PHYSICS INFORMED NEURAL NETWORKS FOR SURROGATE MODELLING AND INVERSE PROBLEMS IN GEOTECHNICS

Presented on the 25th of June 2021
in the Civil Engineering Department
Geo-Energy Laboratory (GEL)
École Polytechnique Fédérale de Lausanne

Nils Frédérik Heiden Olsen

Prof Brice Lecampion, master project supervisor
Prof Stéphanie Chaillat, external expert

Lausanne, EPFL, 2021

Acknowledgements

I would like to thank Prof. Brice Lecampion from the GEL Labaoratory at EPFL for supervising my master project this Spring. You were always present and put all the resources when required, your advises were really valuable. Thank you very much also for being a part of my studies at EPFL and in my master for two years already. It led to many great works, events and opportunities. I would like also to thank my brother, parents, family and friends for the support this semester and specifically Mr. Antoine Spahr with his expertise in Deep Learning.

Lausanne, June 25th 2021 N. O.

Abstract

Finite elements methods (FEMs) have benefited from decades of development to solve partial differential equations (PDEs) and to simulate physical systems. In the recent years, machine learning (ML) and artificial neural networks (ANN) have shown great potential to approximate such systems. Combining sparse or even no data, physics informed neural networks (PINNs) can be trained simultaneously on available data and the governing differential equations to fit a specific model or to compute the solution of an ordinary and partial differential equations. This master project focuses on the implementation of ANN models in order to predict the evolution of an hyperbolic PDE that is the acoustic wave equation in geotechnic problems. Different model architectures and training scenarios are implemented to create a surrogate model that will predict from the first initial wavefields the acoustic propagation. The U-Net model trained in a semi-supervised approach with a physical loss yields promising results, the lower average RMSE and the higher SSIM. This model is able to extrapolate the acoustic propagation on data not shown during its training but lacks of robustness when predicting wavefields at larger time than the ones provided for its training.

Key words:
Finite Element Methods
Artificial Neural Networks
Physics Informed Neural Networks
Deep Learning

Contents

Acknowledgements **i**

Abstract **iii**

List of figures **vii**

List of tables **ix**

1 Introduction **1**
1.1 Contributions 3

2 Literature review **5**

3 Theory **9**
3.1 Fully Connected Neural Networks 9
3.1.1 Fully Connected Neural Networks and physics 11
3.2 Deep convolutional neural networks 13
3.2.1 Deep convolutional neural networks and physics 15

4 Experiments **19**
4.1 Artificial Data 19
4.2 1D Burgers' Equation 20
4.3 Acoustic Wave Propagation 21
4.3.1 Wave equation 21
4.4 Salvus : Finite Element Solver in Python 22
4.4.1 Acoustic wave propagation in Salvus 22
4.4.2 Presentation of the cases 23
4.4.2.1 Homogeneous Case 24
4.4.2.2 Layered Heterogeneous Case 25
4.4.2.3 Crack Heterogeneous Case 25
4.4.3 Simulations outputs from Salvus 27

5 Methodology **29**
5.1 Training Curve fitting 29
5.2 Training the 1D Burgers' Equation 30

5.3 Training Acoustic Wave Equation . . . 33
5.3.1 Models . . . 33
5.3.1.1 MultiScale . . . 34
5.3.1.2 Encoder-Decoder . . . 35
5.3.1.3 U-Net . . . 35
5.3.2 Model Selection . . . 36
5.3.2.1 Loss functions . . . 36
5.3.2.2 Scenarios Tested . . . 37
5.3.2.3 Datasets . . . 39
5.3.2.4 Fixed Parameters . . . 41
5.3.3 Surrogate modelling . . . 41

6 Results and discussion **43**
6.1 Curve Fitting . . . 43
6.2 Burgers 1D Equation . . . 45
6.3 Acoustic Wave Propagation . . . 46
6.3.1 Predictions . . . 47
6.3.2 Model Selection . . . 49
6.3.2.1 Homogeneous Domain . . . 50
6.3.2.2 Heterogeneous Domain . . . 58
6.3.3 Surrogate Modelling . . . 65

7 Conclusion **71**

A Theory **75**

B Experiments **77**

C Methodology **79**
C.1 MultiScale parameters . . . 79
C.2 Encoder - Decoder parameters . . . 80
C.3 U-Net parameters . . . 81

D Results **83**
D.1 Acoustic - Homogeneous Domain . . . 83
D.2 Acoustic - Heterogeneous Layered Domain . . . 89

Bibliography **98**

List of Figures

3.1 Fully Connected Neural Network 11
3.2 Fully Connected Neural Networks with Physical Loss 12
3.3 Encoder-Decoder Network Structure 14
3.4 Data Driven Training Scheme for a CNN 15
3.5 Physics constrained training scheme for a CNN 17

4.1 Artificial Data Presentation 20
4.2 Ricker wavelet 20 Hz Amplitude and Power Spectrum 22
4.3 Homogeneous domain with one source in the middle 24
4.4 Homogeneous domain with sources along the edges 24
4.5 Layered heterogeneous domain with one source in the middle 25
4.6 Layered heterogeneous domain with sources along the edges 26
4.7 Small cracked (Crack 1) heterogeneous domain with sources along the edges . 26
4.8 Big cracked (Crack 2) heterogeneous domain with sources along the edges . . . 27
4.9 Outputs for the heterogeneous layered domain with one source in the middle . 28
4.10 Outputs for the homogeneous domain with one source in the middle 28

5.1 Multi Scale Network Architecture 34
5.2 Encoder Decoder Architecture 35
5.3 U-Net Architecture 36
5.4 Auto-Regressive (AR) training scheme 39
5.5 Sample of the homogeneous dataset used for training 40
5.6 100th sample of the homogeneous dataset 41

6.1 Artificial Experiments Results - Square, Circle and Sinus 44
6.2 Histograms of errors for the artificial simulations 45
6.3 PINN Burgers 1D Equation 46
6.4 Data Driven Burgers 1D Equation 47
6.5 Data Driven and PINN Comparison for the Burgers 1D Equation 48
6.6 Homogeneous domain - Wavefields predictions for the U-Net PINN RES scenario 52
6.7 Homogeneous domain - Training - Wavefields error along the x-axis for the U-Net PINN RES 53
6.8 Homogeneous domain - Training - Predicted pressure and Error at the receivers for U-Net PINN Res 53

6.9 Homogeneous domain - Training - Predictions receivers errors 54
6.10 Space U-Net receivers 55
6.11 RMSE for the time extrapolation on the homogeneous domain 56
6.12 Receivers errors for the time interpolation 57
6.13 Heterogeneous domain - Wavefields predictions for U-Net PINN RES 60
6.14 U-Net PINN RES receivers errors on the Heterogeneous Layered domain 61
6.15 Initial wavefield along the x and the y axis for Salvus 63
6.16 Salvus and Finite Time Difference wavefields for different time steps 63
6.17 Mean Max Error between Salvus and FDM 64
6.18 Train and test sources for the simulations 65
6.19 Predictions for Event 5 for the surrogate models 66
6.20 Evolution of the RMSE for different number of epochs 67
6.21 Metrics for the heterogeneous small and big cracks domain 68
6.22 Error at the receivers locations for the experiments with sources placed on the edges 69

D.1 Homogeneous domain - Wavefields predictions 85
D.2 Metrics computed for the homogeneous domain 86
D.3 Metrics Space Extrapolation - Homogeneous Domain 87
D.4 Metrics Time Extrapolation - Homogeneous Domain 88
D.5 Metrics Training - Heterogeneous Layered Domain 90
D.6 Metrics Space Extrapolation - Heterogeneous Layered Domain 91
D.7 Metrics Time Extrapolation - Heterogeneous Layered Domain 93

List of Tables

4.1 Mesh Characteristics 23

5.1 Presentation of the number of known values and collocations points for each simulation 30
5.2 Description of the dataset used in the model selection 40
5.3 Description of the dataset used in the surrogate modelling 42

6.1 Average RSME on the homogeneous domain for the model selection 50
6.2 Average SSIM on the homogeneous domain for the model selection 51
6.3 Average RSME on the heterogeneous domain for the model selection 58
6.4 Average SSIM on the heterogeneous domain for the model selection 58
6.5 Mean Metrics over the 150 predicted timesteps for the train and test simulations 65

B.1 Salvus - Homogeneous domain parameters 77
B.2 Salvus - Layered heterogeneous domain parameters 77
B.3 Salvus - Cracked heterogeneous domain parameters 78

C.1 MultiScale neural network parameters 79
C.2 Encoder neural network parameters 80
C.3 U-Net neural network parameters 81

D.1 Results Training Data - Homogeneous Domain 83
D.2 Results Space Extrapolation - Homogeneous Domain 84
D.3 Results Time Extrapolation - Homogeneous Domain 84
D.4 Results on the layered training data 89
D.5 Results on the layered space extrapolation data 89
D.6 Results on the layered time extrapolation data 92

1 Introduction

Over the past decades, Finite Elements Methods (FEMs) were used to solve physical problems. The domain is discretised with a mesh and the governing Partial Differential Equations (PDEs) are solved with an iterative time stepping scheme. For engineering problems with fully known physics, their solutions can be modelled with differential equations describing conservation laws (Tartakovsky et al., 2020) such as the Navier-Stokes equation for fluid simulations or the elasto-dynamics equation for elastic simulations for example.

There was a huge increase in the computational power and the data storage capacity since the paper from Bengio and Lecun, 1997 was published. Many fields in machine learning (ML) have been revolutionised and problems using Artificial Neural Networks (ANNs) such as in image classification and other computer vision tasks, speech recognition and autonomous driving just to cite a few could widely benefits from this revolution. This increased the interest and the availability of data and open source softwares for machine learning applications such as Sklearn (Pedregosa et al., 2011) Tensorflow, (Abadi et al., 2016), Keras (Gulli & Pal, 2017) and PyTorch (Paszke et al., 2019). The traditional ML problems use deep neural networks presented in LeCun et al., 2015 are trained using a lot of data. In those images classification tasks or autonomous driving, the large amount of data allows the networks to discover patterns in the data and the trained models can predict the system behaviour under wide range of conditions. This data driven framework has made advancements in earthquakes detection (Ross et al., 2019), fluid mechanics and turbulence modelling (Haghighat & Juanes, 2021).

Despite huge advance in those areas, deep learning has not yet been widely used in the field of scientific computing for engineering applications. Predictive problems with not fully known physics and sparse data still present a significant challenge. In most engineering applications data acquisition is an expensive and time demanding task. Data are often, if not always scarce. However some physics are present in these problems and can be used to reduce the dependency on observational data to understand the problem (Karimpouli & Tahmasebi, 2020).

In this small data regime, if the state-of-the-art machine learning techniques (e.g., deep,

convolutional or recurrent neural networks) are trained using only the available data, they will fail to provide any guarantees of convergence and lack robustness. The networks would fail to converge mostly due to the complexity of the system and un-optimised parameters (Geneva & Zabaras, 2019). Because of the high spatial heterogeneity of natural systems, sparse measurements do not allow to cover the whole distribution of the properties (Tartakovsky et al., 2020). For example, it is impossible to accurately learn the whole spatial distribution of porosity in a sub-surface system from few data only.

Therefore combining the fundamental physical laws and domain knowledge directly in the ML models can provide informative priors that is, strong theoretical constraints and inductive biases on top of the observational ones (Karniadakis et al., 2021) to improve the performances. Inductive biases are referred as assumptions incorporated to the ML model under the form of physical laws and mathematical constraints that the predictions will guarantee. The process from which knowledge coming from observational, empirical, physical or mathematical descriptions of the problem can be used to improve the performance of a machine learning algorithm has emerged as a new sub-field in the machine learning under the name of Physics-Informed Machine Learning (PIML)(Raissi et al., 2019), Physics-Constrained Machine Learning (PCML) (Y. Zhu et al., 2019) or Scientific Machine Learning (SciML) (Lu et al., 2020).

A recent example illustrating this new learning technique is the family of Physics Informed Neural Networks (PINNs). This is a class of Deep Learning (DL) networks that integrate data and abstract mathematical operators such as Partial Differential Equations (PDEs). Approximating PDEs is normally the task of the finite difference method (FDM) and the finite element method (FEM) which are mesh-based methods. Solving PDEs in the weak or in the strong form with deep learning thanks to prior knowledge on the domain or mathematical constraints motivates the development of PINNs. More robust models can be implemented that are less sensitive in the presence of imperfect data (such as missing or noisy values, outliers). They provide accurate and physically consistent predictions, even for extrapolatory/generalisation tasks (Karniadakis et al., 2021).

Surrogate models are used to provide a fast approximate model that can imitate a standard numerical solver at a significantly reduced computational cost (Geneva & Zabaras, 2019). In many situations, such as optimisation or inverse problems, a large number of repeated simulations are required, prioritising the computational efficiency of the numerical simulator. Compared to the traditional mesh-based surrogate methods (Pfaff et al., 2020), DL models and especially PINNs benefit from the advantage of the automatic differentiation and a mesh-free approach to approximate physical systems with known physics. Such surrogate models could break the curse of dimensionality to approximate PDEs (Lu et al., 2020).

In PINNs, the networks are trained simultaneously on available data and the governing differential equations. Once the models are trained, they can be used as a surrogate model to approximate a physical system but also for uncertainty quantification and propagation in problems governed by PDEs. From (Haghighat & Juanes, 2021) the authors state that the three

main areas covered by PINN are (i) model fitting, (ii) discovery of the solutions for ordinary and partial differential equations and (iii) model inversion (physical parameter(s) identification). The networks can be trained to represent the states, some space dependent coefficients and some constitutive relationships. Despite numerous public databases, the volume of useful experimental data for complex physical systems is limited. The specific data-driven approach to the predictive modelling of such systems depends crucially on the amount of data available and on the complexity of the system itself (Karniadakis et al., 2021).

1.1 Contributions

This master project focuses on the implementation of ANN models in order to predict the evolution of an hyperbolic PDE that is the acoustic wave equation in geotechnic problems. Multiple model architectures and training scenarios combining different losses are implemented to create a surrogate model that will predict from the first initial wavefields the acoustic propagation. Comparing all the models and the losses allows to create a baseline actually missing in this field of research and different ideas will be proposed in the end to improve those models.

The most notable contributions from the master project are :

- Comparison of multiple networks architecture and training scenarios (Data driven or physics-constrained) to determine the best model to predict the acoustic wave equation (hyperbolic PDE).
- Report of metrics RMSE, SSIM and so on to create a baseline that can be extended in further works.
- Use the best performing model in a surrogate model to simulate laboratory experiments.
- Creation of a dataset in Salvus for multiple domains : homogeneous, heterogeneous layered and cracked with multiple sources locations

The objective of the surrogate model is to simulate wave propagation in a heterogeneous medium where a crack is propagating when fluid is injected. Indeed, the objective is to reproduce the experiments performed at the Geo-Energy Laboratory at EPFL described in Liu et al., 2020. In this experiment, fluid is injected though a borehole which induce a crack and sensors (sources and receivers) emit and record acoustic emission at different time to model the fracture propagation. Having a surrogate model will allow to run multiple simulations faster than the conventional FEM in order to locate the crack. For example many simulations could be run with the help of heuristics to find the correct configuration of the crack along a 2D plan. All the code and the resources can be accessed online : https://github.com/nfholsen/PDM_PINN

2 Literature review

Machine learning and deep learning models can be used in PINNs to approximate a physical system modelled with differential equations. Learning ordinary or partial differential equations (ODEs and PDEs respectively) using a constraint based loss functions is not a new domain of research. The earliest related works are more than two decades old as presented in Geneva and Zabaras, 2019 and Mo et al., 2019.

Those days in order to solve the PDEs, discretisation methods are used such as finite difference or finite element and iterative time stepping schemes are used to perform the numerical simulations. When large 3D solutions have to be computed then up to billions of parameters in the mesh must be computed and the computational time explodes. Indeed, one needs to run multiple simulations for calibration and tuning of the numerical model and the simulations have to be re-run from scratch for small variations in the parameters.

With their potential to learn hierarchical representations, deep learning techniques have emerged as an alternative to numerical solvers by offering a desirable balance between accuracy and computational cost. This literature review presents the recent researches covering the field of PINNs to solve PDEs and then focuses on the methodologies used to simulate wave equation with deep learning.

Two families of models are used in PINNs: Fully Connected Neural Networks (FCNNs) and Convolutional Neural Networks (CNNs). In the framework presented by Raissi et al., 2019, deep fully connected neural networks are used in a supervised learning methodology to implement a data-driven solution and a data driven discovery of partial differential equations. Collocations points are sampled on the domain and the predictions from the model must satisfy the PDE. On different problems (Schrodinger; Allen–Cahn; Navier–Stokes; Burger's and Korteweg–de Vries equation) the residuals are minimised over the domain and the boundaries/initial conditions are known.

FCNNs were first used and already showed promising and impressive performances for surrogate modelling of forward models with high-dimensional input and output fields (Geneva

& Zabaras, 2019). Indeed, by providing the time and space parameters as inputs, automatic differentiation can be used to compute the partial derivatives required to compute the PDEs residuals as shown in Karniadakis et al., 2021. Using only a few points for training, the solution can be computed anywhere on the domain. This enables a mesh free solution to evaluate the PDEs anywhere on the domain while being trained on only a few points.

The same kind of networks have been used in Moseley et al., 2020, Sun et al., 2020, Tartakovsky et al., 2020, Shukla et al., 2020, Rao et al., 2020, Karimpouli and Tahmasebi, 2020 or even in Python libraries such as DeepXDE in Lu et al., 2020 and SciANN in Haghighat and Juanes, 2021. However for these kind of models, Geneva and Zabaras, 2019 states that if the initial conditions or the boundary conditions or the properties on the domain change, then the model must be retrained. Similar statements were made in Moseley et al., 2020. The authors implemented a FCNN to predict the wave propagation and their model has to be retrained for each new media/domain.

