D test case scatter plot of the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using standard and Hybrid methodologies for the train and test set respectively. Finally the red vertical line shows the separation between the two main regimes.

the case where the local domain of prediction is not large enough to catch the saturation variations with a linear interpolation in order to mitigate the discontinuity. Indeed, we hope to displace the red vertical lines of figure 6.8 to the right by doing so in order to extend the first regime as much as we can.

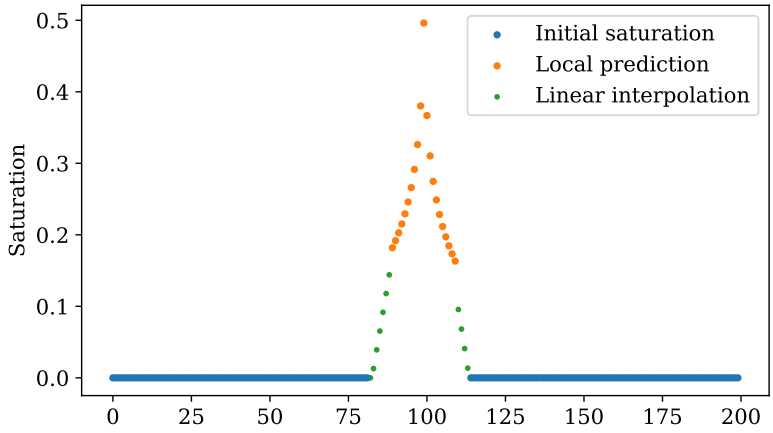


Figure 6.9: Reconstructed global saturation using the initial saturation (blue), the local predicted saturation (orange) and a linear interpolation (green).

We present the results obtained using the short-range local model with linear interpolation on the figure 6.9. We observe that the first regime remain the same as with no interpolation, however the second regime is modified. Indeed, the interpolation reduced the required iterations by a small amount, between 1 and 5 overall.

The table 6.2 presents the reduction of Newton iterations in % for the short-range and short-range with interpolation hybrid Newton. Overall we observe a gain of 5% in the performances which is non-negligible.

6.1.2 Impact of well location

We are now interested in the generalization of the local model at different well locations. Therefore, the well location becomes a parameter in the dataset generation and the information of the well location is included in the implicit pressure. Before getting into the workflow, we detail the intuition of why the

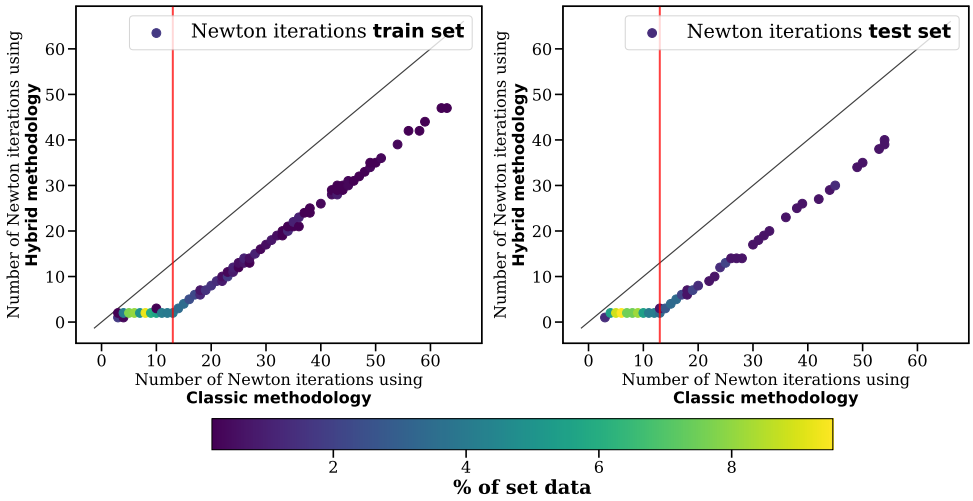


Figure 6.10: Local approach 1D test case scatter plot of the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology with linear interpolation on the train set (left figure) and on the test set (right figure).

well location information is included in the implicit pressure.

Considering two reservoir problems that are exactly identical, with the same well parameters but different well locations, we depict on figure 6.11 the resulting global and local implicit pressure and saturation profiles. It is noteworthy that, despite having identical well events, significant differences arise in both the aspect and values of the pressure profiles. Additionally, the saturation solutions differ, as the gas flows exhibit distinct directions.

Data generation

We generate 5000 injection scenarios at different locations of the mesh using a Latin hypercube sampling strategy with $|q_g| \in]10^{-4.5}, 10^{-3}[m^2/s$, $dt \in]0.5, 5[$ years and the well location between one quarter and three quarter of the domain corresponding to 100 possible cells. We solve for each combination using a fully implicit solver. We then remove the scenarios that result in a production well instead of an injection well (lower imposed pressure at the well than the reservoir local pressure) which results in a dataset consisting of 4686 data points.

Approaches Regimes		Hybrid		
		1	**2**	**All**
Short-range	Train	74%	42%	**51**%
	Test	73%	42%	**52**%
Short-range + interpolation	Train	74%	50%	**56**%
	Test	73%	50%	**57**%

Table 6.2: Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and the short-range with interpolation local approaches applied on the global test case 1 dataset.

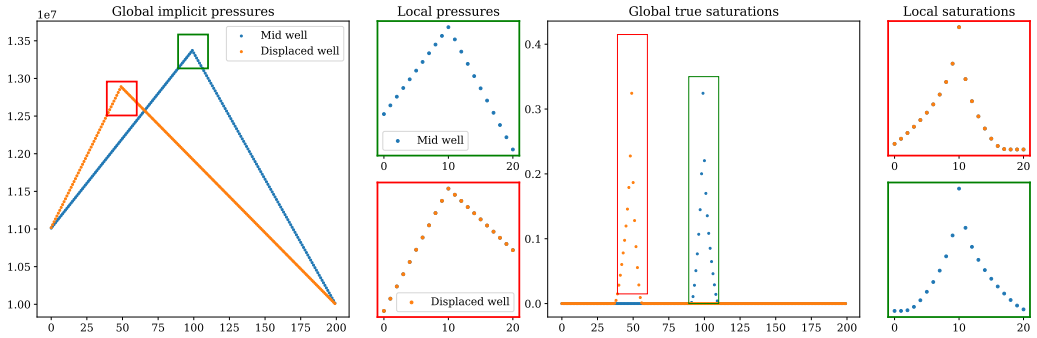


Figure 6.11: Comparison of global (left) and local (upper and lower mid-left) implicit pressure for different well locations and their resulting true global (mid-right) and local (upper and lower right) saturations obtained using the parameters: $q_g = 2 \times 10^{-4} m^2/s$ and $dt = 9.1 \times 10^7 s$.

6.1.2.1 Results

Model training: short-range local domain

We train a FNO model on the local domain of 21 cells using as inputs q_g, dt and P_{imp}. These features are reshaped as: q_g is a map full of zeros except at the well location where it takes q_g as value, dt a constant map and P_{imp} the pressure obtained through the implicit pressure solver. It is to note that the well location information is characterized by P_{imp}.

Each input feature is of size the number of local cells i.e 21. We split it in train/test with a 80/20 ratio. The output or predicted feature is the saturation at time dt under the well injection flow rate q_g. We train a FNO

model, the loss function is the relative L_2 error and we keep the model parameters associated to the lowest test loss. The figure 6.13 shows the loss evolution through 1000 epochs. The best test loss is obtained at epoch 962 and is 3.26×10^{-3} and its corresponding train loss is 4.35×10^{-3}. The training process took around 10 minutes on a NVIDIA A100 GPU.