To allow the computation anywhere on the domain without retraining the model and better generalisation capabilities, an other approach is to treat the inputs and outputs as images and use convolutional neural networks instead of fully connected neural networks. In Geneva and Zabaras, 2019, the model performs an image-to-image regression task in order to predict from an initial field, multiple output fields for the equations.

Surrogate models are built in Mo et al., 2019 and in Y. Zhu et al., 2019 using deep convolutional encoder-decoder networks to model the uncertainty of dynamic multi-phase flows and predict the Darcy's flow respectively. To my best knowledge, very few papers dealt with PINNs for 3D domain and only Mo et al., 2020 implemented a convolutional adversarial auto-encoder to find the parameters of a non-Gaussian conductivity fields in 2D and 3D.

In the work of Geneva and Zabaras, 2019, the authors introduce a deep auto-regressive dense encoder-decoder to predict transient PDEs and they extend their model to a Bayesian framework which allow uncertainty quantification (UQ). Regarding the UQ which is not the scope of this work, one wants to compute the distribution of parameters of great importance and its sensitivity to noise in the input data.Y. Zhu et al., 2019 uses Gaussian processes for parameter inversion and the Kullback-Leibler divergence between the model predictive density and the reference conditional density is minimised.

Deep learning models have been also applied to solve the wave equation. Especially for geophysics problems, PINNs have been applied to various inverse problems. The work in Li et al., 2020 estimates subsurface ground properties, such as rock porosity and permeability. Seismic data was used in neural networks and full-waveform inversion models to simulate subsurface flow processes and rock physics. In W. Zhu et al., 2021, the authors demonstrated that combining DNNs and numerical PDE solvers PINNs are capable of solving a wide class of seismic inversion problems, such as velocity estimation, fault rupture imaging, earthquake location and source–time function retrieval.

In Moseley et al., 2020 a fully connected deep neural network was implemented to solve the wave equation and to simulate its propagation. The authors used the PDE of the 2D acoustic wave equation to enforce the physics during the training when the initial conditions became silent (the source term is 0 on the domain). They trained their FCNN with a physical loss on an earth realistic domain with many source locations. The authors were able to predict the wave propagation on a new location close to the ones provided during the training for this experiment. Especially for the wave equation, the authors noticed that deep neural networks typically rely entirely on their training data and therefore perform poorly outside of it. This explains why so many experiments were required for the training of their earth-realistic model. However this was not the case for Raissi et al., 2019 as their deep learning framework for surrogate modelling and inverse problems showed good generalisation capabilities outside the training data but on other problems than the acoustic wave equation.

In Shukla et al., 2020, the wave equation was used to detect cracks as the wave velocity is a good proxy for the defects. The authors identified parameters by allowing the wave velocity to be space dependent on the domain and when the velocity had a brutal change the crack could be detected. Karimpouli and Tahmasebi, 2020 uses the Gaussian process for parameters inversion too to inverse the wave velocity parameter first using the acoustic 1D wave equation and then for the 2D elastic wave equation. Using PINNs the authors were able to invert also the compressional and shearing waves velocity (V_p and V_s) as well as the density ρ of the homogeneous medium.

The Eikonal equation was solved in Smith et al., 2020 to predict the velocity in different soil layers. The equation characterises the first time of arrival in 3D heterogeneous medium given a source-receiver pair. In Rao et al., 2020, the authors implemented a PINN to model elastodynamics outputs and more particularly the elastic wave equation in fully confined domain, infinite domain and semi-infinite domain with "hard" enforced boundary conditions where most of the papers enforced the boundary conditions in a "soft" way. The displacement and the stress fields are computed by the PINN.

Other works aimed at simulating the wave equation propagation by using neural networks as surrogate models using a data driven approach. A data driven approach means that no PDE is used in the neural networks. In Weiqiang et al., 2017 and Alguacil et al., n.d. the authors used a CNN architecture developed by Mathieu et al., 2016. Originally used to predict the next video-frame, the CNN architecture down-samples the inputs to encode the video frames in a latent space which are recursively up-sampled and combined with the inputs frames to predict the next frame. Such model architecture can be used to predict wavefields as they can be represented as images. Their training doesn't rely on any physics and the inputs are the 4 previous elasto-dynamics parameters for heterogeneous media. For each of the 4 previous frames (wavefields) 5 inputs fields (x and y displacements and velocities and also the density field) are used and the outputs are the 5 fields for the next time-frame. Their papers lack metrics and provide only the value of the losses and some predictions for the model in case of Weiqiang et al., 2017 and some predictions along the line as well as the evolution of the

Root-Mean-Squared-Error (RMSE) in Alguacil et al., n.d. It is not straightforward to assess the performance of their models on an other dataset. In this paper a loos function that combines the Mean-Squared-Error (MSE) between the true and the predicted wavefield and the Gradient Difference Loss (GDL) is used. The goal of the GDL is to enforce the model to predict sharp edges as introduced in Mathieu et al., 2016.

In Sorteberg et al., 2018 and Fotiadis et al., 2020, the authors solve the surface wave propagation governed by the Saint-Venant (SV) equations with CNNs. Despite describing the wave propagation in shallow depth, the wavefields can be visually compared to the acoustic waves equations propagating in underground earth media. Sorteberg et al., 2018 initially proposed a model combining a convolutional with a recurrent neural network trained with no other knowledge of the operating environment than the visual inputs being the wavefields. Fotiadis et al., 2020 introduced three new model architecture to improve the wavefield predictions and two of these architectures lead to improved accuracy in long-term prediction over previous attempts made by Sorteberg et al., 2018. One of the architecture used and proven to be really effective at predicting the wavefields is the U-Net. The U-Net has been used for optical flow estimation (Fischer et al., 2015) and also in physical systems to predict the sea temperature (de Bézenac et al., 2017) and to accelerate the simulation of Navier-Stokes equations (Thuerey et al., 2020).

3 Theory

Machine learning (ML) is a method in which an algorithm improve automatically through experience and the use of data. The goal of machine learning is to capture pattern in data from a population to make predictions. The main inference tasks type can the be defined as Classification, Regression and Density Estimation. In machine learning one can find methods such as decision trees, logistic regression, support vector machine (SVM), clustering methods and deep learning (DL) with all the artificial neural networks (ANNs). The ANNs are often referred as Deep Learning since the models contain a lot of layers making them deep. The ANN models consist of fully connected neural networks, convolutional neural networks, recurrent neural networks, residual networks and reinforcement learning just to cite a few. This section will present how Machine Learning and more specifically how Deep Learning models are trained and a physical mathematical constraint can be implemented in a neural network to create a PINN and solve a certain PDE.

Neural Networks can be trained in a supervised or unsupervised fashion. For example, in the framework presented by Raissi et al., 2019, the neural networks are trained to solve a task and in order to do so supervised learning is used. The model will predict an output $\hat{\mathbf{y}}$ given a certain input $\mathbf{x}$. The predicted output is compared to the true output $\mathbf{y}$ and the loss is computed. The loss represents the error between the predictions and the true value for the actual model parameters. The model will recursively update the weights via automatic differentiation and back-propagation algorithm during the training phase. The principle of learning is to find the best parameters in the model minimising the loss.

3.1 Fully Connected Neural Networks

Predicting physical quantities with a neural network is associated with the task of regression presented above. Indeed one wants to predict continuous quantities Y (outputs) based on values taken by the features X (inputs). In Haghighat and Juanes, 2021, Mo et al., 2019 and Goodfellow et al., 2016, a single-layer forward neural network with inputs $\mathbf{x} \in \mathbb{R}^m$, outputs

$\mathbf{y} \in \mathbb{R}^n$ and d hidden units is constructed as :

$$\mathbf{y} = \mathbf{W}^1 \sigma(\mathbf{W}^0 \mathbf{x} + \mathbf{b}^0) + \mathbf{b}^1 \tag{3.1}$$

with $\mathbf{W}^0 \in \mathbb{R}^{d \times m}, \mathbf{W}^1 \in \mathbb{R}^{n \times d}$ are the weights and $\mathbf{b}^0 \in \mathbb{R}^d, \mathbf{b}^1 \in \mathbb{R}^n$ are the biases. σ is the activation function. A single forward network can approximate any measurable function independently of the size of input features m or the activation function. This is a useful property when one wants to solve PDEs. The network is expected to match them due to the universal approximation theorem for the neural network which states that :

A feed-forward network with a linear output layer and at least one hidden layer with any "squashing" activation function (such as the logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired nonzero amount of error, provided that the network is given enough hidden units". Goodfellow et al., 2016

So technically if the neural network is trained with sufficient data, an MLP with only one layer and sufficient training data could match every function. We can construct a general L-layers fully connected neural network as composition of Σ^i functions :

$$\mathbf{y} = \Sigma^L \circ \Sigma^{L-1} \circ \cdots \circ \Sigma^0(\mathbf{x}) \tag{3.2}$$

with σ^i being an activation functions. This allows to create nonlinear transformations. Some common activation functions are:

- ReLu : $\hat{x} \rightarrow \hat{x}^+$
- sigmoid : $\hat{x} \rightarrow 1/(1+e^{\hat{x}})$
- tanh : $\hat{x} \rightarrow (e^{\hat{x}} - e^{-\hat{x}})/(e^{\hat{x}} + e^{-\hat{x}})$

In case of a two dimensional input domain, the inputs would be (x_i and y_i) pair and the output the value to predict. The number of neurons for the output is one since only one value has to be predicted. If the model was performing a multi-class classification task for example then the number of neurons in the last layer would correspond to the number of classes. Also, if two values would have to be predicted (for example $\hat{u}$ and $\hat{v}$) for each x and y location then there would be two outputs neurons. It is important to notice than for every hidden layers (all layers except the first and the last) "activation functions" are used to create non-linearity in the model. If it was not the case, then the whole network would be a combination of affine mapping and the advantages of using numerous hidden layers will be lost.

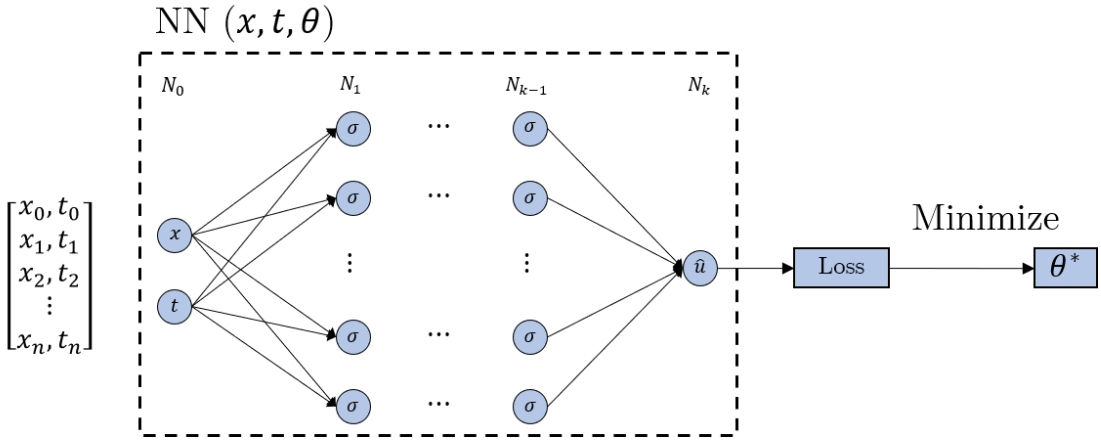


Figure 3.1 – Illustration of a Fully Connected Neural Network which takes as inputs the location and the time (x and t respectively) and outputs the predicted velocity $\hat{u}$.

3.1.1 Fully Connected Neural Networks and physics

A physics informed neural network incorporate both the measurements or data available as in a FCNN but also the partial differential equation. The PDE is embedded in the loss function of the neural network using automatic differentiation. The PDE that can be incorporated are integer orders PDEs, fractional PDEs or stochastic PDEs as presented in Karniadakis et al., 2021.

Consequently, the loss function includes errors that can be written as a combination of the network outputs (MSE_u), residuals of the differential equation (MSE_v) and initial (MSE_0) and boundary (MSE_b) conditions (Karimpouli & Tahmasebi, 2020). The losses are a combination of the supervised learning from data measurements on the boundary, initial conditions and the domain if measurements are available and an unsupervised loss from collocation points sampled on the domain. The total loss is presented in the Equation 3.3 below. λ parameters can be used with the loss to regularise each loss and emphasise one more term more than the others for example. If none are used then all the loss terms accounts at the same proportion in the total loss.

$$MSE_{tot} = \lambda_u MSE_u + \lambda_v MSE_v + \lambda_0 MSE_0 + \lambda_b MSE_b \tag{3.3}$$

with :

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} (u_i - \hat{u}_i)^2 \tag{3.4}$$

$$MSE_v = \frac{1}{N_v} \sum_{i=1}^{N_v} (L_{\mathbf{x}} \hat{u}_i)^2 \tag{3.5}$$

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_u}(u_{0,i} - -\hat{u}_{0,i})^2 \tag{3.6}$$

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b}(u_{b,i} - \hat{u}_{b,i})^2 \tag{3.7}$$

in the equations, u, u_0, u_b, $\hat{u}$, $\hat{u}_0$ and $\hat{u}_b$ are the domain, initial and boundary values of the function and the predictions from the network respectively. A PINN uses the automatic differentiation to compute the partial derivative of the PDE. The $tanh$ activation function is used to create non-linearity between the hidden layers since it is indefinitely differential.

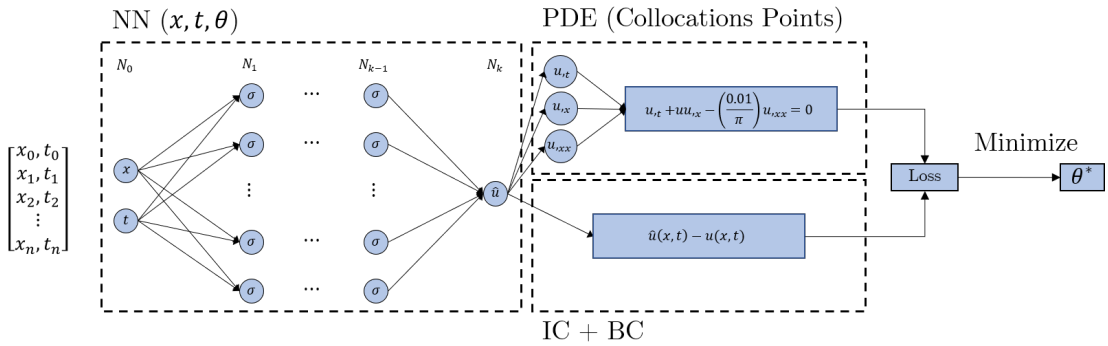


Figure 3.2 – Illustration of a fully connected neural networks with a physical loss term for the Burgers 1D Equation.

To illustrate, Figure. 3.2 presents the PINNs with the Burger 1D equation. The inputs are the locations and the times (x and t) and the output is the velocity $\hat{u}$ for that specific location. Automatic differentiation is used to compute the spatial and temporal partial derivatives, residuals are computed for the collocations points and the loss minimised. On the initial and boundary conditions, since real values are known because the values are fixed, they are used to enforce those conditions.

For the Burger 1D equation, the total loss function combining the known values of the boundary and initial conditions (u_{data}) and the residuals will be :

$$MSE_{tot} = \lambda_{data}\frac{1}{N_{data}}\sum_{i=1}^{N_{data}}(u_i - \hat{u}_i)^2 + \lambda_v\frac{1}{N_v}\sum_{i=1}^{N_v}\left(\frac{\partial \hat{u}_i}{\partial t} + \hat{u}_i\frac{\partial \hat{u}_i}{\partial x} - \frac{0.01}{\pi}\frac{\partial^2 \hat{u}_i}{\partial x^2}\right)^2 \tag{3.8}$$

For the one 1D seismic wave equation presented in Karimpouli and Tahmasebi, 2020, the only output is $u(x,t)$. The following loss function combining measurements and physical constraint with the wave equation is built with the first and second derivatives of the network

output with respect to x and t :

$$L = \frac{1}{N_u}\sum_{i=1} N_u(u_i - \hat{u}_i)^2 + \frac{1}{N_v}\sum_{i=1}^{N_v}\left(\frac{\partial^2 \hat{u}_i}{\partial t^2} - V^2\frac{\partial^2 \hat{u}_i}{\partial x^2}\right)^2 \tag{3.9}$$

With $\hat{u}$ the output of the network. For the surrogate modelling the velocity value V is known. To perform an inversion, the velocity is introduced as an unknown parameter to the network and while the network is training, V will be optimised and the $\hat{u}$ predictions will be conducted simultaneously.

3.2 Deep convolutional neural networks

If MLPs were built on large input signals (large images, high resolution signal, ...), the resulting models would have way too much parameters to estimate, thus the computational cost would be immense. That is why convolutional layers are used and convolutional neural networks (CNNs) are implemented to greatly reduce the number of parameters being learnt. It is based on the intuition that some small pattern can be found on multiple places. They are particularly suited for image processing because they exploit the spatially-local correlation of the data by enforcing a local connectivity pattern between neurons of adjacent layers (Goodfellow et al., 2016). Sparse connectivities are created and parameters are shared through the convolutional layers. A convolutional layer apply the same linear transformation on small local part of the input and preserve the spatial structure.

CNN models provide better generalisation than the MLPs because the kernels trained in the convolutional layers can detect some shapes such as edges, curves, regardless on their location in the inputs allowing a certain robustness to shifting. Therefore, if one wants to simulate the wavefield propagation when the source is located at an other location than its original training location it should have better results (Y. Zhu et al., 2019).

Hence CNN architecture comes in mind when working with wavefields because they can be considered as big images. One pixel represents the value at one specific location on the mesh for one specific time. Multiple wavefields can by stacked in the same principles as RGB images and can be used as inputs to train our neural networks. CNNs were implemented in Alguacil et al., n.d.; Fotiadis et al., 2020; Sorteberg et al., 2018; Weiqiang et al., 2017 to predict acoustic wavefields, in Mo et al., 2019; Y. Zhu and Zabaras, 2018; Y. Zhu et al., 2019 to solve flow problems and even in Mathieu et al., 2016 to predict the next frames of a movie.

A convolutional layer is composed of a series of convolution kernels which compute the feature maps. For a 2D image the feature map h containing trainable parameters is computed

as follow :

$$h^q_{u,v}(x_{u,v}) = \sigma\left(\sum_{i=1}^{k'_i}\sum_{j=i}^{k'_j}\mathbf{W}^q_{i,j}\mathbf{x}_{u+i,v+j}\right) \tag{3.10}$$

The output from a convolutional layer h consists of N_{out} feature maps. There are two important parameters for the convolutional layer : the padding which pad the border of the images with zeros to preserve the size and the stride which delimits the distance between two successive locations of the filter.

Using convolutional layers in a PINN means that the network can perform an image-to-image regression task by employing an encoder-decoder network structure. Such structure consists of encoding and decoding layers (see Goodfellow et al., 2016) and dense blocks (see Fig 3.3). In the dense block (Y. Zhu & Zabaras, 2018), the outputs of each layer are connected with all

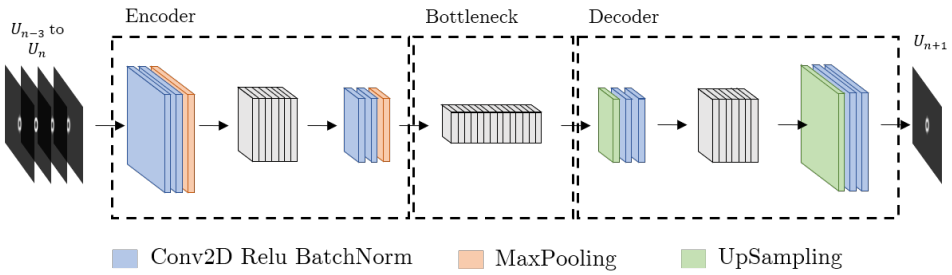


Figure 3.3 – The Encoder-Decoder Network Structure

successor layers. It means that the outputs from the $z(l)$ layer will receive the outputs from all previous layers. Hence the number of feature maps will grow at a rate K which is the number of feature maps (the grey stacked layers in Fig. 3.3) in each step.

The inputs in the encoder-decoder architecture go into successive transformations of coarse-refine processes. In the beginning high-level coarse features are extracted from the inputs using the encoder and then refined to the desired output image through a decoder. When the inputs reach the high-level coarse feature maps, a latent representation from the image is learnt (the bottleneck layers in Fig. 3.3). Then the decoder will reduce the number of feature maps and increase the size of the image until the output is reached. This type of architecture enhances information propagation through the network and reduces network parameters compared to fully connected neural networks. Thus one should be able to construct accurate surrogate models with limited training data.

An illustration of the training scheme for a CNN with the acoustic wave equation is presented in Figure 3.4. The model receives four input wavefields spanning from u_{t-4} to u_{t-1} and should predict the next wavefield $\hat{u}_t$. Because the CNN is trained in a supervised manner, the true wavefield u_t is also provided The loss is calculated based on the predicted and the true wavefield and the weights are updated. The process is repeated for many "pairs" of

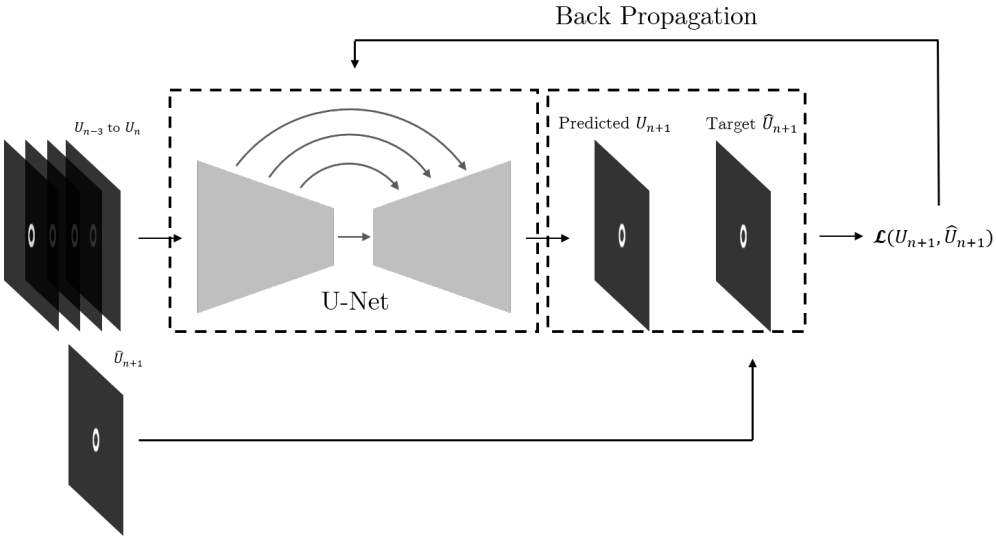


Figure 3.4 – Data driven training scheme for a CNN applied to the acoustic wave propagation.

inputs/output for many different timesteps for a certain number of iterations referred as epoch.

3.2.1 Deep convolutional neural networks and physics

Unfortunately, automatic differentiation cannot be used in the same way as in the physics constrained FCNN. The spatial and temporal information are not available anymore since only the only output available is a physical quantity (velocity, pressure ..) and not the spatial or temporal coordinates. Convolutional neural networks still require gradients in the governing PDEs to be numerically approximated. In Y. Zhu et al., 2019, the authors found that finite difference method based approximations to compute spatial gradients were very effective for elliptic PDEs. Thus it will be used in this work. For the temporal derivatives, Geneva and Zabaras, 2019 proposed that for PINN a single time-step can be optimised if the discrepancy between the model's predictions and the prediction of a discrete numerical time integration method ($T_{\Delta t}$) is minimised.

The authors state that the L_2 minimisation between the model's prediction and a consistent numerical time integrator method ($T_{\Delta t}$) can be interpreted as the L_2 minimisation of the residual of the discretised PDE. The proof presented in the appendix (see Sec.A) from Geneva and Zabaras, 2019, we see that the network can be trained with the predictions from the neural net and the true values computed with the spatial and temporal integration implemented with a finite time difference scheme (FDM) for a number of time-steps. For the homogeneous acoustic wave equation the finite time difference scheme used to calculate the next target

wavefield is given by :

$$c^2 \frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta x^2} + c^2 \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta y^2} - \frac{u^{n+1}_{i,j} - 2u^n_{i,j} + u^{n-1}_{i,j}}{\Delta t^2} = f^n_{i,j} \tag{3.11}$$

The parameters c^2, Δx^2, Δy^2 and Δt^2 are parameters used during simulations and known. All of them are provided to the surrogate modes. Moreover, if we consider the source term $f^n_{i,j}$ to be 0 then the initial conditions are silent.

The network is trained by minimising the residuals of the wave equation. This means that the neural network is forced to predict the next time step and it is computed with Equation 3.12 (Eq. 3.11 rearranged) by solving for $u^{n+1}_{i,j}$.

$$u^n_{i+1,j} = -u^{n-1}_{i,j} + 2u^n_{i,j} + c^2 \frac{\Delta t^2}{\Delta x^2}\left(u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}\right) + c^2 \frac{\Delta t^2}{\Delta y^2}\left(u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}\right) \tag{3.12}$$

The value computed from $C = c\frac{\Delta t}{\Delta x}$ is known as the Courant number. When one desire to model the wave equation in FDM the weak form of the PDE is implemented and the Courant-Friedrichs-Lewy (CFL) condition has to be respected if one wants to reach convergence. For the wave equations it establishes the following relationship :

$$c\Delta t \leq \Delta x \tag{3.13}$$

with c the velocity, Δt is the time step and Δx is the smaller mesh resolution. It means that the distance the wave travels in a single time step cannot exceed the size of a single element in the mesh. When the domain is heterogeneous, the scheme has to be adapted to handle the variable wave velocity. The term $q(\mathbf{x}) = c^2(\mathbf{x})$ is introduced and contains the velocities for all the domain constant for the entire propagation. The finite difference method to compute the acoustic wave propagation adapted from Eq. 3.11 becomes :

$$\begin{aligned} u^{n+1}_{i,j} = &- u^{n-1}_{i,j} + 2u^n_{i,j} \\ &+ \frac{\Delta t^2}{\Delta x^2}\left(\frac{1}{2}\left(q_{i,j} + q_{i+1,j}\right)\left(u^n_{i+1,j} - u^n_{i,j}\right) - \frac{1}{2}\left(q_{i-1,j} + q_{i+1,j}\right)\left(u^n_{i,j} - u^n_{i-1,j}\right)\right) \\ &+ \frac{\Delta t^2}{\Delta y^2}\left(\frac{1}{2}\left(q_{i,j} + q_{i,j+}\right)\left(u^n_{i,j+1} - u^n_{i,j}\right) - \frac{1}{2}\left(q_{i,j-1} + q_{i,j+1}\right)\left(u^n_{i,j} - u^n_{i,j-1}\right)\right) \end{aligned} \tag{3.14}$$

No damping is considered in those schemes. Waves can die out by two mechanisms. In 2D and 3D the energy of the wave spreads out in space- Energy conservation requires the amplitude to decrease. The damping is very small in many wave phenomena and thus only evident for very long time simulations. This makes the standard wave equation without damping relevant for a lot of applications (Langtangen & Linge, 2018).

Figure 3.5 illustrate how a CNN model will be trained with in a physic-constrained approach. The model has the same inputs as presented in Figure 3.4 to predict the next wavefield in

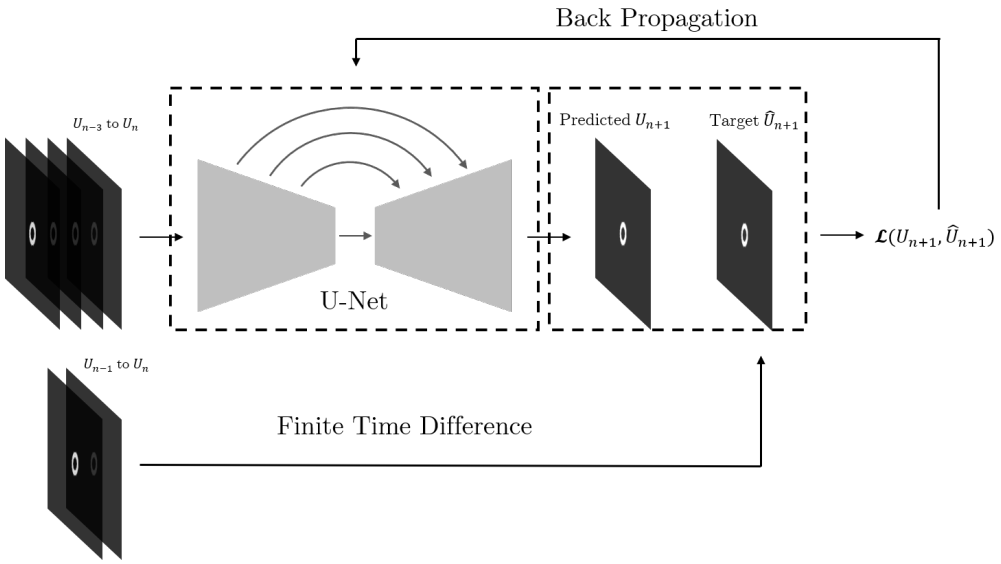


Figure 3.5 – Physics constrained training scheme for a CNN applied to a single time-step of the acoustic wave propagation.

case of the acoustic wave equation. The same methodology could be applied for others PDEs and also multiple input parameters (x velocity, y velocity, density and pressure ...) The true wavefield is computed using the finite time difference scheme.

To implement and enforce boundary conditions in this methodology, ghost cells are used (Geneva & Zabaras, 2019; Langtangen & Linge, 2018). An other loss term could have been added as in the FCNN with physics constrained training. The boundary conditions are enforced using circular padding in the convolutional layers.

4 Experiments

Deep learning models can be implemented using Python libraries specifically designed for solving Physics Informed Machine learning problems such as DeepXDE (Lu et al., 2020) a solver using Tensorflow, SciANN (Haghighat & Juanes, 2021) a wrapper using also Tensorflow and NeuroDiffEq a solver using PyTorch as back-end. However none of them proved to be efficient for the experiments conducted in this master project, especially for the acoustic wave propagation. Hence the models were developped from scratch.

To test the different methodologies, multiple experiments are conducted to assess the capabilities of neural networks as universal function approximation, to test that automatic differentiation can be used to compute the partial derivatives of PDEs and also that finite difference scheme can be used to efficiently train PINNs.

4.1 Artificial Data

A set a artificial data is generated to test the predictive capabilities of neural networks and their ability to match every function given a sufficient amount of training data and a sufficient predictive power. The predictive power here is referred at the amount of layers present inside the network. The goal is to test if a neural network can match complicated shapes. The domains are presented in Fig. 4.1.

The domains have to present some characteristics shown by the PDEs. The first is filled with squares. There is almost as many purple squares than yellow. For the network, there will be as many zeros to predict as ones and there is a clear pattern. Its task is to match the ones and zeros perfectly given the sharp transition between each region. Such sharp shapes are present in PDEs acoustic wave equation and Burgers equation for examples.

The second domain contains a circular shapes whose goal is to represent a wavefield. Two characteristics makes this domain interesting. The first one is the sharp edges between the artificial wavefield and rest of the domain and the second is the class imbalance. There are much more zeros than ones which represents also the wavefield propagation in a medium

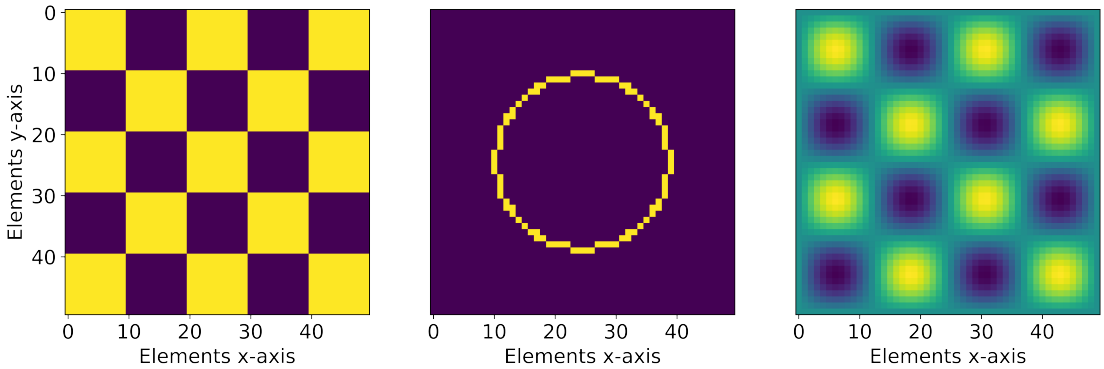


Figure 4.1 – The three artificial domains to test the predictive power of the neural networks.

where the pressure is different than zero only at the wavefield location. The last domain is the $sin(4\pi x)sin(4\pi y)$ function computed on the $[0,1] \cup [0,1]$ intervals. If the network fails to predict such shapes then it will be difficult to predict the evolution of the acoustic wave equation.

4.2 1D Burgers' Equation

The Burger 1D equation is a first order PDE in time and has been implemented in many papers Geneva and Zabaras, 2019; Lu et al., 2020; Raissi et al., 2019 and many GitHub repositories provide the python code to solve it using PINNs. The goal is to implement this problem and solve it with the PyTorch library in python. Since a lot of resources can be found online, we can implement a model with the same architecture and compare the results with the results obtained in Haghighat and Juanes, 2021. By doing so, we can assess that the PINN is working and validate its implementation.

The Burgers equation for this experiment is the following :

$$u_{,t} + uu_{,x} - \left(\frac{0.01}{\pi}\right) u_{,xx} = 0 \quad t \in [0,1], \quad x \in [-1,1] \tag{4.1}$$

The mesh size is 200x200 and the time-step for the FEM is $d_t = 0.005[s]$. The initial and the boundary conditions are :

$$u(t=0,x) = -\sin(\pi x), \quad u(t, x=\pm 1) = 0$$

4.3 Acoustic Wave Propagation

4.3.1 Wave equation

The wave equation is used to model a lot of different physical, natural and technological problems. Using waves, one can transport information along different scales. In geophysics, one can use waves to simulate earthquakes, estimate the subsurface structure, characterise the interior of the Earth and other planets with full waveform inversion or carry out non destructive testing (Moseley et al., 2020).

The wave equation as solved in Moseley et al., 2020, Karimpouli and Tahmasebi, 2020 and Weiqiang et al., 2017 belongs to the family of hyperbolic PDEs and those PDEs (also 1D Burger's equation presented in Sec. 4.2) can exhibit sharp discontinuities. They can be complex to solve for most traditional numerical methods (Sun et al., 2020). Wave propagation can be simulated with the acoustic and the elastic wave equation. The acoustic wave equation is different than the elastic. In the first one, a scalar is used for the pressure's magnitude where as in the second one a vector is used which gives the pressure and the directions. The acoustic wave propagation governs the propagation of pressure waves in liquid or gases. Thus shear waves or longitudinal waves motion are not possible.