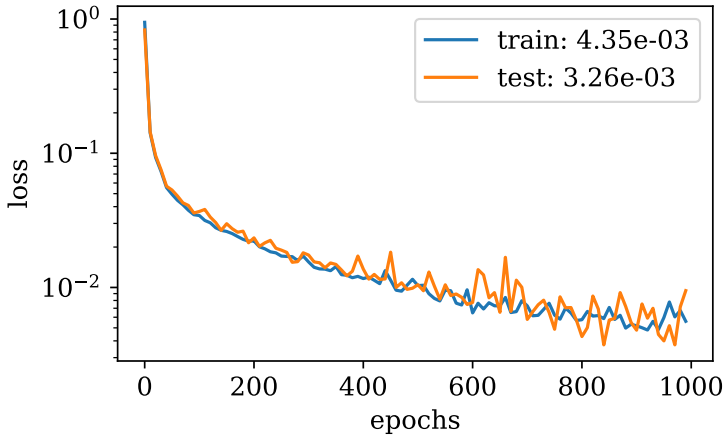


Figure 6.12: Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 3.26×10^{-3}. The corresponding train loss value is 4.35×10^{-3}.

Model training: long-range local domain

We train a FNO model on the local domain of 81 cells using as inputs q_g, dt and P_{imp}. These features are reshaped as: q_g is a map full of zeros except at the well location where it takes q_g as value, dt a constant map and P_{imp} the pressure obtained through the implicit pressure solver. It is to note that well location information is characterized by P_{imp}.

Each input feature is of size the number of local cells i.e 81. We split it in train/test with a 80/20 ratio. The output or predicted feature is the saturation at time dt under the well injection flow rate q_g. We train a FNO, the loss function is the relative L_2 error and we keep the model parameters associated to the lowest test loss. The figure 6.13 shows the loss evolution through 1000 epochs. The best test loss is obtained at epoch 967 and is 2.72×10^{-3} and its corresponding train loss is 4.27×10^{-3}. The training process took around 15 minutes on a NVIDIA A100 GPU.

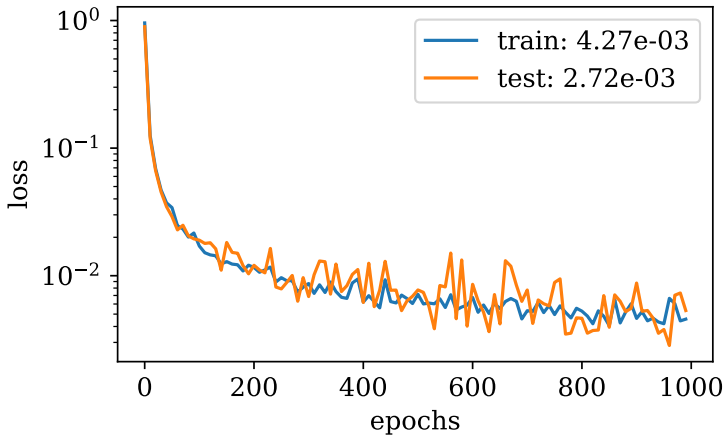


Figure 6.13: Relative L2 loss evolution through 1000 epochs for the short-range local model. The lowest test loss value is 2.72×10^{-3}. The corresponding train loss value is 4.27×10^{-3}.

Hybrid Newton performances

The saturation prediction model is plugged into the solver as an initialization of Newton's method with the implicit pressure. We highlight on the figure 6.14 the two main observed behaviours. The left figure corresponds to the local domain not reaching the well extension (regime 1) impact while the right figure corresponds to a local domain superior or equal to the well extension impact (regime 2).

We apply the local hybrid initialization on the whole dataset for the short-range and long-range local domains and show the results in terms of newton iterations per dataset on the figure 6.15. We observe again the two regimes separated by a red vertical line.

The table 6.3 sums up the result for the long-range local domain with different well locations. We observe that the local domain size is crucial as there is an important difference in performances between the short-range and long-range local domains. Moreover, if we compare the long-range with constant well location and long-range with variable well-location, it seems that the second regime is better handled using a variable well location. The variable well location is an important first step towards a generalized well model which would work in any circumstances.

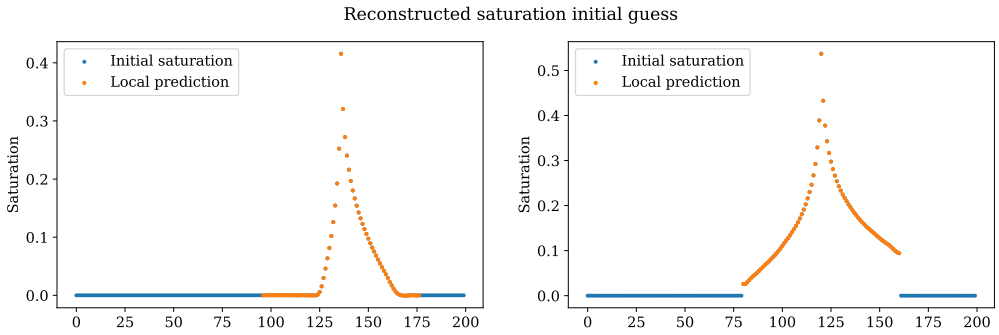


Figure 6.14: Example of reconstructed initial saturation guess composed of two parts, local neural network prediction and initial saturation in the reservoir. The left figure highlights a case where the local domain does not reach the well extension impact while the right figure shows a case where the size of the local domain is superior or equal to the well extension impact.

Approaches Regimes		**Hybrid**		
		1	**2**	**All**
Short-range & variable well location	Train	56%	19%	**47%**
	Test	57%	19%	**49%**
Long-range & variable well location	Train	85%	75%	**82%**
	Test	85%	76%	**83%**

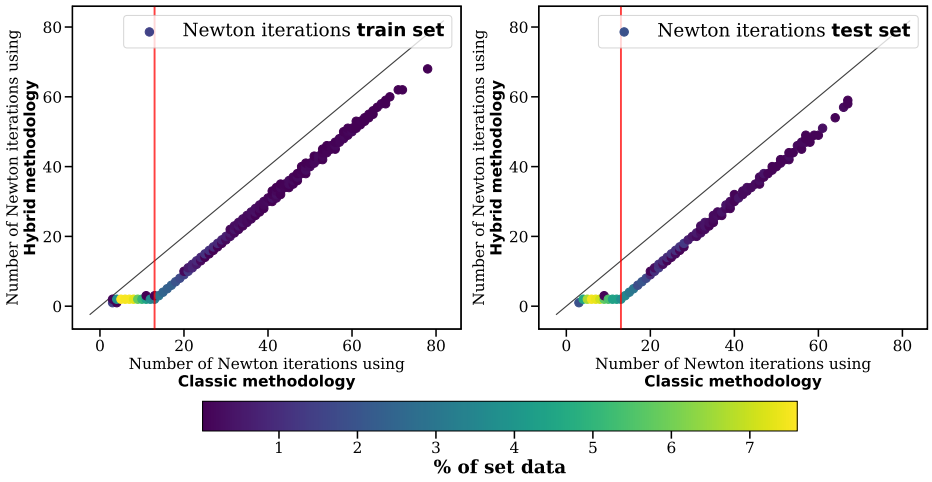
Table 6.3: Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and long-range with different well locations local approaches.