Waves propagation can also be modelled with the Eikonal equation (bin Waheed et al., 2021; Smith et al., 2020) and the Helmholtz equation (Song et al., 2021). The Eikonal equation gives the first time of arrival of a wave compared to the variation of pressure or velocities. The Helmholtz equation is obtained when a Fourier transform is applied to the wave equation. The frequency becomes a variable of the system instead of being a parameter. The Eikonal and Helmholtz equations are not in the scope of this work.

Only the acoustic wave equation will be modelled in the experiments and it is given by the following formula :

$$\rho\nabla\cdot\left(\frac{1}{\rho}\nabla p\right)-\frac{1}{v^2}\frac{\partial^2 p}{\partial t^2}=-\rho\frac{\partial^2 f}{\partial t^2} \tag{4.2}$$

In this setting $p(t,\mathbf{x})$ describes the pressure response in the acoustic medium known as the wavefield. The $f(t,\mathbf{x})$ term is the point source term for the source injection. v is the wave velocity in the medium and ρ its density. When the density of the medium is homogeneous and the source term negligible, Equation 4.2 can be simplified to obtain the canonical form of the wave equation :

$$\nabla^2 p-\frac{1}{v^2}\frac{\partial^2 p}{\partial t^2}=0 \tag{4.3}$$

According to Moseley et al., 2020, the wave equation is difficult to solve despite its linearity. Indeed one can observe complex phenomena when waves propagate in complex media. Refraction, reflection and transmission can occur at boundaries between layers. A large

number of waves can interact in different velocity regions with different range of amplitude and frequencies.

4.4 Salvus : Finite Element Solver in Python

The finite element package in python Salvus (Afanasiev et al., 2019) is used to generate training data for the neural networks. The acoustic wave equation is solved for different problems with different geometries to simulate the wave propagation. The wavefields can be saved as images and then used to train the neural networks and assess their predicting performances.

4.4.1 Acoustic wave propagation in Salvus

Salvus uses an abstract formulation of the acoustic wave equation. This allows to treat different type of parameters to characterise the medium. The density $\rho[kg/m^3]$ and the sound speed velocity v_p or $c[m/s]$ are used in the simulations. In case of a 2D simulation in a time interval I the general form is given by :

$$m_0(\mathbf{x})\partial_t^2\phi(\mathbf{x},t) - \nabla\cdot(m_1(\mathbf{x})\nabla\phi(\mathbf{x},t)) = f(\mathbf{x},t) \tag{4.4}$$

The parameters m_0 and m_1 need to be converted with $m_0 = \rho^{-1}c^{-2}$ and $m_1 = \rho^{-1}$ to obtain the following equation :

$$\rho^{-1}(\mathbf{x})c^{-2}(\mathbf{x})\partial_t^2\phi(\mathbf{x},t) - \nabla\cdot(\rho^{-1}(\mathbf{x})\nabla\phi(\mathbf{x},t)) = f(\mathbf{x},t) \tag{4.5}$$

When the inputs units are a Force Density $[N/m^3]$ then the output obtained $\phi\dot{\rho}^{-1}$ corresponds to the pressure P in $[N/m^2]$. If the source units were of Volume Density Injection Rage $[1/s]$ then the ϕ_t would have to saved in order to obtain the pressure P. For the Salvus simulations, a Force Density is applied for each source to compute the pressure ϕ in $[Pa]$ or in $[N/m^2]$.

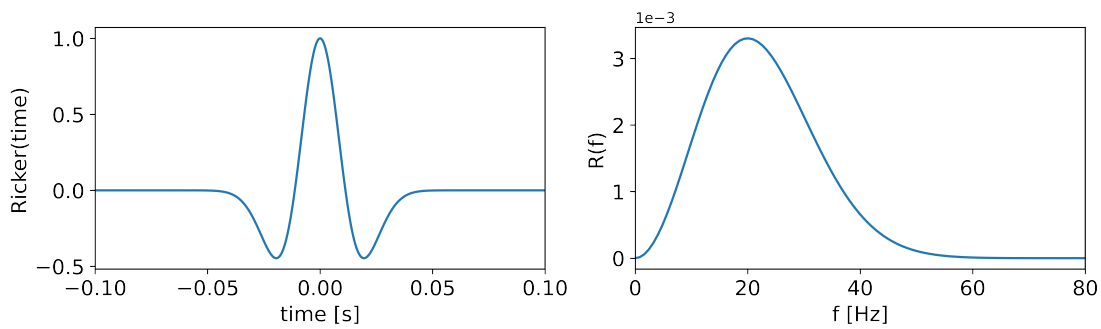


Figure 4.2 – The amplitude of the Ricker wavelet which will by the input for the sources located in the medium. The most energetic frequency in at 20 [Hz]

The input function generating the waves is the Ricker wavelet function (Ricker, 1953). It is the

second derivative of the Gaussian defined in Y. Wang, 2014 by the following equations :

$$r(t) = (1 - 2\pi^2\omega_p^2 t^2)\exp(-\pi^2\omega_p^2 t^2) \tag{4.6}$$

$$R(f) = \frac{2\omega}{\sqrt{\pi}\omega_p^3}\exp(\frac{\omega^2}{\omega_p^2}) \tag{4.7}$$

In Salvus, the initial conditions on the entire domain are expected to be zero. Since the Ricker function presented in Fig. 4.2 is centred around 0, artefacts will be generated by the simulation and the data obtained will be wrong if the simulations start at 0.0 [s]. To resolve this issue, an optional time-shift can be applied to the Ricker function. The solution chosen for my case was to start the simulations before 0.0 [s] to have the null part of the Ricker equation.

4.4.2 Presentation of the cases

All the different simulations performed in Salvus are listed in this section. One can find the different layers and domains composing each experiments. To run a simulation in Salvus parameters have to be specified. The geometry and the mesh characteristics allow to create the domain. Then physical parameters have to be assigned such as the medium density and velocity. Finally the receivers and sources locations as well as the experiment start, end and interval (time-step) duration need to be specified. Some basics parameters are common for all the experiments and they are presented in Table. 4.1. The Geometry, Mesh and Source parameters are the same used in the paper written by Moseley et al., 2020.

Geometry $[m]$		$x_{min} = 0$, $x_{max} = 1500$, $y_{min} = 0$, $y_{max} = 1500$							
Mesh		$nelem_x = 300$, $nelem_y = 300$							
Source		Ricker activation function centred at 20 $[Hz]$							
Receivers	x	0	0	500	1000	1500	1500	1000	500
locations $[m]$	y	500	1000	1500	1500	1000	500	0	0

Table 4.1 – Common Mesh Characteristics for all experiments conducted in Salvus

The varying parameters will be the sources locations. For some experiments, only one source will be located in the middle of the domain, for others one source will be placed at (x,y) = (1000,1000) [m] or on the edges of the domain to simulate sensors located on the sides of a laboratory sample. The latest locations are more realistic sensors locations in real case applications. The duration of the experiments will also change varying from -0.1 [s] to 0.4 [s] to -0.1 [s] to 0.8 [s] and also the media velocities for an homogeneous domain or heterogeneous domains.

The simulations conducted will range from a homogeneous domain with only one source

located in the centre to a medium with a propagating crack with all the sources and receivers placed on the edges.

4.4.2.1 Homogeneous Case

The first medium has a constant velocity and density and its sole source is placed in the centre of it (Fig. 4.3) . All the receivers outputs will be the same because the receivers are located to the same distance from the source. The difference for second experiment is the source location changed from (x,y) = (750,750) [m] to (x,y) = (1000,1000) [m]. The last experiment will have its sources located along the edges (See Fig. 4.4). The detailed parameters used in Salvus to run the simulations are presented in the appendix in Table B.1 for the homogeneous cases.

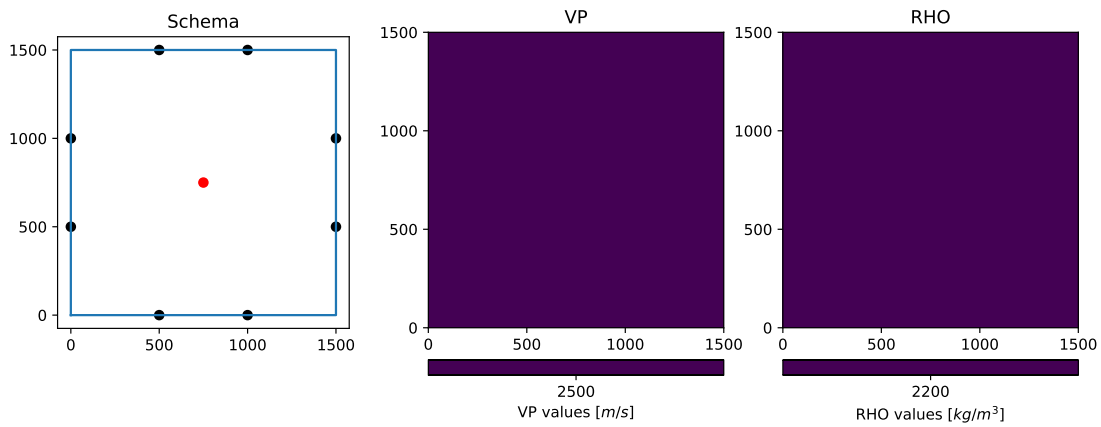


Figure 4.3 – Homogeneous domain with a source located in the middle of the domain and receivers on the sides.

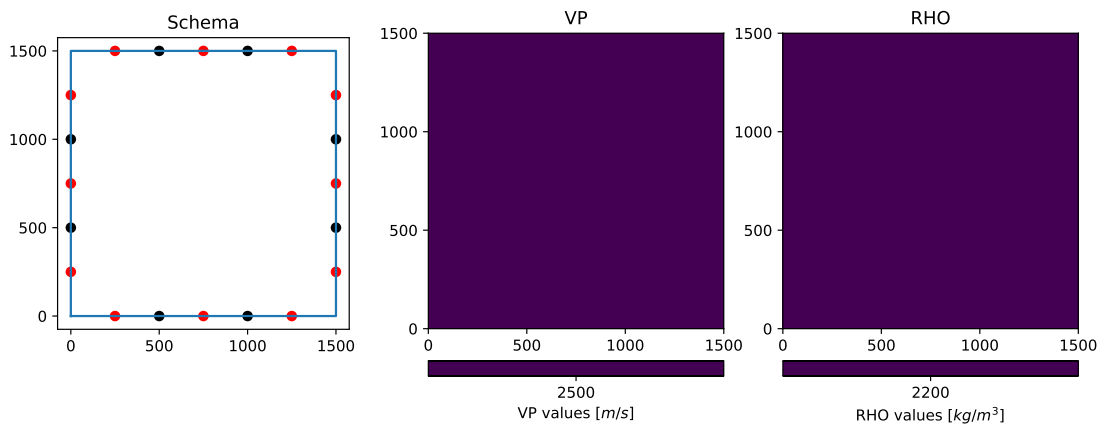


Figure 4.4 – Homogeneous domain with sources and receivers located along the edges of the domain.

4.4.2.2 Layered Heterogeneous Case

The second type of experiments is conducted on a layered medium meaning to represent an earth realistic model. Because of the more complex geometry, reflected and diffracted wavefield will occur during the simulation at the boundaries between the layers because of the different velocities. The sources will be located at the same places than the homogeneous cases : in the middle, then (x,y) = (1000,1000) and finally along the edges. The v_p values describing the compressional wave velocity is assigned randomly between 2000 $[m/s]$ and 2600 $[m/s]$ with an interval of 100 $[m/s]$ for each layer. The depth and the number of each layers are also random but a maximum of 6 layers was chosen. The density ρ is computed with the Gardner's empiric relationship Gardner$_1 974 which derive the bulk density from the P-wave velocity. \rho = \alpha v_p^{\beta} = 0.31 v_p^{0.25}$

(4.8)

Where α and β are empiric constants that depend on the geology and set from Gardner et al., 1974 to 0.31 [-] and 0.25 [-]respectively.

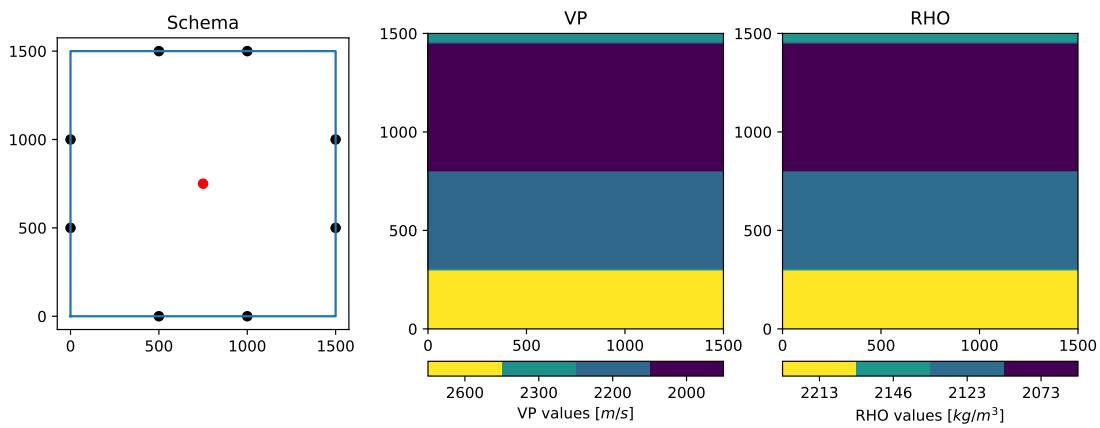


Figure 4.5 – Layered heterogeneous domain with a source located in the middle of the domain and receivers on the sides.

4.4.2.3 Crack Heterogeneous Case

Two experiments with an elliptical crack are simulated to reproduce a crack propagating when fluid is injected into the rock. Neural networks are in order to act as a surrogate model to

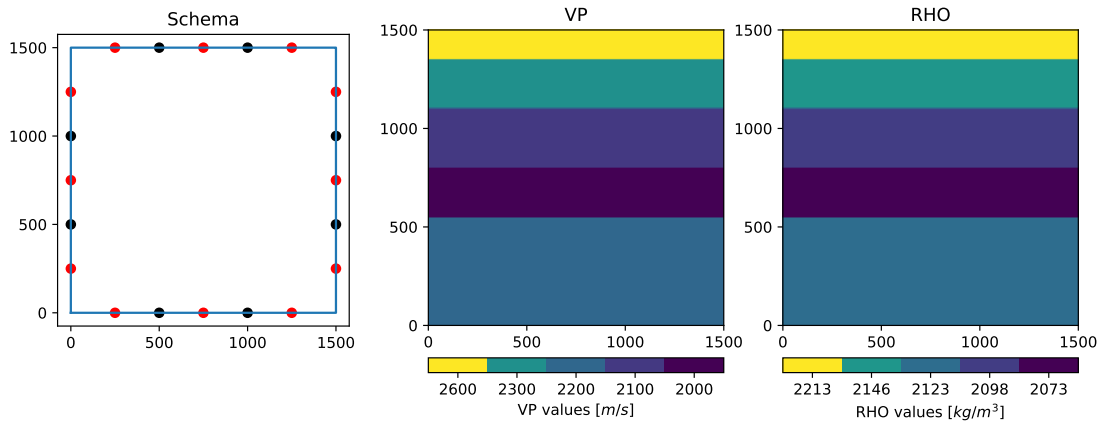


Figure 4.6 – Layered heterogeneous domain with sources and receivers located along the edges of the domain.

reproduce the wave propagation for this kind of experiments as the ones conducted in the laboratory by Liu et al., 2020. In contrary of the experiments conducted by the authors, no borehole is present and the simulation is in 2D instead of 3D. The dimension of the small crack is (a,b) = (250,40) [m] for the big and the small axis and for the big crack (a,b) = (500,40) [m]. Only elliptical cracks are simulated here but the code can create more complex fractures shapes such as a combination of rectangular, circular and elliptical. All the parameters are presented in Table B.3.

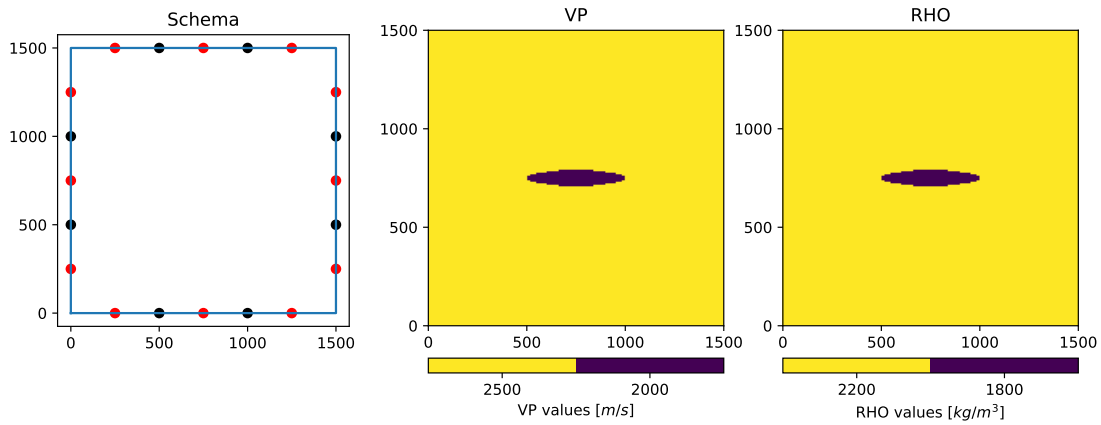


Figure 4.7 – Small cracked heterogeneous domain with sources and receivers located along the edges of the domain.

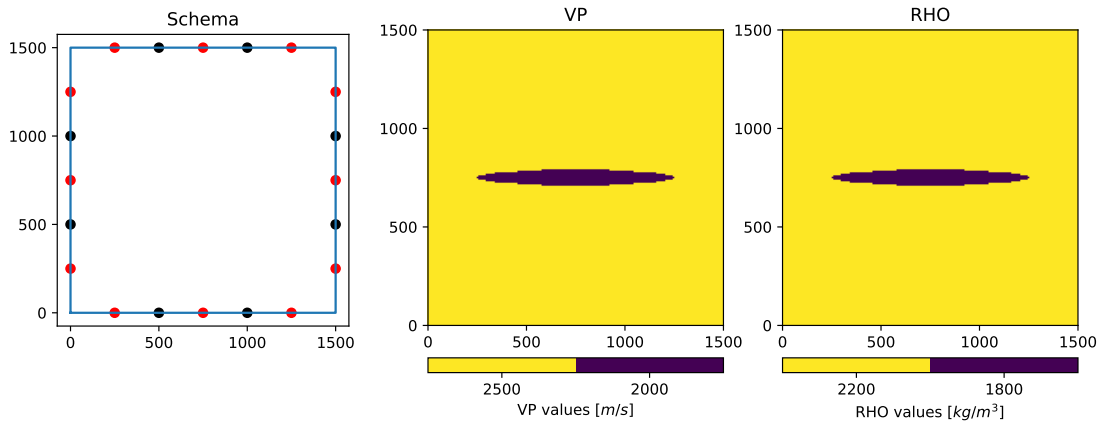


Figure 4.8 – Big cracked heterogeneous domain with sources and receivers located along the edges of the domain.

4.4.3 Simulations outputs from Salvus

The simulations with Salvus will be run with a timestep interval of 0.0002 [s]. This means that for a total duration of 0.5 [s] 2501 timesteps will be required to satisfy the CFL condition since Salvus uses explicit analytical methods. To save some space, the wavefields are saved for every 10 timesteps. For example if a simulation of 0.5 [s] is run with only one source (one event), around 2.8 [Gb] of data are generated when the wavefields are saved every 10 timesteps. When all outputs are saved more than 27.0 [Gb] of data are generated. Running all simulations and saving all timesteps will represent more than 1.0 [Tb] of data and could not fit on my device.

Figure 4.9 shows the amplitude for the receivers on the left and the wavefields on the right for t = 0.4[s]. It is normal that at first values are zero because the wavefield has not reach the receivers. The vertical lines on the receivers plot (left) show what is the value read by the receivers at the time the snapshot is taken. One observes that the response of the amplitude for all receivers are the same because the receivers are all located at the same distance from the source.

The Figure 4.10 presents the output at the same time but for the layered heterogeneous domain. Because of lower average velocity, the wave-front has not reach the receivers. The wave-front is not reaching all the receivers at the same time because of the different layers. There is a diffracted created by the change in velocity at y = 300 [m] which decreases from 2200 [m/s] to 2000 [m/s].

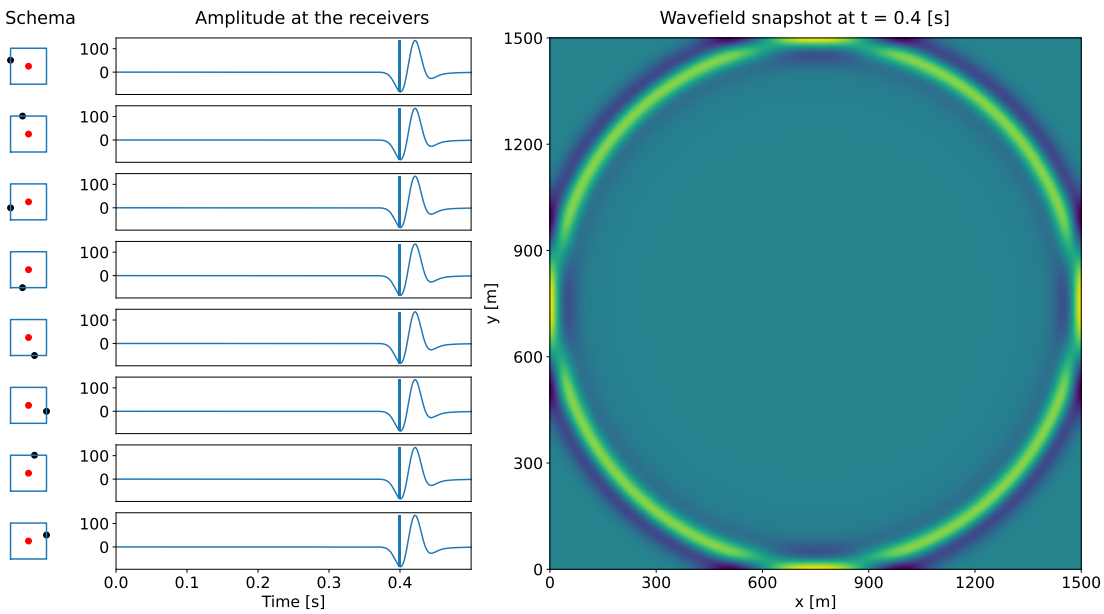


Figure 4.9 – Outputs for the homogeneous layered with one source in the middle at t=0.4[s]. In the left are shown the receivers amplitude and the corresponding wavefield on the right.

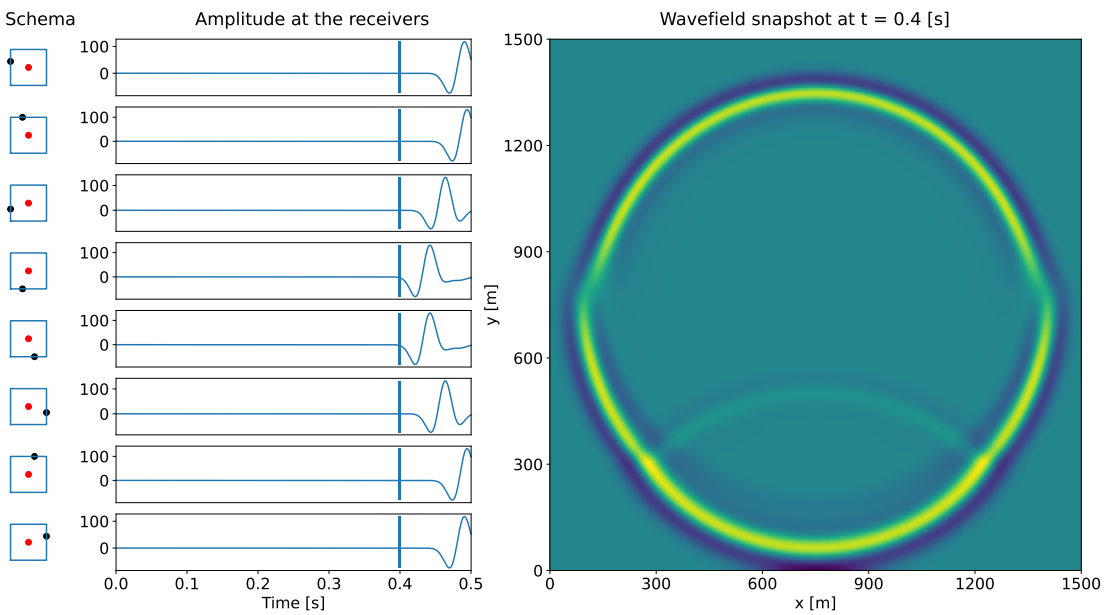


Figure 4.10 – Outputs for the heterogeneous domain with one source in the middle at t=0.4[s]. In the left are shown the receivers amplitude and the corresponding wavefield on the right.

5 Methodology

The methodology will describe the training data, the loss functions and the metrics used in order to reproduce the neural networks. Three separated sections are presented because of the three different tasks carried out in this master thesis.

5.1 Training Curve fitting

The artificial data are trained with a MLP which takes as inputs the x and y locations and the output will be the target values either 1 or 0 for the Squared and Circular domain (See Fig. 4.1) or the $sin(4x)sin(4y)$. The squared domain contains exactly 52 % of 1 of 48 % of 0. This percentage is really imbalanced for the Circular domain since there is only 3.5 % of 1. The model might interpret the 1 as outliers since it will be much more logical to predict only 0 over the entire domain.

The training and the test data are the same set and they are taken from the whole domain. Since the mesh is 50x50 there is 2'500 training data points or values for the models to fit. The inputs will be the x and y locations and the output will be the target values. The architecture of the model is quite simple and is presented below along its parameters :

- **Architecture** (Number of neurons per layer) **:** 2 - 100 - 100 - 1
- **Activation function for the hidden layers :** ReLu
- **Optimiser :** Adam, with a learning rate $\mu = 0.001$ (See Eq. 5.8)
- **Epochs :** 100'000

For each of the three simulations, the models will be trained from scratch with the MSE loss. The gradient descent algorithm is used and the model parameters/weights are updates after

every epoch. The corresponding loss function for one epoch is :

$$L = \frac{1}{N_u} \sum_{i=1}^{N_u} (u_i - \hat{u}_i)^2 \tag{5.1}$$

5.2 Training the 1D Burgers' Equation

For the 1D Burgers' equation, a comparison will be made against the data driven and the physics informed neural networks models. The two models will have the same architecture, the only difference will be the inputs adn the loss. The data driven neural network will be trained in a supervised approach and will receive data sampled from the domain.

The PINN will be trained in a semi supervised manner with the PDE directly given in the loss function and to implement the physics. The partial derivatives are computed with the automatic differentiation as they are explicitly given in the inputs. The inputs for the Burgers' problem are (x, t) and the output is the velocity (u). Then the residuals are computed and the mean square error is minimised on the sampled collocation points.

Models are trained using a certain proportion of available output values, N_u in Table. 5.1 and the number of collocation points for the PINN is fixed.

Table 5.1 – Presentation of the number of known values and collocations points for each simulation. Each row corresponds to one training scenario.

(a) Data Driven Training

N_u	N_0	N_f
0	100	0
50	100	0
100	100	0
200	100	0
500	100	0
1'000	100	0
2'000	100	0
5'000	100	0
10'000	100	0

(b) PINN Training

N_u	N_0	N_f
0	100	10'000
50	100	10'000
100	100	10'000
200	100	10'000
500	100	10'000
1'000	100	10'000
2'000	100	10'000
5'000	100	10'000
10'000	100	10'000

For the data driven approach, data are sampled on the initial conditions and on the boundary u_0 but also in the domain u. The model is trained to predict the value given a certain input, the same way as in a regression. No physic is enforced and the loss function for one epoch is

the following :

$$L = \frac{1}{N_u}\sum_{i=1}^{N_u}(u_i - \hat{u}_i)^2 + \frac{1}{N_0}\sum_{i=1}^{N_0}(u_{0,i} - \hat{u}_{0,i})^2 \tag{5.2}$$

The loss function for the PINN is composed of two terms : the physics/residuals loss and the initial + boundary conditions (IC + BC) loss. For the IC + BC loss, random points are sampled on the boundary and on the initial conditions. The model should predict for those the correct value and no physic is enforced. IC + BC values must be provided to guarantee the uniqueness of solution of the PDE. The PINN's loss function for one epoch with $\hat{u}$ the predicted value and u the true value is the following :

$$L = \frac{1}{N_u}\sum_{i=1}^{N_u}(u_i - \hat{u}_i)^2 + \frac{1}{N_0}\sum_{i=1}^{N_0}(u_{0,i} - \hat{u}_{0,i})^2 + \frac{1}{N_v}\sum_{i=1}^{N_v}\left(\frac{\partial \hat{u}_i}{\partial t} + \hat{u}\frac{\partial \hat{u}_i}{\partial x} - \left(\frac{0.01}{\pi}\right)\left(\frac{\partial^2 \hat{u}_i}{\partial x^2}\right)\right) \tag{5.3}$$

The training algorithm for the data driven and the PINN training schemes vary a bit. For the Data Driven as shown in Algorithm. 1 the inputs are only evaluated on the domain. For the PINN training scheme (Algorithm. 2) first the inputs from the IC + BC are evaluated by the model and then the collocation points from which the PDE will be enforced. The same model

is used and the total loss is the summations of the two.

Algorithm 1: Training Algorithm for the Burgers Equation - Data Driven scheme

Input : Neural Network : $f(\cdot, \mathbf{w})$; Training data $u_{tot} = [u_{IC+BC}; u_{domain}]$; Number of epochs: N; Minibatch size: M; Learning rate : μ (Adam optimiser)

for n *to* N **do**
  $\mathscr{L}^i = 0$;
  for m *to* M **do**
    $\hat{u} \leftarrow f(\mathbf{u}_{tot}, \mathbf{w})$;
    $\mathscr{L}^i = \mathscr{L}^i + \mathrm{MSE}(\hat{u}^m, u^m)$;
  end
  $\nabla \mathbf{w} \leftarrow \mathrm{Backprop}(\mathscr{L}^i)$;
  $\mathbf{w} \leftarrow \mathbf{w} - \mu \nabla \mathbf{w}$;
end

Output : Trained Neural Network

Algorithm 2: Training Algorithm for the Burgers Equation - PINN scheme

Input : Neural Network : $f(\cdot, \mathbf{w})$; Training data u_{domain} and $u_{physics}$; Number of epochs: N; Minibatch size: M; Learning rate : μ (Adam optimiser)

for n *to* N **do**
  $\mathscr{L}^i = 0$;
  for m *to* M **do**
    $\hat{u} \leftarrow f(\mathbf{u}_{domain}, \mathbf{w})$;
    $\hat{u}_v \leftarrow f(\mathbf{u}_{physics}, \mathbf{w})$;
    $\mathscr{L}^i = \mathscr{L}^i + \mathrm{MSE}(\hat{u}^m, u^m) + \mathrm{MSE}\left(0, \hat{u}_{v,t} + \hat{u}_v \hat{u}_{v,x} - (0.01/\pi)\hat{u}_{v,xx}\right)$;
  end
  $\nabla \mathbf{w} \leftarrow \mathrm{Backprop}(\mathscr{L}^i)$;
  $\mathbf{w} \leftarrow \mathbf{w} - \mu \nabla \mathbf{w}$;
end

Output : Trained Neural Network

In Algorithm. 2 one notices that the MSE loss is used to minimise the residuals. Indeed, if we take the predicted residual values and then applying the mean squared error with respect to 0 as the target value, it is the same as enforcing the residual to be 0. No regulariser parameters are used between the data driven and the physics constrained loss in the PINN. The models parameters for the MLPs are the following :

- **Architecture** (Number of neurons per layer) **:** 2 - 20 - 20 - 20 - 20 - 20 - 20 - 1
- **Activation function for the hidden layers :** Tanh
- **Optimizer :** Adam, with a learning rate $\mu = 0.01$ (See Eq. 5.8)

- **Epochs :** 50'000

5.3 Training Acoustic Wave Equation

The choice of convolutional layers seems obvious when working with the wave equation because of the amount of data. Such architecture could greatly reduce the amount of parameters used as well as the training time. As previously explained in Sec. 3.2, the wavefield simulated with Salvus can be treated globally as images instead of taking each point on one wavefield individually. The previous work done in that field, notably Alguacil et al., n.d.; Fotiadis et al., 2020; Sorteberg et al., 2018; Weiqiang et al., 2017 used CNN to model the acoustic wave propagation.

The authors implemented different model architectures and losses which all yield different performances. From the literature, it was found that no research was done using CNN with physics constrained loss to simulate the acoustic wave propagation. The only paper using physics to constrain the training was Moseley et al., 2020 but the authors used a FCNN. Hence the objective is to train different CNNs and compare their performance while being trained solely in a supervised manner referred as "Data Driven". "PINN" will be used to refer to models when they are trained in a semi-supervised or self-supervised manner with the acoustic wave PDE. The goal is to see at what extend the physics constrained learning could be applied to the acoustic wave equation.

The deep convolutional models implemented is to be surrogate models in order to predict the acoustic wave propagation will be presented in that section, as well as data used to train the models, the different losses implemented in the training scenarios, the metrics used in order to assess their performances and the model selection.

5.3.1 Models

In order to create a surrogate model for the wave propagation different convolutional models and losses will be compared. The detailed model architectures can be found in the appendix (See sec. C). In this part, only the characteristic of the model will be presented. The MultiScale implemented by Mathieu et al., 2016 and used in Weiqiang et al., 2017 and in Alguacil et al., n.d. will be tested alongside the Encoder-Decoder and the U-Net model (Ronneberger et al., 2015). The Encoder-Decoder is tested since it is used in Mo et al., 2019 for the multi-phase flow predictions and we are interested to see if such an architecture can be applied to the

wave equation. In Fotiadis et al., 2020, the authors compared Recurrent Neural networks and a U-Net model, the latest yielding the best performances in a data driven approach. Hence it would be really interesting to see its performance for different losses and training schemes. The U-Net has a similar architecture to the model used Geneva and Zabaras, 2019 and possible to compare different approach developed by the authors.

5.3.1.1 MultiScale

The structure of the MultiScale network depicted in Figure. 5.1 aims to learn the next wavefield given the 4 previous wavefields. The deep neural network will do so by learning an end-to-end mapping of the wavefields. From the papers of Alguacil et al., n.d. and Weiqiang et al., 2017 the models are basically four multi-scale encoding, and decoding convolutional neural networks. The model used will be composed of three multi-scale networks with input size of $n \times n$, $n/2 \times n/2$ and $n/4 \times n/4$. If a fourth model would have been used then the last input would have been of size $n/8 \times n/8$. So for the inputs of size (300,300) the inputs will be (300,300), (150,150) and (75,75).