6.2 2D SHPCO2

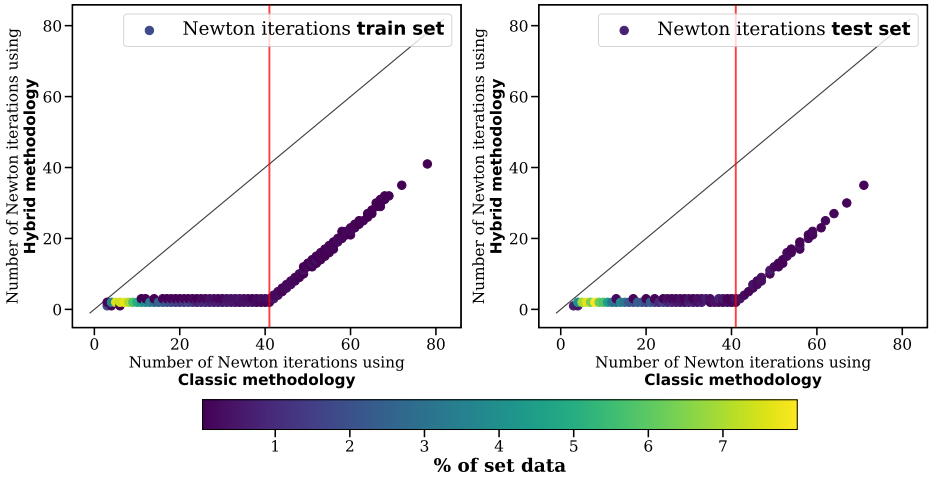
We are now interested in applying the local approach on a more realistic test case, we will be using the adapted SHPCO_2 reservoir geometry presented in 2.2.

6.2.1 Objectives and Workflow

In this section, we want to compare the performances of the local approach with the global approach performances, therefore we apply the local methodology on a test case where the global methodology has already been evaluated,



(a) Short-range local domain



(b) Long-range local domain

Figure 6.15: Short-range (figure a) and long-range (figure b) local approach 1D with a variable well location test case scatter plot of the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure). The color bar shows the distribution of Newton iterations using standard and Hybrid methodologies for the train and test set respectively. Finally the red vertical line shows the separation between the two main regimes.

i.e the test case 1 of the global approach presented in 5.4.2. We exclusively assess the 2D local approach using a fixed well location. The decision to focus on a constant well location is rooted in the challenges associated with training a machine-learning model for a 2D problem featuring a variable well location. The introduction of a variable well location leads to sampling challenges that complicate dataset generation, particularly considering that the number of degrees of freedom increases significantly and the prohibitively high cost of generating each data point.

Therefore, as a natural and cheaper way to have access to a variable well location model, we use a local initialization using the fully implicit solver directly on the local domain. Indeed, the solver itself allows the handling of any well location. This idea is inspired from Domain Decomposition philosophy even though to the best of our knowledge, we have not seen domain decomposition techniques used for this kind of problem.

Regarding the evaluation of the method, we want to assess the generalization capacity of the hybrid initialization. Therefore, we evaluate the model on three cases derived from the test case 1 and compare it with the domain decomposition approach. Each case corresponds to a different well location, we train a model on one location and apply it to the two others. We sum up the cases in the figure 6.16.

6.2.1.1 Case 1: In Distribution

The case 1 is the straightforward evaluation of the local approach on the global approach test case 1. The well location is indicated by a 1 on the figure 6.16.

6.2.1.2 Case 2: Near Distribution

We displace the well at a short-range location from the case 1. We generate a new dataset and apply the hybrid methodology with the model obtained in the case 1. This should result in local implicit pressure fields slightly different from the reference case 1 and therefore 'Near Distribution' (ND). The well location is indicated by a 2 on the figure 6.16. It's corresponding location on the grid is $(1675, 1725)$.

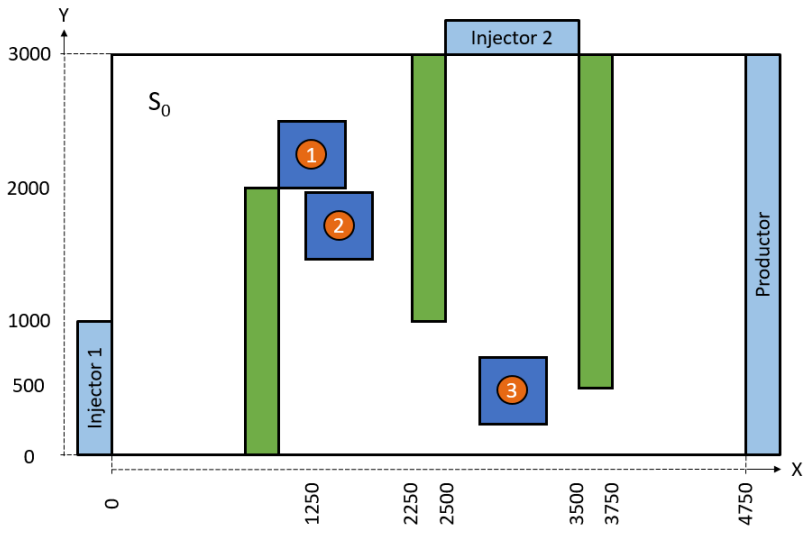


Figure 6.16: 2D SHPCO2 case geometry with three different local domains. Each local domain has a well at its center.

6.2.1.3 Case 3: Out of Distribution

Finally, we displace the well at long-range location from the reference case 1. We generate a new dataset and apply the hybrid methodology with the model obtained in the case 1. This should result in local implicit pressure fields far different from the reference case 1 and therefore 'Out of Distribution' (OOD). The well location is indicated by a 3 on the figure 6.16. It's corresponding location on the grid is $(2975, 425)$.

6.2.2 Domain Decomposition

In this section, we detail the "Domain Decomposition" methodology that we use as a reference for future performances comparison. The main idea of the local approach is that we want a local approximation of the solution in the near-well region. Therefore, the straightforward way is to use what we have, i.e the fully implicit solver. This approximation is interesting as it only requires to solve small linear systems. To do so, we need to formulate a local problem from the global configuration. We use the notation introduced at the Mesh definition and notations section 3.1.

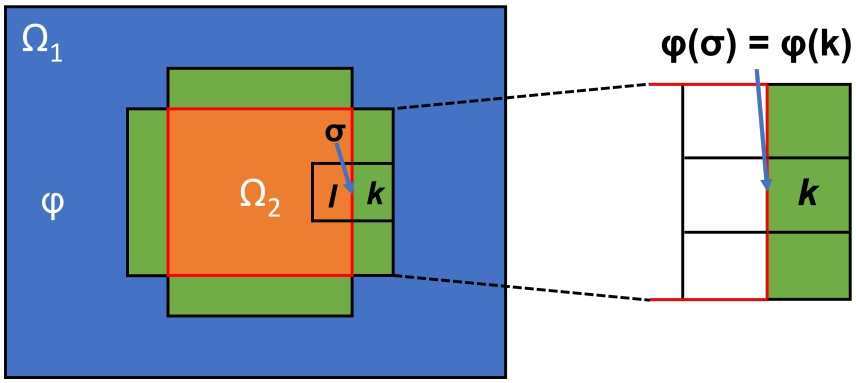


Figure 6.17: Example of global domain Ω_1 and local domain Ω_2. The cell function ϕ applied on a face σ from the boundary of Ω_2 is equal to ϕ applied to the cell k.

Let Ω_1 be the global domain mesh formed by the set of cells $\mathcal{K}_h^1$ and the set of faces $\mathcal{F}_h^1$. Similarly, let $\Omega_2 \subset \Omega_1$ be a local domain mesh formed by the set of cells $\mathcal{K}_h^2$ and the set of faces $\mathcal{F}_h^2$. With $\mathcal{K}_h^2 \subset \mathcal{K}_h^1$ and $\mathcal{F}_h^2 \subset \mathcal{F}_h^1$.

Regarding the cell properties, we directly extract them from the global domain to the local domain. Formally, let φ be a cell property function on the global domain Ω_1. Then for $k \in \mathcal{K}_h^1 \cap \mathcal{K}_h^2$ and $l \in \mathcal{K}_h^2$, $\varphi(k) = \varphi(l)$.

Thereafter, to extract face properties from the global domain to the local domain in order to create boundary conditions, we use the cells from the global domain which have a face in common with the local domain boundary faces. Let $\mathcal{F}_h^{2,b}$ the set of boundary faces of Ω_2 and $\mathcal{F}_h^{1,i}$ the set of internal faces of Ω_1. For $\sigma \in \mathcal{F}_h^{2,b} \cap \mathcal{F}_h^{1,i}$ such that for $k, l \in \mathcal{K}_h^1$, $\sigma = \partial k \cap \partial l$. Only one cell between k and l is not inside $\mathcal{K}_h^2$. Suppose $k \notin \mathcal{K}_h^2$, then $\varphi(\sigma) = \varphi(k)$.

It is to note that, in this work, we do not consider local domains that reaches the boundaries of the global domain, therefore, we do not detail this case.

In practice, we extract cell properties such as permeability, porosity, saturation, and pressure from the global domain to the local domain cells. Additionally, we extract properties like pressure and saturation from the global domain cell properties to the boundary faces of the local domain.

After the creation of the local domain problem formulation, we solve it using

the fully implicit solver. We then use the resulting pressure and saturation as local guesses that we will concatenate with the global standard initialization. Therefore, this is different from the hybrid initialization as this last uses the global implicit pressure and the concatenation of the local saturation guess and the global saturation standard initialization.

6.2.3 Data generation

We use the same data generation method than the one used for the case 1 in the global approach. It is to note that as we use for each case the same seed for the Latin hypercube sampling, the resulting parameters are the same for all cases. The only parameter that varies between cases is the well location. Otherwise the process is the same and we remind it here.

We launch simulations with a constant reservoir configuration except for three parameters, $S0$ the initial saturation, q_g the well injection flow rate and dt the time-step. We allow a maximum of 200 Newton iterations to converge. The convergence criterion is based on the residual norm and we iterate till $dt\|R\|_\infty/V \leq \epsilon$ with $\epsilon = 1. \times 10^{-6}$, V the cell volume and R the residual of the physical system.

We generate 5004 parameter combinations through a Latin Hypercube Sampling strategy within the following ranges: $S0 \in [0, 0.6]$, $|q_g| \in [10^{-5}, 10^{-3}]m^2/s$ which corresponds to a well pressure $\in]10, 20]$ MPa and $dt \in [0.1, 10]$ years. Note that P_{imp} is obtained using the Implicit Pressure solver (IMP).

6.2.3.1 Local domain extension

We use the SHPCO_2 reservoir geometry and consider the reference well location. Regarding the choice of the local domain size, we do not want to take heterogeneities into account. There is a barrier of permeability in a range of 4 cells from the well, therefore the maximum size is a square of 9x9 cells (4 cells extension in both direction plus the well cell).

6.2.3.2 Model training

We keep the same train and test datasets as the test case 1, i.e 4003 data for training and 1001 data for testing. We train a Fourier Neural Operator with q_g, dt, P_{imp} and S as input maps. We use the same reshaping methodology

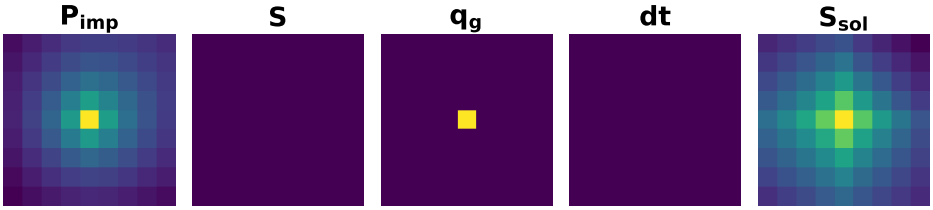


Figure 6.18: Qualitative view of local input features.

as the global approach for the input features. The Pressure and Saturation fields can be used straightforward as images of shape (9,9). The time-step is reshaped into a constant map of value dt and size (9,9). Finally, the well injection flow rate q_g is reshaped as a (9,9) map of zeros everywhere except at the well location, where it takes the value q_g.

We train the Fourier Neural Operator architecture presented in 5.2 with $Ni = 4$ and $Nc = 64$ for 15 minutes on a NVIDIA A100 GPU representing a total of 1500 epochs. We use Adam optimizer with a learning rate of 10^{-4}, a momentum of 0.9 and a weight decay of 10^{-4}. We use a relative $L2$ loss. We keep the model parameters corresponding to the lowest loss test value. The minimum loss value on the test set is 2.5×10^{-3} and is reached at the epoch 1368. Its corresponding train loss value is 2.5×10^{-3}. We show the loss evolution on the figure 6.19.

6.2.4 Results

We present the results of the local approach on the test case 1. First we detail the effect of the methodology on single samples and then we analyse on whole datasets.

6.2.4.1 Single prediction

Before presenting the results over the totality of each case, let us first have a look of what happens at the single prediction scale. We show on figure 6.20 an example of local saturation guess (upper left) predicted by the machine learning model on a sample coming from the case 1. We show the reconstructed global saturation guess (upper right) composed of the global initial saturation and the local guess. Then, we show the solution (lower left) and the error between the solution and the global guess (lower right). Qualita-

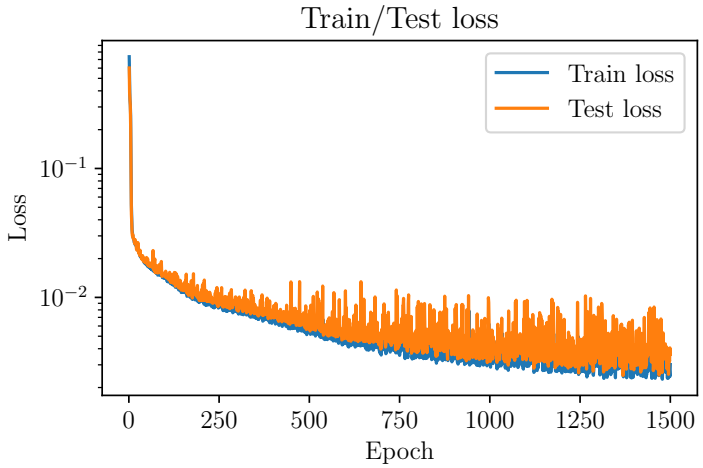


Figure 6.19: Relative L_2 loss error through 1500 epochs. The lowest test value is 2.5×10^{-3} reached at epoch 1368 and the corresponding train loss value is 2.5×10^{-3}.

tively, we observe that the local prediction is accurate as the error in the predicted is lower than outside the local domain.

In the figure 6.21, we present, on the same sample, the qualitative results of the Domain Decomposition method. We observe a similar behaviour, as the error between the global guess and the solution is small in the local domain and higher around the local domain.