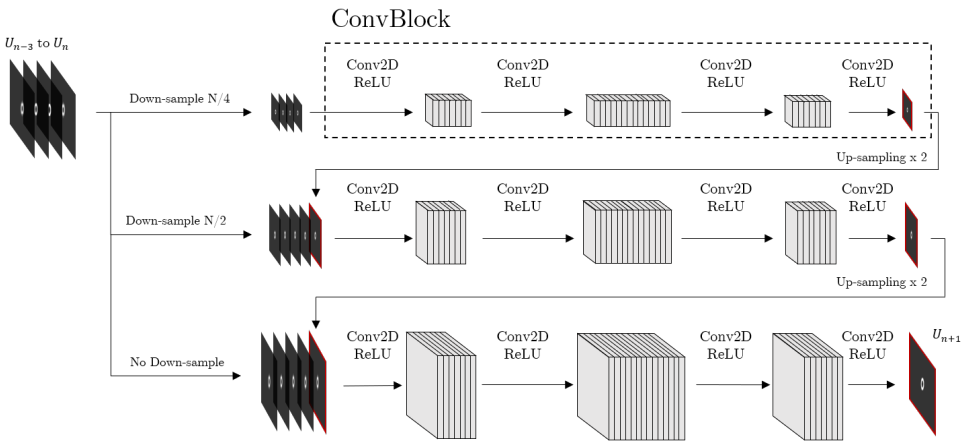


Figure 5.1 – Multi Scale Network Architecture

The three convolutional networks are connected successively from small scale to the large scale. The output from the small network will be up-scaled to match the dimension of the next model. The network at scale (150,150) takes both the inputs (300,300) down-scaled but also the output of the previous network (75,75) up-scaled to predict the next wavefield, which in turn will be up-scaled and used by the network (300,300). The main advantages of this multi scale structure is that the structure skip the pooling and unpooling pairs in the network and could preserve the high frequency information (Mathieu et al., 2016).

5.3.1.2 Encoder-Decoder

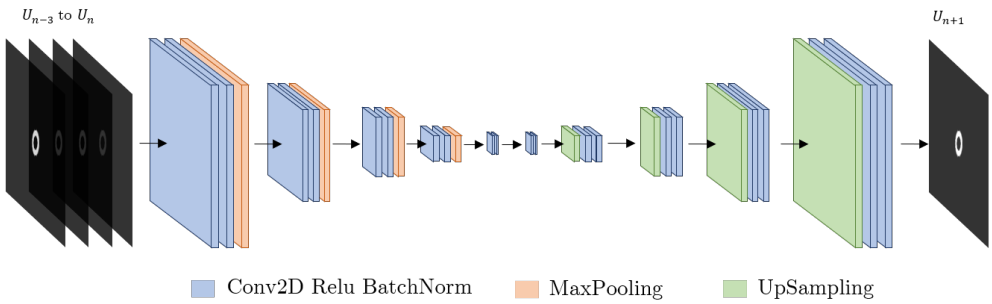


Figure 5.2 – Encoder Decoder Architecture

An Encoder-Decoder model is also modelled because it was previously used in other paper to model PINN (Mo et al., 2019). It performs a regression between high dimensional images. The images go through a coarse-refine process. The encoding layers increases the number of features maps until a latent representation of the inputs in that case the wavefields are learnt. Then the number of features maps is decreased and the output of each block are up-scaled to generate in the end the next wavefield.

The encoder operates on the input images to extract a set of high-level coarse feature maps, which are then fed into the decoder to eventually construct the output images.

The auto-encoder used by many paper for image to image regression task. It consists of many convolutional layers, the outputs is down-sampled with a max-pooling layer which basically reduce the size of the image by two. On the Figure 5.2 it correspond to the orange layers. The ConvBlock is a serie of two convolutional layers followed by the ReLu activation function and a BatchNorm layer.

5.3.1.3 U-Net

The encoder is composed of four blocks each containing two convolutional layers with kernel size of 3 and a padding of 1, followed by ReLu activation functions and Batch Norm layers. The first three blocks includes a max pooling layer of stride 2 meanings that the image will be reduced in half. The number of features maps doubles at each layers. Skip connections are also employed to copy the features maps from earlier. High level information coarser features maps are combined with fine grained local information of lower layers over the whole domain. The network architecture can be seen in Figure 5.3.

To remain consistent with the others model , it will take as input the 4 previous wavefields and

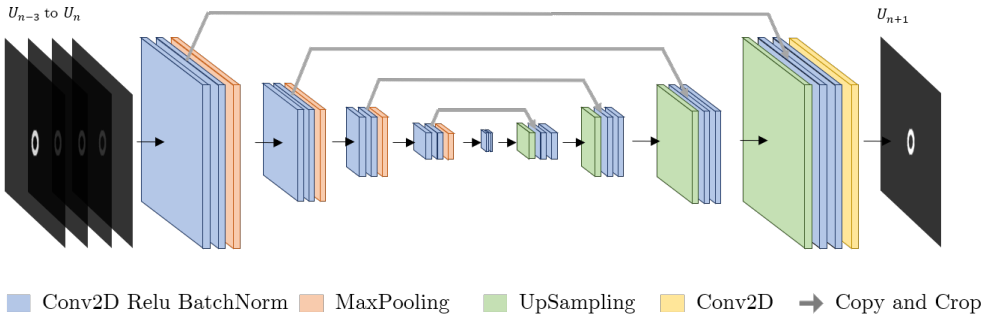


Figure 5.3 – U-Net Architecture

predict the next one.

The U-Net architecture has been used a lot for image segmentation tasks. Its architecture looks a lot like the encoder but its main difference is that after going through two Conv2D Relu BatchNorm blocks the outputs are by passed and concatenated in the up-sampling part of the network. The last layer is only a convolutional channel whose goal is to output only one channel, the next wavefield.

5.3.2 Model Selection

The goal is to compare the data driven and PINN methodology on a simple homogeneous domain with the source located in the middle for the different model architectures and loss functions. The model yielding the best results on this case will be kept and then more difficult cases will be implemented in the extrapolation section.

5.3.2.1 Loss functions

The different individual loss functions implemented to train the models are :

- The Mean Square Error (MSE) Loss or L_2 loss:

$$L_{MSE}(Y, \hat{Y}) = \frac{1}{N}||Y - \hat{Y}||_2^2 \tag{5.4}$$

- The Gradient Difference Loss (GDL) :

$$L_{GDL}(Y, \hat{Y}) = \frac{1}{N}||\nabla Y - \nabla \hat{Y}||_2^2 \tag{5.5}$$

- The Mean Absolute Error (MAE) Loss or L_1 loss :

$$L_{MAE}(Y, \hat{Y}) = \frac{1}{N}|Y - \hat{Y}| \tag{5.6}$$

Different loss functions are implemented in order to train the neural networks. They can be used individually or combined.

$$L_{tot} = \frac{1}{N}\sum_{k=1}^{N}(\lambda_{MSE}L^k_{MSE} + \lambda_{GDL}L^k_{GDL} + \lambda_{MAE}L^k_{MAE}) \tag{5.7}$$

with λ_{MSE}, λ_{MSE}, $\lambda_{MSE} \in \{0, 1\}$, N the total number of samples on which the summation is performed. Here, Y is the true wavefield and $\hat{Y}$ is the predicted wavefield. The most common loss functions is the Mean Squared Error. It tends to penalise largely big error and was the only loss function used in Fotiadis et al., 2020; Sorteberg et al., 2018 . In Alguacil et al., n.d.; Weiqiang et al., 2017, the authors introduced the Gradient Difference Loss (GDL) to enforce the sharpness of the image. Indeed since the gradient is computed, increase the sharpness of the predictions.Mathieu et al., 2016 obtained blurry results using only the MSE. The Mean Absolute Error (MAE) enforces the L1 distance between the prediction and the true value. Its value will be larger for smaller error than the L2 norm (MSE), and smaller in case of large error. By adding this loss term in the total loss, one could expect to match closer the small errors.

5.3.2.2 Scenarios Tested

In total, 6 training strategies will be tested combining different loss functions and supervised or unsupervised training. Because each model will be trained with each of the strategy, 18 scenarios will be implemented in total. The different strategies are presented in the list below. There will be three data driven/supervised and three physics-constrained/unsupervised strategies.

- MSE : a supervised training strategy with the MSE loss. The Eq.5.7 will have the parameters $\lambda_{MSE} = 1$ and $\lambda_{GDL} = \lambda_{MAE} = 0$.

- MSE + GDL : a supervised training strategy that combine the MSE loss with the GDL loss. This is the loss function implemented in Alguacil et al., n.d. and Weiqiang et al., 2017. The Eq. 5.7 will have the parameters $\lambda_{MSE} = \lambda_{GDL} = 1$ and $\lambda_{MAE} = 0$.

- MSE + GDL + MAE : the last supervised training strategy. All the parameters in Eq. 5.7 will be equal to one, $\lambda_{MSE} = \lambda_{GDL} = \lambda_{MAE} = 1$.

- Physics constrained residual (PINN RES) : This strategy will be trained in a semi-supervised manner. Instead of using the real output to train the network, the residuals are minimised. For the one timestep, $[u_{n-1}, u_n, \hat{u}_{n+1}]$ are used in Eq. 3.11 to compute and minimise the residuals with the MSE loss. u_{n-1}, u_n are the wavefields from the inputs and $\hat{u}_{n+1}$ the output from the neural network residual.

- Physics constrained MSE (PINN MSE) : This strategy is also semi-supervised. For one timestep, the MSE loss between the network output and the true output wavefield computed with Eq. 3.12 (Finite difference method). This is slightly different than the last method since u_{n-1}, u_n are used to compute u_{n+1}. $\hat{u}_{n+1}$ are the predictions from the neural network. If the medium is homogeneous then Eq. 3.14 is used.

- Physics constrained MSE with the Auto-Regressive training scheme (AR PINN MSE) : This is an unsupervised training strategy. It uses the same method as the Physics constrained MSE (PINN MSE) for one timestep but the predicted output will be used as the input value to predict the next timestep and so on. The Auto-Regressive (AR) training algorithm (See Algorithm 3) was developed by Geneva and Zabaras, 2019. Its advantages is that only the initial conditions are required and it can be trained for as long as we want. The training procedure is illustrated in Figure 5.4.

Algorithm 3: Auto-Regressive Training inspired from Geneva and Zabaras, 2019

Input : Neural Network : $f(\cdot, \mathbf{w})$; Number of epochs: N; Number of intervals: T; Back-prop interval: P; Learning rate : μ (Adam optimiser)

```
for n in N do
    u_inputs = {u_{t0-4}, u_{t0-3}, u_{t0-2}, u_{t0-1}};
    for t = 0 to T do
        L^i = 0;
        for p = 0 to P do
            û^i ← f(u_inputs, w);
            u^i = T_Δt(U^i, F_Δx);
            L^i = L^i + MSE(û^i, u^i);
            u_inputs = {u_{i-3}, u_{i-2}, u_{i-1}, u_i}
        end
        ∇w ← Backprop(L^i);
        w ← w − μ∇w;
    end
end
```

Output : Trained auto-regressive model $f(\cdot, \mathbf{w})$

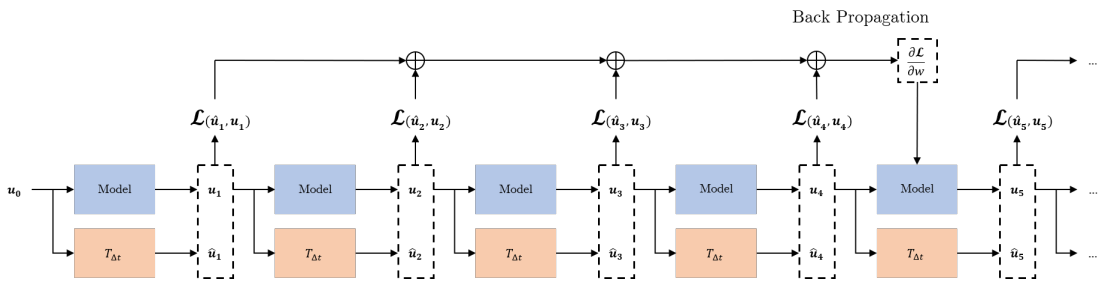


Figure 5.4 – Auto-Regressive (AR) training scheme. The back-propagation of the AR PINN MSE covers four predictions and all contributes to the gradient descent update. The loss accumulates over the four predictions. $\hat{u}_i$ are the model predictions and u_i are the target value computed using the numerical time integrator implemented with the finite difference method.

5.3.2.3 Datasets

Every models and every training strategies will be trained and evaluated on two different domains : homogeneous (See Fig. 4.3) and heterogeneous (See Fig. **??**). Their performances are computed on the base case but also on two different extrapolations.

Base case : The base case will contain the data from one experiment and the models will be trained on it. The source is located in (x,y) = (750,750) [m].

Space extrapolation : The first extrapolation is to take the trained networks on a different

source and see how they perform on a case with new initial conditions. One wants to see how they predict the new wavefields. The source location is different (x,y) = (1000,1000) [m] instead of (x,y) = (750,750) [m] for the training case. Hence new diffracted and reflected waves along the edge of the sample will appear.

Time extrapolation : The second extrapolation assess the models' and training strategies performances on the same experiment than the training case but on a longer time scale. The first half of the data is seen during the training and the second half will be new data. We want to see for how long the model is able to predict some relevant wavefields.

All the experiments used for the model selection are presented in Table. 5.2

Table 5.2 – Description of the dataset used in the model selection. For the Time extrapolation the data in parenthesis indicates that data are the same than the Base case.

Name		Type	Time	# of sources	# of Wavefields
Homogeneous	Base	Training	0.1-0.5	1	151
	Space	Test	0.1-0.5	1	151
	Time	Test	(0.1-0.4); 0.4-0.9	1	351
Heterogeneous (Layered)	Base	Training	0.1-0.5	1	151
	Space	Test	0.1-0.5	1	151
	Time	Test	(0.1-0.4); 0.4-0.9	1	351

Figure 5.5 shows an example of the data provided to the network during the training of a supervised scenario. The models will have at their disposal the four previous wavefields (Inputs 1 from 4) and the true value (next wavfield) that the model aim to predict (Output). The model will predict the next wavefield and the MSE loss for example is computed based on the prediction from the model and the true value. In case of the semi or unsupervised learning the models only receive the Inputs. Using four wavefields is an arbitrary number also used in Alguacil et al., n.d. and Weiqiang et al., 2017. The smaller number of timesteps was two in order to compute the u_{n+1} wavefields with the finite difference method.

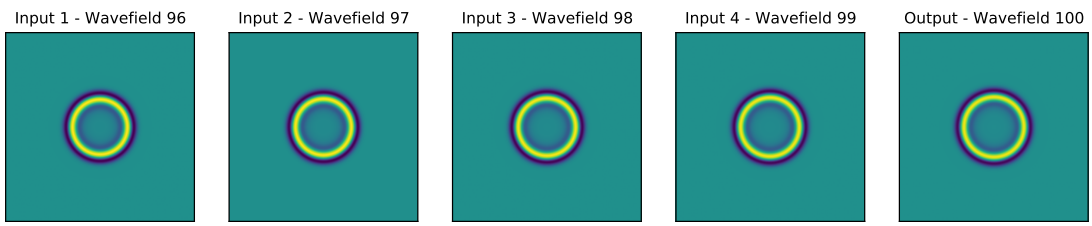


Figure 5.5 – First sample of the homogeneous dataset used for training. Inputs 1-4 are provided to the model to predict the next wavefield which is compared to the true expected wavefield (Output)

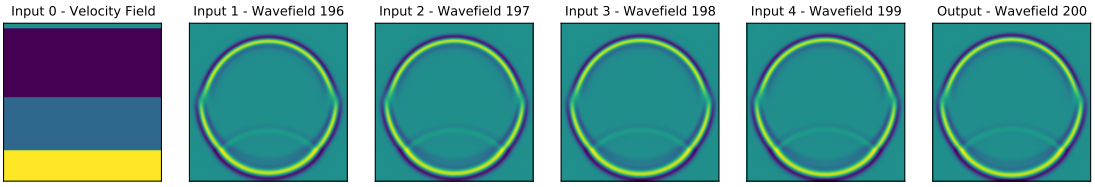


Figure 5.6 – Hundredth sample of the homogeneous dataset. Inputs 1-4 and the velocity profile are provided to the model to predict the next wavefield which is compared to the true expected wavefield (Output)

5.3.2.4 Fixed Parameters

- **Optimiser :** The ADAM (Adaptive moment estimation) optimiser (Kingma & Ba, 2014) is used in the stochastic gradient descent algorithm to update the parameters in the model for each mini-batch. The ADAM uses the moving average of each coordinates and its square to re-scale it. It can be seen as a momentum ($\hat{m}_t$) with a pre-coordinate re-scaling with ($\hat{v}_t$).

$$w_{t+1} = w_t - \frac{\mu}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{5.8}$$

  with $\hat{v}_t = (\beta_2 v_{t-1} + (1-\beta_2) g_t)/(1-\beta_2)$ and $\hat{m}_t = (\beta_1 m_{t-1} + (1-\beta_1) g_t)/(1-\beta_1)$ and g_t the sum of the gradients with respect to the parameters for the mini-batch. For the deep learning models $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-08$ and the fixed learning rate for all models $\mu = 1e-05$. An additional weight decay as also been used (Loshchilov & Hutter, 2017) which impose an L2 norm penalty.

- **Number of epochs :** 500

- **Mini-batch size :** 4

5.3.3 Surrogate modelling

From the model selection, only the best model will be kept and used as a surrogate model to predict the wavefields for sources and receivers located only on the sides in order to simulate laboratory experiments. One has to keep in mind that the training samples have to be relevant for the real cases. For example, if one wants to see how the wavefield propagates for sources on the edges, then the training dataset will contains mostly cases where the sources are located on the edges of the domain. If one is interested to model acoustic seismic prospecting, then

only the sources will be located only at the top of the domain. The same procedure goes with the duration of the simulations, the frequencies used by the sources and the domain characteristics. In this master project, only different source locations were extrapolated and in case of the crack a different domain was used too.