We show on figure 6.22 the evolution of residual norms through Newton iterations associated to the example presented in figure 6.20 and 6.21. The stopping criterion is 10^{-6}. The standard Newton's method requires 8 iterations to converge. Using the local approach to create an initial guess closer to the solution leads to a reduction down to 4 Newton iterations for the hybrid initial guess and 4 Newton iterations for the Domain Decomposition initial guess. Regarding the quality of the initialization, we observe that the hybrid initialization starts at a value over 10^2, which is ten times higher than the standard initialization. Meanwhile, the Domain Decomposition initialization has the lowest initial residual norm. However, we observe that after the second Newton iterations, the hybrid and DD residual norms are quasi-equal. This is interesting as we would suppose that an initialization closer to the solution would result in a lower residual norm. The residual norm is com-

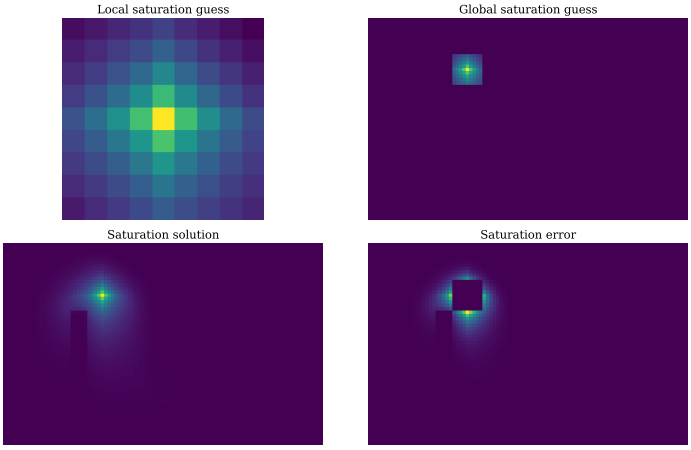


Figure 6.20: Case 1 local saturation machine learning prediction example (upper left), global saturation guess (upper right), saturation solution (lower left) and saturation error between the global guess and the solution (lower right).

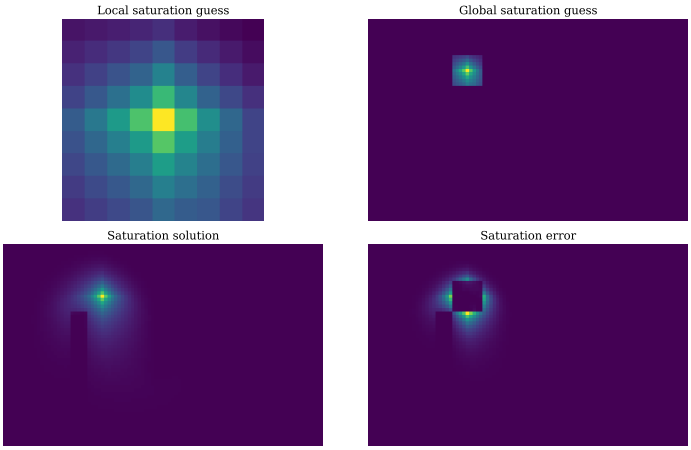


Figure 6.21: Case 1 local domain decomposition saturation solution example (upper left), global saturation guess (upper right), saturation solution (lower left) and saturation error between the global guess and the solution (lower right).

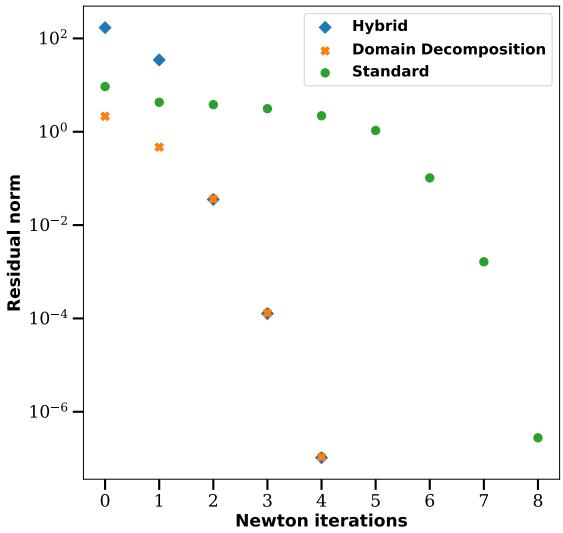


Figure 6.22: Residual evolution through Newton iterations.

puted using the following formula: $dt\|R\|_{\infty}/V$ with R the residual composed of two parts R_w the water residual and R_g the gas residual. dt is the time step and V the cell volume constant over the domain. Therefore, this metric is not adapted to measure the quality of the initialization.

6.2.4.2 Case 1: In Distribution results

We apply the hybrid methodology on the test case 1 dataset. The saturation machine learning model has been trained on this dataset, therefore we expect to assess the 'in distribution' numerical performances of the method.

In the figure 6.23, we present a comparative analysis between the standard Newton methodology and the Hybrid Newton methodology. We observe a substantial decrease in the number of Newton iterations, by 45.6% for the training set and by 45.4% for the test set. Notably, we emphasize the transition point between two regimes using a vertical red line, which occurs approximately after 7 Newton iterations. The first regime, strictly at the left of the vertical line (classic Newton iterations < 7) corresponds to the cases where the local domain extension is larger than the well impact extension while the second regime, at the right of the red vertical line (classic Newton iterations ≤ 7), corresponds to the opposite case, i.e the local domain

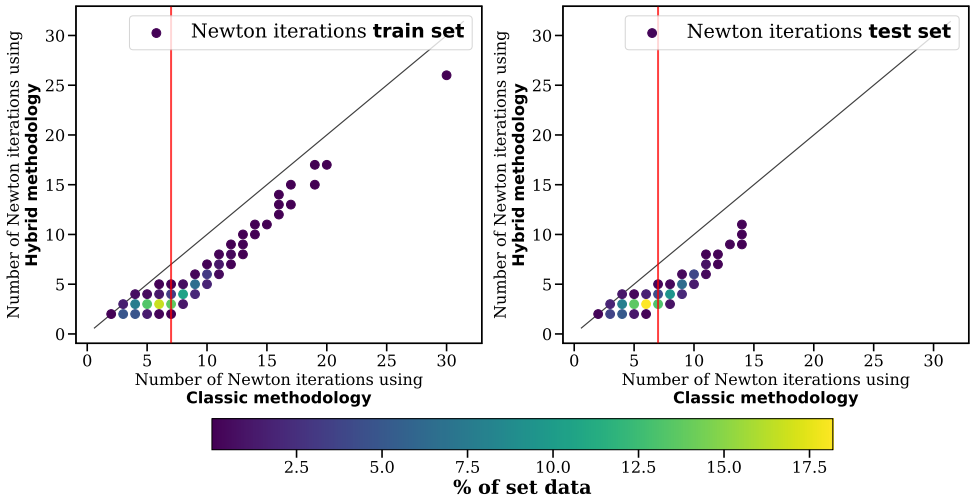


Figure 6.23: Test case 1 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure).

extension is smaller than the well impact extension.

Approaches / **Regimes**		**Hybrid**		
		1	**2**	**All**
Short-range & variable well location	Train	41.5%	48.3%	**45.6**%
	Test	41.4%	48.3%	**45.4**%

Table 6.4: Reduction of Newton iterations in % for each case compared to the standard approach for the short-range and long-range with different well locations local approaches.