Homogeneous surrogate modelling : Eight sources located on the sides of the domain are used to train the model and the predictions are performed on the twelve available sources in order to see if the model can predict the wavefield for new locations on a homogeneous domain.

Layered heterogeneous surrogate modelling : Eight sources located on the sides of the heterogeneous layered domain are used to train the model and the predictions are performed on the twelve available sources in order to see if the model can predict the wavefield for new locations on a this heterogeneous domain. Since the sources are located on the edges at different location, they will all have different diffracted and reflected waves to predict.

Crack surrogate modelling : Two domains are considered for the crack surrogate modelling, one small crack (Crack 1) and one big crack (Crack 2) as presented in Figures. 4.7-4.8. The surrogate model will be trained only with the wavefields generated by 8 receivers on the small crack. Then the propagation for the twelve sources will be computed. Using the model trained on the Crack 1, the predictions for the Crack 2 will be computed too.

Table 5.3 – Description of the dataset used in the surrogate modelling. For Heterogeneous (Crack 2) the surrogate model has been trained with eight sources from Heterogeneous (Crack 1).

Name	Type	Time	# of sources	# of Wavefields
Homogeneous	Training	0.1-0.5	8	1208
	Test	0.1-0.5	4	604
Heterogeneous (Layered)	Training	0.1-0.5	8	1208
	Test	0.1-0.5	4	604
Heterogeneous (Crack 1)	Training	0.1-0.5	8	1208
	Test	0.1-0.5	4	604
Heterogeneous (Crack 2)	Test	0.1-0.5	8	1208
Training from Crack 1	Test	0.1-0.5	4	604

6 Results and discussion

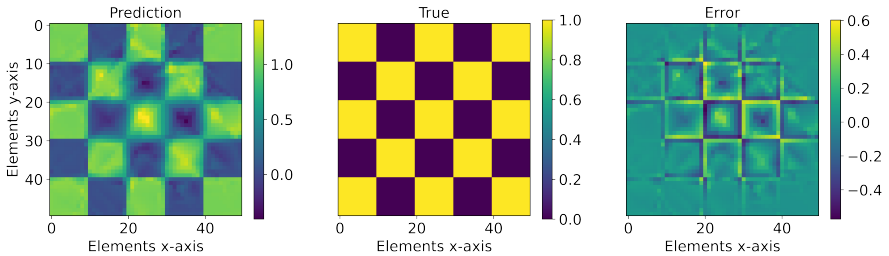
6.1 Curve Fitting

Neural networks are used to fit a function or complex shape in a domain. The predicted, trues and errors will be shown. The error here seems a relevant metric because it can highlight the relevant regions where the network fails to predict the correct values. If the relative error was chosen and since there are a lot of true values which are 0, it would totally diverge and be impossible to plot.

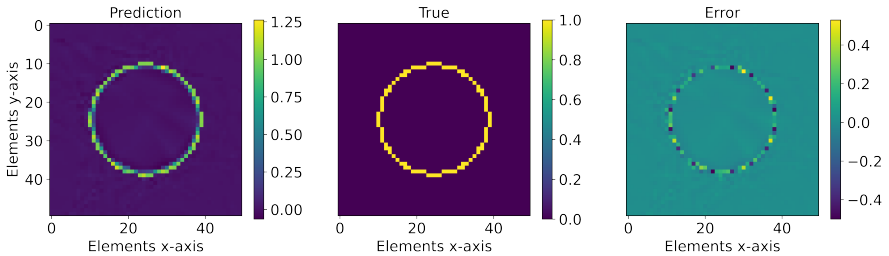
Given the network architecture (only two hidden layers of 100 units each trained for 100'000 epochs) and its capabilities to predict a 2D domain, we can observe a nice fit for all the functions. Even when the domain shows really sharp edges (Fig. 6.1a and Fig. 6.1b) it can predict well those discontinuities in general. For the circular domain, the MLP model can fit the data quite well even for the coarse 50x50 mesh. Indeed, the circle predicted has more the shape of a circle than the actual true circle.

There are few predicted values above 1 or below 0 for the Square and Circular domain. The predictions in the middle of the square domain are messy, the edges appears blurry on the error plot (Fig. 6.1a on the right) but along the edges on the domain the predictions are almost perfect. For the Sinusoidal function, except on the lower right edge of the predictions where the model fails to predict the 0 values, the distribution of errors are really small and centred around 0.

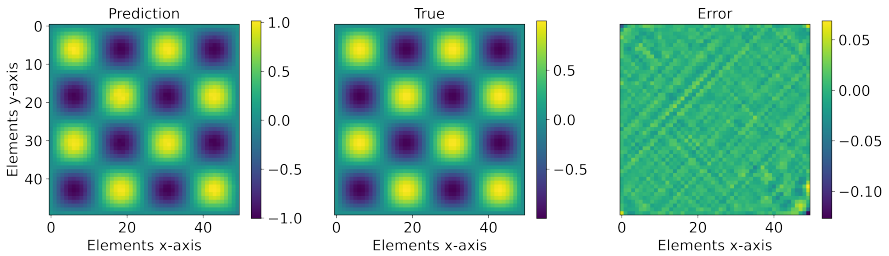
Computing the mean value of the error for the three experiments gives 0.035, 0.019 and -0.011 [-] for the Squared, circular and Sinusoidal domains respectively. The mean value is not a good metric for these kinds of problems since negative and positive errors compensate each



(a) Squared domain fitting



(b) Circular domain fitting



(c) Sinusoidal function

Figure 6.1 – Results for the artificial experiments composed of a Squared, Circular and Sinusoidal domains.

other. The median for the three experiments is 0.032, 0.020 and -0.012 [-]. Since the two are really close we can assume that the error for those experiments is normally distributed. This is confirmed by the histograms plot on Figure 6.2. The best metric to use would be the median of the absolute value for the error. It is more robust to outliers than the mean. For example if the model fails totally to predict the value at one location, the mean would change more than the median. The |median| for the three experiments is 0.048, 0.023, 0.012 [-]. When the absolute value is used on the errors, we observe that the new distributions follow a power law with most of the errors centred around 0 (Figure 6.2).

The amount of 0 and 1 for the squared domain is distributed with an equal number of zeros and ones. For the circle, only 3 % of points in the domain are present in the circle with the one value. This explains the shape of the distribution of errors. It is much wider for the square

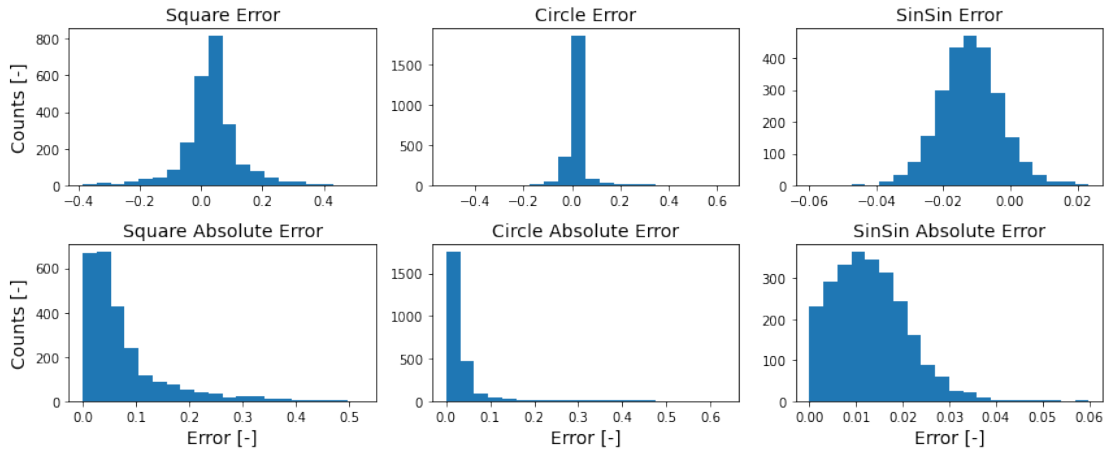


Figure 6.2 – Histograms of errors for the artificial simulations. The first row contains the histograms with the distribution of errors and the second row the distribution of the absolute value of the errors. 20 bins are used in the histograms

than for the circle since the model can learn very well to predict zeros.

Further works with the shapes tested could be to extend the Squared and Sinusoidal domains and see if the simple MLP network can also predict outside of its training domain. For example, it could be interesting to see if the model can continue predicting the patterns such as the squares or in the contrary if the predictions will smooth as presented in Haghighat and Juanes, 2021. The authors uses the function $f(x,y) = sin(x)sin(y)$ as it satisfies the following PDE ($f,_{xx} + f,_{yy} + 2f = 0$) and can be trained with a physics informed neural network.

6.2 Burgers 1D Equation

Previous section shows that a simple MLP can learn and predict quite a simple domain without the help of any physics components. This section presents the results when a Data Driven neural network and Physics Informed neural network are trained to solve the Burgers equation.

The 2'000 known value points are used to show how many points are required to obtain as accurate predictions as the physics enforced approach. Not so many points are sampled along the sharp discontinuities but still the network can learn it and predict it with a really small error.

Multiple networks are trained with different number of known values to assess the performances of the two approaches. For a small number of known values, the PINN yields better interpolation than the data driven approach because of the collocations points. When the

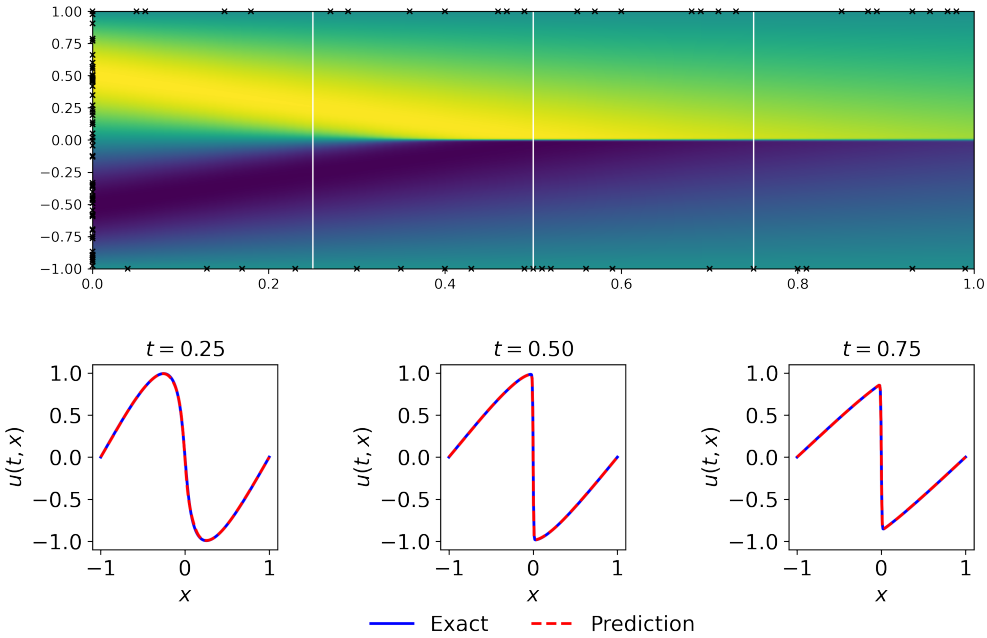


Figure 6.3 – Results for the Burgers 1D equation with the physic loss. The **x** marks represented all the known values used to enforce the boundary and the initial conditions

number of known values exceeds 2'000 then the data driven methodology yields better results. It means the norm is smaller. The two decrease linearly on the log scale but the data driven is better than the PINN for the same values of known values. This difference of the performance may be due to the complexity of the network. Since there is a relatively small amount of parameters, it might be difficult for the PINN to learn both the physics and also to predict the values where as the data driven only need to predict the values. The task that the PINN models need to learn might be too complex explaining why it yields the best results when only really few values are used. This is a good illustration for real life problem, obtaining a lot of data to solve a physical problems might be difficult and complicated but combining physics with few data sampled at certain locations might be a good trade off. Some researches has been made to select those locations to find the best locations at which one need to record the data and train the neural networks. Sun and Wang, 2020 worked on it by sampling the flow in arthritis simulation along certain lines.

6.3 Acoustic Wave Propagation

For a fair comparison, the performances of the models will be tested on the training data and also and the space and time extrapolation cases. Since a recursive prediction algorithm

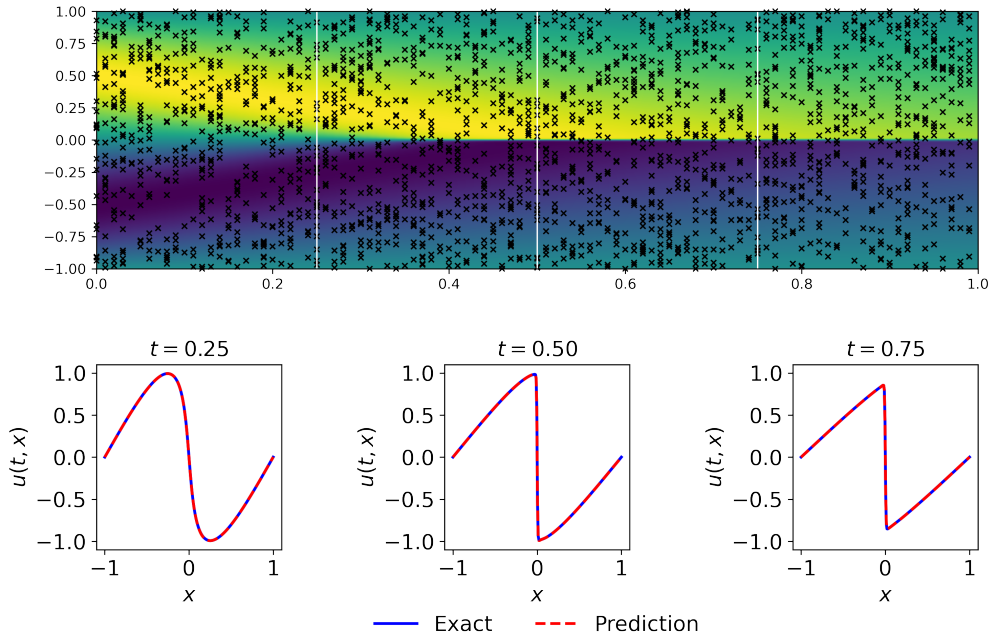


Figure 6.4 – Results for the Burgers 1D equation with the data driven driven approach. the x marks show all the known values randomly selected used to enforce the equation on the domain, the boundary and initial conditions.

is used the neural network starts with some known values only for the first prediction. The performances can be biased since it has already seen them during the training but as the predictions evolves, we hope that the model can keep on going predicting correct wavefield since the output of the network will be unseen data for it.

6.3.1 Predictions

From a set an initial wavefield the network predicts a set of discrete wavefield for a numerous timesteps. The set of initial wavefields acts as initial conditions and is provided to the model. In a recursive manner as presented in Geneva and Zabaras, 2019 the next wavefield is predicted. The predicted output is concatenated and will be the inputs in order to predict the next wavefield. After few predictions, the model has only inputs which are the predictions from previous timesteps and no true wavefield. The task might be complicated because if there is a small prediction error in one of the wavefield then the error propagates through time and the following predictions will be totally erroneous. The algorithm inspired by Geneva and Zabaras,

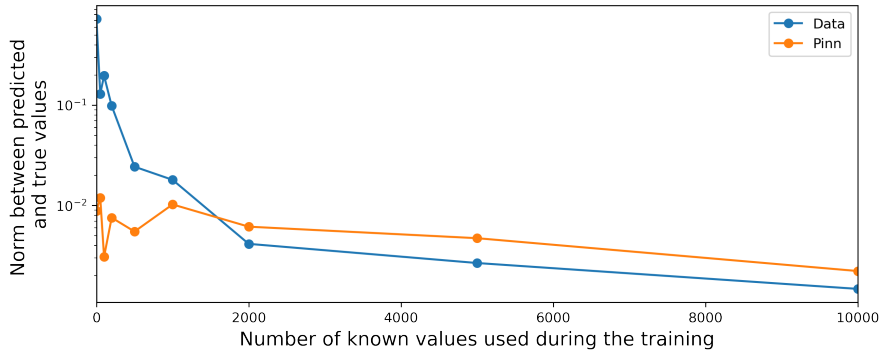


Figure 6.5 – The comparison between the Data Driven and the Physics informed approaches for the Burgers 1D equation.

2019 is the following :

Algorithm 4: Recursive predictions

Input : Trained Neural Network : $f(\cdot, \mathbf{w})$; Initial states: $\mathbf{u}_{inputs} = \{\mathbf{u}_{t_0-4}, \mathbf{u}_{t_0-3}, \mathbf{u}_{t_0-2}, \mathbf{u}_{t_0-1}\}$,
Number of timesteps to predict from t_0:T

$\mathbf{u}_{out} = [\quad]$; **for** t *in* T **do**

  $\mathbf{u}_t \leftarrow f(\mathbf{u}_{inputs}, w)$;

  $\mathbf{u}_{out}[t] \leftarrow \mathbf{u}_t$;

  $\mathbf{u}_{inputs} \leftarrow \{\mathbf{u}_{t-3}, \mathbf{u}_{t-2}, \mathbf{u}_{t-1}, \mathbf{u}_t\}$;

end

Output : Predicted time series $\mathbf{u}_{out} = [\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{T_{max}}]$

The only difference is that the model does not take a set of initial conditions u_0 but 4 continuous wavefields. To assess the performance of each model and its loss, different metrics are used to evaluate the generated wavefields. From Alguacil et al., n.d.; Weiqiang et al., 2017, the metrics used were the Peak Signal to Noise Ratio (PSNR), the Sharpness where the derivatives have to be computed, the Root Mean Squared Error or even the relative RMSE. Fotiadis et al., 2020 uses the much more convenient metrics of the Structural Similarity index (SSIM) bounded between 0 and 1 and reflects how close are two images by computing some intrinsic parameters such as the mean and the deviation. Most of the metrics will be discarded and only the RMSE, SSIM will be kept. New metrics such as the Mean Absolute Value of the error and the ratio between the Median Absolute Error and the maximum true value are used.