In the global approach on the same test case, we noted a reduction of Newton iterations by 54% for the training set and by 53% for the test set. This is quite interesting as for the global approach, the machine learning model saturations on a grid of 5700 cells, while for the local approach, we predict on a grid of 81 cells and at the end, we observe a difference of performances less than 10%. Therefore, the gain in training and inference times of the local machine learning model seems to be worth compared to the training time and the inference time of the global approach machine learning model.

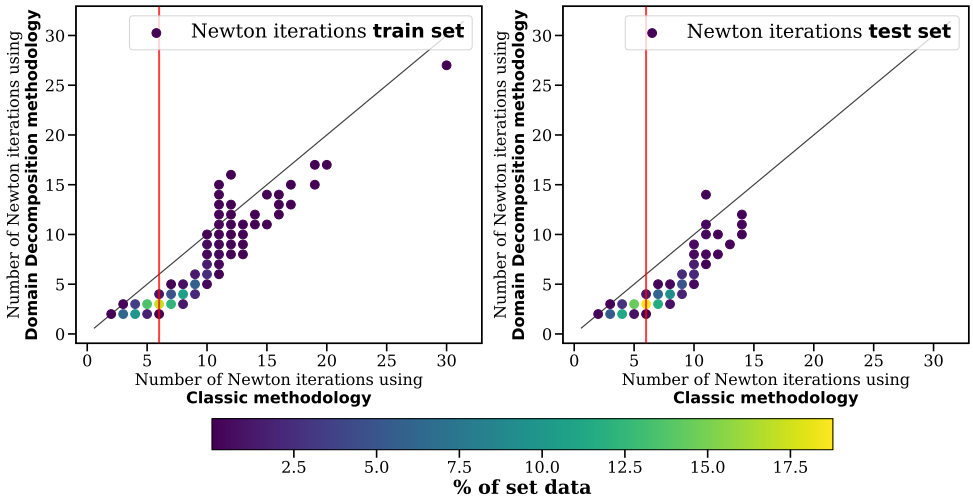


Figure 6.24: Test case 1 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using domain decomposition methodology on the train set (left figure) and on the test set (right figure).

Next, in the figure 6.24, we depict a comparison between the standard Newton methodology and the Domain Decomposition Newton methodology. We observe a noteworthy reduction in the number of Newton iterations, with a decrease of 44.6% for the training set and by 44.7% noted for the test set.

These numerical performances are comparable with the local hybrid approach and therefore, are worth compared to the global approach. Moreover, at equal numerical performances on the number of Newton iterations, the difference is on the computational cost of the creation and the inference each method.

6.2.4.3 Case 2: Near distribution results

We apply the hybrid methodology on a case where we displaced slightly the well from the test case 1 location. Therefore, the local pressure fields should be slightly different from the reference test case 1. We use the saturation predictive model trained on the test case 1 dataset. This case assesses the generalization potential of the methodology for 'near distribution' cases.

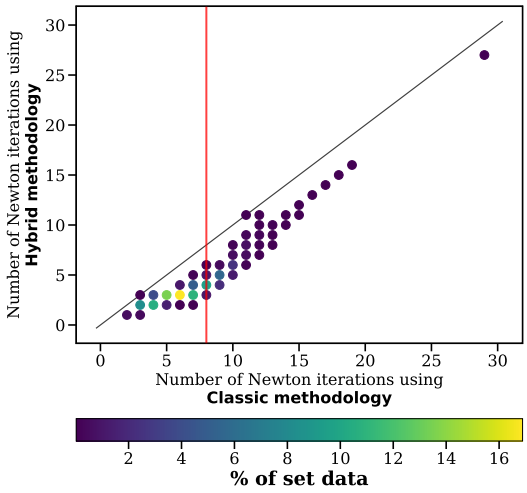


Figure 6.25: Case 2 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure).

We show on the figure 6.25 the comparison between standard Newton methodology and Hybrid Newton methodology. We observe a total reduction of Newton iterations, amounting to a reduction of 41.3%.

We show on the figure 6.26 the comparison between standard Newton methodology and Domain decomposition Newton methodology. We observe a total reduction of Newton iterations by 45.8%.

6.2.4.4 Case 3: Out of Distribution results

This last case assesses the 'out of distribution' application of the local approach. By positioning the well far from it's first location, the local pressure fields are far different from the original distribution. We apply the hybrid methodology on the resulting dataset using the saturation machine learning model trained on the test case 1 dataset.

We show on the figure 6.27 the comparison between standard Newton methodology and Hybrid Newton methodology. We observe a total reduction of Newton iterations by 17.2%.

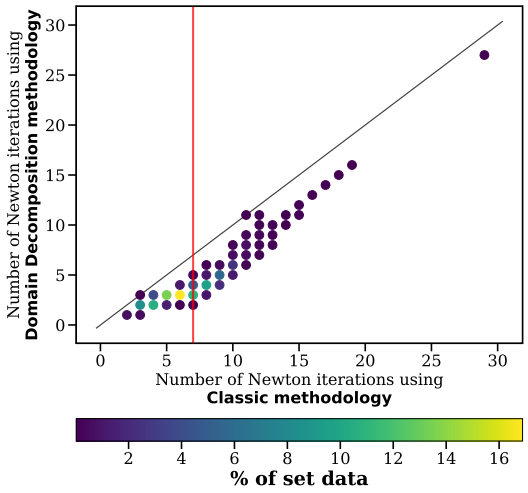


Figure 6.26: Case 2 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using domain decomposition methodology on the train set (left figure) and on the test set (right figure).

This result is expected as the new points that we infer are far from the training distribution. Moreover, we observe that in the first regime (left to the red vertical line), many cases have worse performances using the hybrid methodology than the standard methodology. Indeed, if the local domain contains the majority of the saturation variations over the whole domain, then a saturation prediction far from the solution can be worse than the standard initialization. In the second regime (right to the red vertical line), we observe that as the standard required number of Newton iterations increases, the hybrid approach reduces the requires number of Newton iterations compared to the standard approach. This can be explained as the local domain does not catch the majority of saturation variations on the global domain. Therefore, giving an initial saturation shape in the near-well region and let Newton's method correct and extend it is worth.

We show on the figure 6.28 the comparison between standard Newton methodology and Domain decomposition Newton methodology. We observe a total reduction of Newton iterations by 41.6%. This outperform by far the hybrid approach, which is expected as the hybrid methodology is applied on points where it should not work by construction.

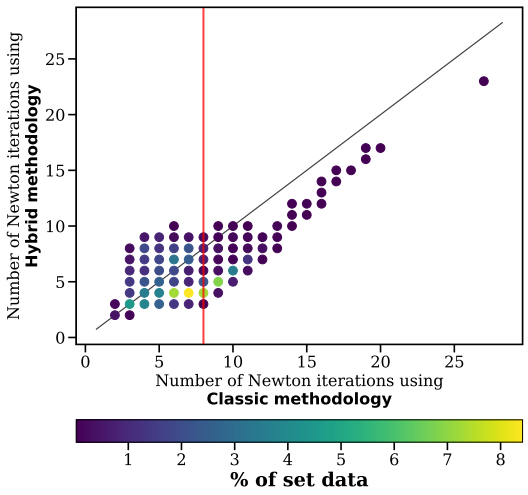


Figure 6.27: Case 3 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using hybrid methodology on the train set (left figure) and on the test set (right figure).

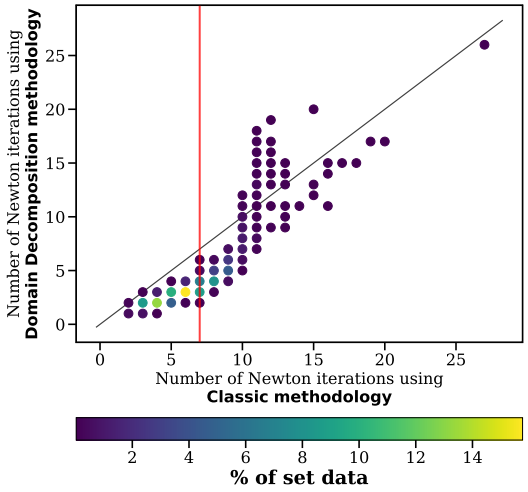


Figure 6.28: Case 3 local approach scatter plots showing the number of Newton iterations needed to converge using standard methodology versus using domain decomposition methodology on the train set (left figure) and on the test set (right figure).

6.2.4.5 Discussion and perspectives

We sum up the main results in terms of numerical performances in the table 6.5. Overall, we observe that the machine-learning hybrid methodology shows significant results for 'In Distribution' (case 1) and 'Near Distribution' (case 2) datasets while the Domain decomposition methodology shows significant and similar results for all three cases as it does not depend on a training distribution. Considering strictly these results, local domain decomposition methodology is better to use in all cases.

Approaches Regimes		**Hybrid**			**Domain Decomposition**		
		1	**2**	**All**	**1**	**2**	**All**
Case 1: ID	Train	41.5%	48.3%	**45.6**%	41.3%	45.5%	**44.6**%
	Test	41.4%	48.3%	**45.4**%	41.5%	45.7%	**44.7**%
Case 2: ND		38.5%	45.5%	**41.3**%	44.5%	46.7%	**45.8**%
Case 3: OOD		3.%	**39**.%	17.2%	46.4%	**38**.%	41.6%

Table 6.5: Reduction of Newton iterations in % for each case compared to the standard approach for the three local approaches applied on the global test case 1 dataset. 'Regime 1' correspond to the case where the local domain extension is smaller than the well impact extension and 'regime 2' is the opposite. 'All' regime refers to both regime 1 and 2.

Methodology comparison

We aim at comparing with a larger sight the hybrid methodology and the domain decomposition methodology. There are two main stages, an offline phase and an online phase. Regarding the offline phase, the domain decomposition does not require any while the hybrid methodology requires the creation of a dataset through the resolution of global problems which are numerically expensive as we want them to require an important amount of Newton iterations. Then the saturation model needs to be trained on this dataset, for the local approach, this part does not require a lot of time (e.g 15 minutes in this case). Then, for the online phase, the domain decomposition methodology requires a Newton's method to solve the local problem. This is not costly in the presented case as we consider a 9x9 local domain. However, for three dimensional problems or problems with more complex physics, the cost may raise quite fast. The hybrid methodology requires the inference of the saturation model. Therefore, the cost is proportional to the number of

parameters of the Neural Network. The FNO model has millions of parameters. In this study we did not focus on reducing the number of parameters as we first wanted a performing predictive model.

Conclusion and Perspectives

In this study, we constructed an initialization of Newton's method aiming at being closer to the solution than the standard initialization for well events problems, which often requires an important number of Newton iterations.

This initialization process consists of two steps. First we construct a local guess around the well where the majority of saturation variations are. Subsequently, we concatenate this local guess with the standard global initialization resulting in an initial guess that catches the main global variations in saturation, which is the unknown bringing the main non-linearity.

We propose two local guesses in pressure and saturation, the first one is obtained through the supervised learning of a Fourier Neural Operator model on a dataset generated during the global approach study. This model predicts a local saturation map, then for the pressure, we use an implicit pressure solver which only requires to solve one linear system as an initial guess. The second guess is obtained through a domain decomposition technique where we reformulate the local problem and solve it using a fully implicit solver. The resulting solutions in pressure and saturation are used as a local guess. This domain decomposition technique is particularly interesting as it handles any well locations.

We first assessed the machine-learning initialization local approach to 1D problems, where we studied the impact of local domain size and well locations. Our result highlights the important trade-off in terms of performances between a short-range and a long-range local domain. Moreover, we show that it seems possible to learn a local saturation predictive model for any well location.

When moving to 2D problems, it is challenging to generate a training dataset for any well location. Therefore, we used the domain decomposition initial guess as it can handle any well location and we simply realised some tests for different distribution for the machine-learning local guess. Subsequently, we applied the domain decomposition and the machine-learning local approaches to three cases. The first one 'In distribution' (ID), the second one 'Near-

distribution' (ND) and the last one 'Out of Distribution' (OOD). This name refers to the fact that we train the machine learning model predicting the saturation on the ID dataset and apply it on the ND and OOD datasets afterwards without any further training.

Both methods show a significant reduction in the required number of Newton iterations to converge, around 45% for the domain decomposition in all three cases while around 45% for the hybrid methodology in the ID set up. As soon as we step outside of the training distribution, we observe an important diminution of the numerical performances compared to the ID, around 41% reduction of Newton iterations for the ND case and down to 17% acceleration for the OOD case. However, for each case we can separate two main regimes, the first one corresponds to a local domain extension that catches (i.e larger than) the main saturation variations induced by the well event and the second regime corresponds to the cases where the local domain does not contain the main global variations of saturations. The first regime matches the cases with low standard Newton iterations while the second one matches the cases with high number of Newton iterations. An important result is that the hybrid methodology and the domain decomposition methodology show the same numerical performances for the second regime, with a reduction between 39% to 45% of Newton iterations depending on the case. For problems which require an important number of Newton iterations to be solved, both methodologies seem to be valuable. However for cases in the first regime, generally speaking, using a local guess may not be worth as the gain is smaller, but if we want to use one, the domain decomposition methodology is recommended. The choice between the hybrid or domain decomposition methodology to construct a local guess depends on the inference time. Indeed, if the hybrid methodology is significantly faster to infer than the domain decomposition methodology, then an interesting trade off could be to build a dataset using the local domain decomposition guess to accelerate the simulation and then train a neural network when enough data are generated. It is to note that, overall, the global approach from previous chapter outperforms the local approach. Indeed, for any given well event, the global initialization is in the quadratic convergence zone while we distinguish two cases for the local approach.

Chapter 7

Conclusion and perspectives

7.1 Conclusion

The main objective of this thesis was to adapt recent advances in physics informed deep learning in order to alleviate the impact of well events in the numerical simulation of multiphase flow in porous media.

We assessed the potential of machine learning models as direct solvers for two problems, the steady-state heat equation and the incompressible two-phase flow equation. Our results highlighted that it is possible to accurately learn parameter-to-solution operators using suitable neural architectures such as neural operators. However, when iterating in time using this 'black-box' model, errors accumulate and the prediction drifts away from the solution. In contrast, a traditional numerical solver provides guarantees on the solutions.

Therefore, we investigated hybrid approaches, combining the predictive capability of neural networks and the robustness of the traditional solvers. In practice, we proposed to adapt the hybrid Newton's method in two different ways, first as a global saturation initialization and second as a local saturation initialization in the near-well region where all the main variations of saturation are located.

For the global hybrid Newton's method, our results showed, using the Fourier Neural Operator as a predictive model, that it is possible to significantly re-

duce the required number of Newton iterations to reach convergence for a wide variety of well injection scenarios. Moreover, the global approach seems to scale with the difficulty: the more iterations required by the standard Newton's method, the larger the acceleration provided by the hybrid Newton's method. This method is particularly adapted to cases where the well location is constant and the goal is to optimize the injection scenarios: we are able to run an important amount of simulations and train an accurate machine learning model. However, it is not well-suited for handling well location optimization.

Regarding the local hybrid Newton's method, the 1D study highlighted the impact of the trade-off between small and large local predictive domains and that it is possible to learn the solution in saturation for a wide range of well locations. When moving to a 2D case, developing a predictive model for different well locations is challenging due to sampling issues. Therefore, we proposed two hybrid initial guesses, the first one using the Fourier Neural Operator to predict the saturation in the near-well region for a constant well location, and the second one using a fully implicit solver on a local domain, which can be seen as a domain decomposition method. We applied those two approaches to three test cases, including out of distribution test cases for the FNO method. Overall, our results showed that the domain decomposition initial guess performed equal or better for all test case compared to the machine learning initial guess while being able to handle any well location. Moreover, we distinguished two different behaviours: the first one corresponds to an initialization in Newton's quadratic convergence zone, and the second one to a 'half-way' initialization. In contrast, the global approach initialization is always in Newton's quadratic convergence zone resulting in better performances. Nevertheless, for the local approach when moving to more difficult cases (3D geometry, complex physics, etc.), we expect the machine learning initial guess to infer significantly faster and hence provide an improvement with respect to the domain decomposition initial guess.

Therefore, we recommend using the global hybrid Newton's method for problems with constant well location and the goal of optimizing the injection scenarios. On the other hand, when the goal is to optimize the well locations, we recommend using the local hybrid Newton's method as it is more flexible. Moreover, there are some avenue of research that we wish to emphasize for both the global and local hybrid Newton's strategies.

7.2 Perspectives

7.2.1 Local approach - Generalized well model and multi-well simulations

In this thesis, we did not assess the potential of a machine learning model for the 2D local hybrid Newton's method with a variable location. Indeed, the introduction of a variable well location leads to sampling challenges that complicate dataset generation, particularly considering the prohibitively high cost of generating each data point. Therefore, a perspective would be to develop a generalized well model through a synthetic and general dataset of small domains around a well. Indeed, generating a synthetic dataset by solving only small local problems should reduce the cost of the dataset generation. To this end, we have initiated the development of several ideas and will briefly present the underlying intuition.

The primary challenge lies in generating realistic implicit pressure fields. When working with a constant saturation, the implicit pressure field is entirely determined by the boundary conditions and the well injection parameter. Sampling the well injection parameter is straightforward, as it consists of a single spatially independent value.

Regarding the boundary pressure, one possibility is to explore the following workflow: within a square local domain composed of four boundary groups (left, up, right, down), we randomly sample boundary pressure faces from a variable number of boundary groups (between 2 and 4). Subsequently, we apply a correction to ensure physical consistency among the sampled points. Following this, we perform linear interpolation between the sampled pressures to obtain a boundary pressure for each face. Finally, we consider all rotations of this pressure configuration and, for each of them, launch the implicit pressure solver. Figure 7.1 illustrates an example of the implicit pressure in a local domain obtained through this workflow. It is to note that the local domain is circumscribed to a virtual circle. In the two left figures, the green and orange points correspond, respectively, to the same points on the circle and the local square domain.

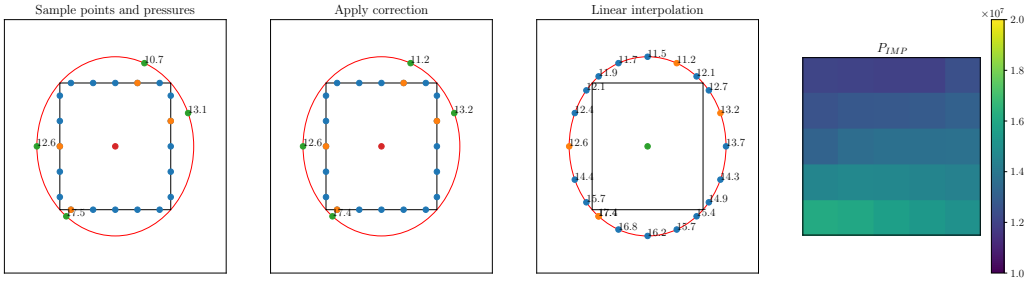


Figure 7.1: Example of pressure obtained through the developed workflow.

7.2.2 Moving towards realistic problems

7.2.2.1 Integration in an industrial software

We applied our methodology on a 2D synthetic test case and running the simulations of a library that we developed on our own. An objective would be to challenge our results by incorporating in a more sophisticated 3D reservoir simulation software such as OPM (Open Porous Medium) [2].

7.2.2.2 Working with different scales

We applied the Hybrid Newton's method on a specific reservoir geometry, the S (small) size SHPCO_2 mesh in our case. However, Neural Operators should have an invariance to discretization property. Therefore, it should be possible to learn at a specific grid resolution and then infer at a different resolution. This can be interesting in the upscaling or downscaling process, i.e a process to infer high or low resolution information from low or high resolution properties. An idea could be to generate a low resolution database, train a neural operator in a supervised manner and then infer on a finer grid resolution. The low resolution database should be cheap to generate and may lead to non-negligible speed-up for the handling of well events on a higher resolution grid.

7.2.2.3 Taking heterogeneities into account

In this work, we did not consider small scale heterogeneities. Heterogeneities denotes the non-uniform distribution of properties such as the permeability of the porosity within subsurface, which can lead to complex and variable behaviour. This is a fundamental property in geosciences, any realistic model

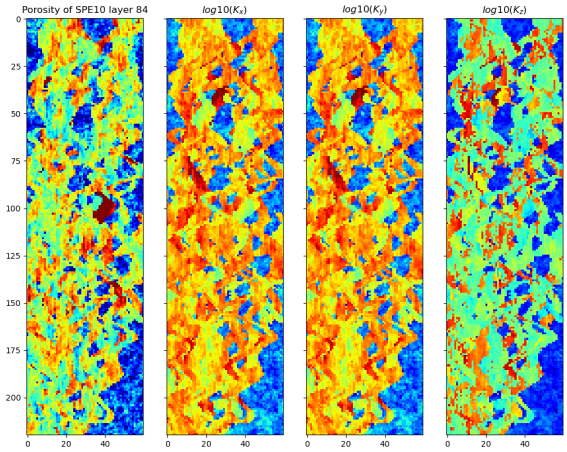


Figure 7.2: Example of permeability field (K) at the layer 84 from the SPE10 comparative solution project.

should take into account the spatial and/or time variability of the porosity for example. Therefore, an essential step should be to take into account small scale heterogeneities in the presented test cases and apply the hybrid Newton's method. Another test case including heterogeneities is the SPE10 [1] comparative solution project.

Bibliography

[1] M. A. Christie and M. J. Blunt. Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques. *SPE Reservoir Evaluation & Engineering*, 4(04):308–317, 08 2001.

[2] Atgeirr Flø Rasmussen, Tor Harald Sandve, Kai Bao, Andreas Lauser, Joakim Hove, Bård Skaflestad, Robert Klöfkorn, Markus Blatt, Alf Birger Rustad, Ove Sævareid, Knut-Andreas Lie, and Andreas Thune. The open porous media flow reservoir simulator. *Computers & Mathematics with Applications*, 81:159–185, 2021. Development and Application of Open-source Software for Problems with Numerical PDEs.