Root Mean Square Error (RMSE)

$$RMSE(Y, \hat{Y}) = \sqrt{\frac{1}{N}\sum^{N}(Y - \hat{Y})^2} \tag{6.1}$$

The relative root mean square error represents the evolution of the root mean square error relative to the maximum spatial value of the pressure target value.

Median Error over the max true value Provides indication over the distribution of error. If the value is really small then ther is a relatively small error or a large distribution of the error. It needs to be analysed with the Mean Absolute error.

Structural Similarity Index (SSIM) When comparing images, the mean squared error (MSE) is not highly indicative of perceived similarity. Structural similarity aims to address this shortcoming by taking texture into account Z. Wang et al., 2004.

Let x and y be the two images to compare, then the SSIM between the two pictures is (bounded by 0 and 1) :

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6.2}$$

With μ_x and μ_y are the average of x and y, σ_x^2 and σ_y^2 are the variances and σ_{xy} is the covariance of x and y. two variables $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are introduced to stabilise the division with weak denominator. $k_1 = 0.01$ and $k_2 = 0.03$ and L is the dynamic range of the pixel values.

6.3.2 Model Selection

In the model selection the procedure will be to select the model presenting the best predictive capabilities in average. Indeed, the tasks on which they will be tested are the homogeneous and heterogeneous layered domains for the training, the space and time experiments. One model might be better than the others when the source is placed at an other location but fails totally when one desires to predict for a longer period. The goal is to obtain the most robust model.

First the results are presented when the models are trained with the homogeneous cases then

Table 6.1 – Average RSME on the homogeneous domain for the model selection

(a) Training RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.79e+01	1.98e+01	1.85e+01
L2	1.05e+01	4.74e+00	7.88e+00
L2 GDL	7.72e+00	2.88e+00	4.34e+00
L2 GDL MAE	1.46e+01	**2.56e+00**	**2.80e+00**
PINN MSE	1.4e+01	7.18e+00	1.29e+01
PINN RES	**2.39e+00**	9.85e+00	3.28e+00

(b) Space extrapolation RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.74e+01	1.96e+01	2.07e+01
L2	1.12e+01	1.03e+01	1.51e+01
L2 GDL	1.22e+01	9.90e+00	4.70e+00
L2 GDL MAE	1.44e+01	1.11e+01	4.76e+00
PINN MSE	1.47e+01	1.1e+01	**3.86e+00**
PINN RES	**9.16e+00**	1.19e+01	6.53e+00

(c) Time extrapolation RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.85e+01	1.97e+01	1.79e+01
L2	1.74e+01	1.28e+01	5.38e+04
L2 GDL	1.65e+01	1.23e+01	**9.72e+00**
L2 GDL MAE	1.92e+01	1.2e+01	1.28e+01
PINN MSE	1.89e+01	1.45e+01	9.18e+04
PINN RES	**1.21e+01**	1.59e+01	1.22e+03

(d) Average RMSE for training, space and time

	U-Net	Encoder	MultiScale
AR PINN MSE	1.79e+01	1.97e+01	1.90e+01
L2	1.30e+01	9.28e+00	1.79e+04
L2 GDL	1.21e+01	8.36e+00	**6.25e+00**
L2 GDL MAE	1.62e+01	8.55e+00	**6.79e+00**
PINN MSE	1.59e+01	1.09e+01	3.06e+04
PINN RES	**7.88e+00**	1.26e+01	4.10e+02

the layered heterogeneous. The best performing model on those two cases will be kept for the extrapolations when the models are trained with sources located on the edges of the domain (ie. surrogate modelling) . The results presented are the recursively predicted wavefields at t = [0,0.05,0.1,0.2,0.3] [s] from the initial wavefields or timesteps 0,25,50,100,150 and the error at the receivers locations.

6.3.2.1 Homogeneous Domain

Since there are a huge amount a data generated, only the results deemed relevant are shown in this section for each experiments. More detailed materials can be found in the appendix (Sec. D.1) or in the GitHub repository online. For example, the wavefields as presented in Figure 6.6a can be generated for the 3 models, the 6 different losses and for each experiment (6 in total for the model selection). Hence only the results from best model in average are shown as well as the best model for the specific task.

The model is trained on homogeneous model, with the source located in the middle for only 1 event and then tested when the same event is extended into the time and when the source is placed on a other location. We expect than the model will predict accurately the training data since it has been trained on them. Then, when it is tested on the new experiments, if the predictions are convincing, we can state that the model will have learn the underlying physic equation that is the acoustic wave propagation.

Table 6.2 – Average SSIM on the homogeneous domain for the model selection

(a) Training SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	2.83e-01	2.70e-01	3.63e-01
L2	6.93e-01	9.08e-01	8.32e-01
L2 GDL	7.92e-01	9.56e-01	7.87e-01
L2 GDL MAE	5.42e-01	9.59e-01	9.36e-01
PINN MSE	5.37e-01	8.62e-01	8.04e-01
PINN RES	9.38e-01	8.41e-01	8.21e-01

(b) Space extrapolation SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	3.33e-01	3.38e-01	4.36e-01
L2	6.33e-01	8.25e-01	8.44e-01
L2 GDL	7.03e-01	8.23e-01	8.62e-01
L2 GDL MAE	5.79e-01	8.13e-01	9.18e-01
PINN MSE	5.65e-01	7.94e-01	9.12e-01
PINN RES	7.76e-01	7.79e-01	8.43e-01

(c) Time extrapolation SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	2.05e-01	1.81e-01	3.13e-01
L2	4.14e-01	6.32e-01	4.65e-01
L2 GDL	4.80e-01	6.46e-01	5.68e-01
L2 GDL MAE	3.66e-01	6.47e-01	6.25e-01
PINN MSE	3.48e-01	5.93e-01	4.18e-01
PINN RES	5.86e-01	5.83e-01	4.97e-01

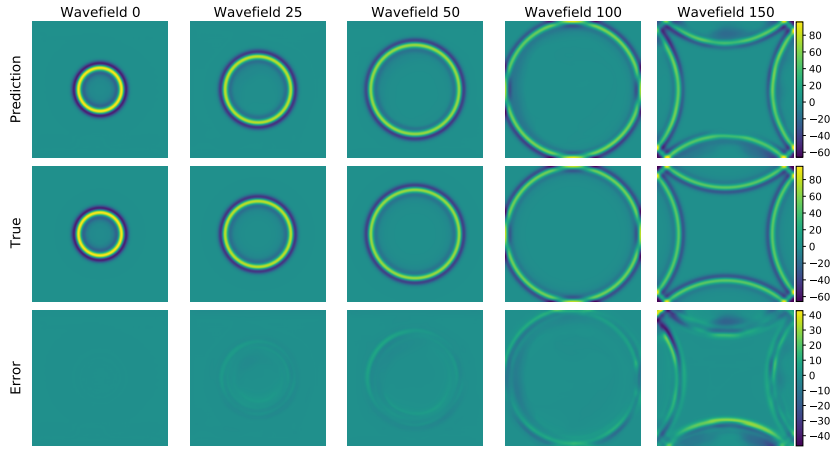
(d) Average SSIM for training, space and time

	U-Net	Encoder	MultiScale
AR PINN MSE	2.73e-01	2.60e-01	3.70e-01
L2	5.80e-01	7.88e-01	7.13e-01
L2 GDL	6.58e-01	8.08e-01	7.39e-01
L2 GDL MAE	4.95e-01	8.06e-01	8.26e-01
PINN MSE	4.83e-01	7.49e-01	7.11e-01
PINN RES	7.66e-01	7.34e-01	7.20e-01

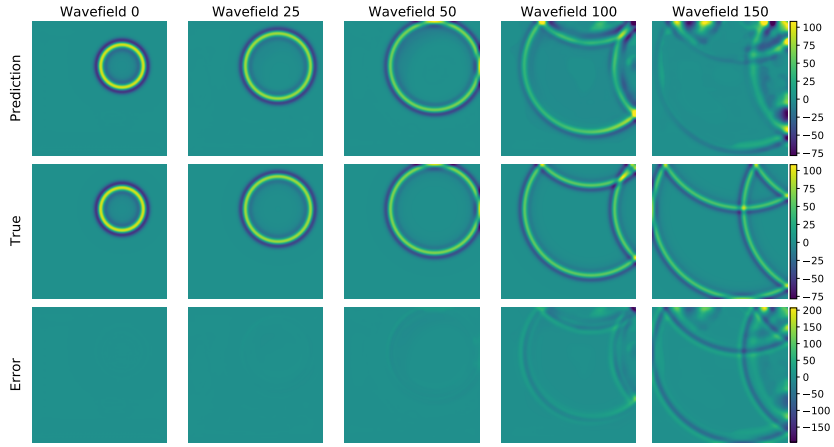
We observe on Figure 6.6a that the error between the predictions and the true wavefields remains really small until the 100th recursively predicted timestep for this model. All negative and positive pressure regions generated by the Ricker pulse are well matched. The wave is reflected along all the edges which is quite nice. However, one can see small errors being propagated after the wave is reflected against the edges of the domain and they increase. It is a crucial moment for the model because before the error is homogeneously distributed (see Wavefield 100) and after the reflection there are two regions with bigger error (Wavefield 150: upper left part and lower middle region). In the latest the wave-front is not as sharp as before.

Visually it is very difficult to quantify the error using the plots presented in Figure 6.6a. It can only provide insights at the global performances of the model and where the errors are located. It is more useful to investigate how the error propagates along a line. Since the source is located in the middle and the real acoustic wave propagation is the same along all the directions an evolution of the error is presented in Fig. 6.7 along the x-axis for the 5 selected wavefields.

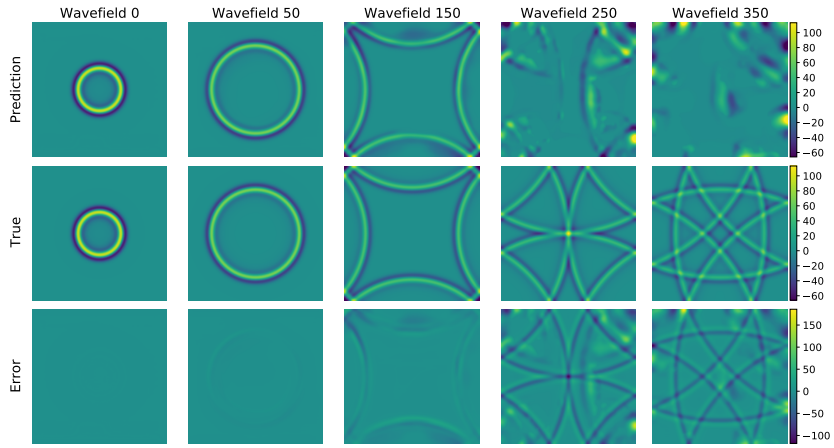
On the relative wavefield error plot, the error for each timesteps is divided by the maximum true values for this timestep. This allows to highlight the locations of the errors along the cut. One quickly notices that the errors are almost symmetric and that they travel along the wave-front. The errors for the first predicted timestep are so small that they coincide with the 0 line. Bigger relative errors appear when the wave-front hits the edges but the model still predicts no pressure when it is supposed to be 0.



(a) U-Net PINN RES on training data with the source located in the middle



(b) Space extrapolation for U-Net PINN RES with the source located at (x,y) = (1000,1000)



(c) Time extrapolation for -UNet PINN RES

Figure 6.6 – Wavefields predictions for the selected timesteps for the U-Net PINN RES scenario

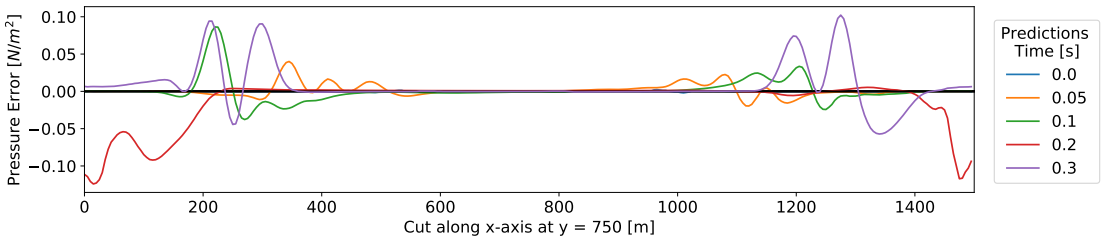


Figure 6.7 – Predicted wavefields error along the x-axis at y = 750 [m] for the U-Net PINN RES on training data for the five selected wavefields

An important output and task required from the networks is their acoustic predictions at some specific locations. As shown in Fig. 4.3, receivers were placed on the edges of the domain to record the pressure and we can extract the predicted pressure at those locations and compare it with the pressure computed by Salvus. In real experiments, receivers are the only means to obtain the acoustic signal and they can be used to compute the crack location, opening and angle when an injection is performed in a sample (Liu et al., 2020). If the errors between the predicted and the true values of the receivers are small and can be detected at the same time then the best model can definitely be used as a surrogate model.

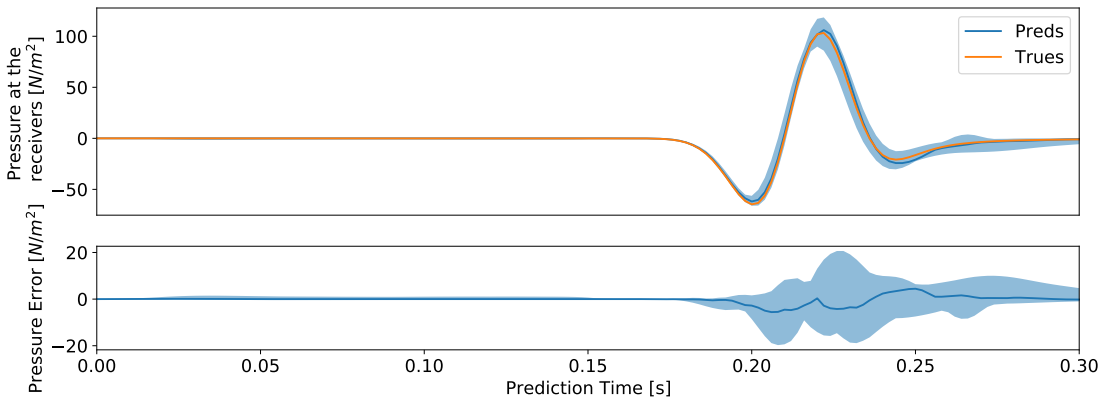
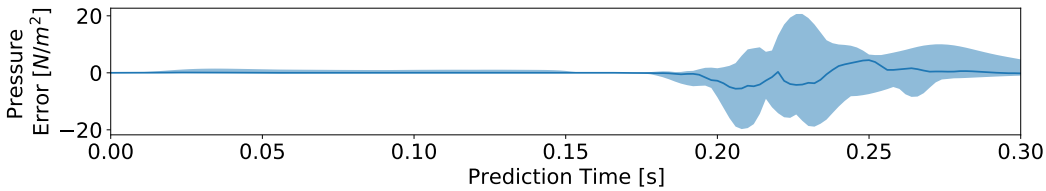
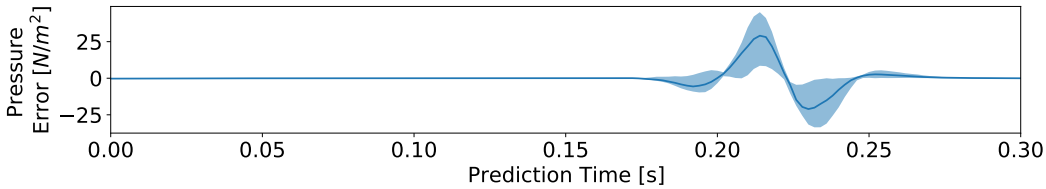


Figure 6.8 – Predicted pressure and Error at the receivers for U-Net PINN Res on the training data for the homogeneous model. The mean value of the receivers is shown in bold as well as the minimum and maximum interval with the shaded contour

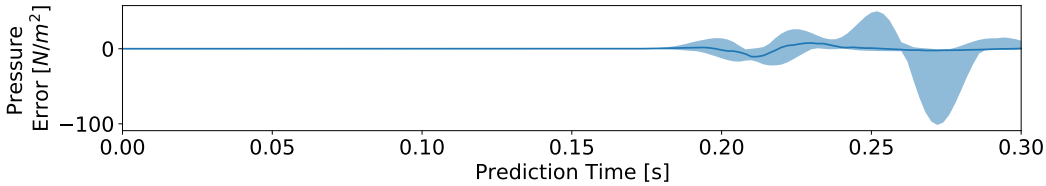
Fig. 6.8 shows the best pressure predictions and true values for the receivers located on the edge of the domain. The mean value and the minimum and maximum interval are shown because the receivers are all located at the same distance from the source. Their pressure should be the same for every timesteps (See Fig. 4.9). The two curves for the predicted and true values overlay each other and gives the error plot presented below it. The pressure arrives at the same time at the receivers.



(a) U-Net receivers errors



(b) Encoder Receivers errors



(c) MultiScale Receivers errors

Figure 6.9 – Comparison of the error at the receivers between the true and the predictions on the training case. Bold line is the mean and the shaded area in blue is the min max interval.

The errors presented in Figure 6.9 are for the best loss for each different model architecture on the training data. Each of them shows an average RMSE over the 150 predicted timesteps below 2.8 in Table 6.1. At first one can expect their predictions to be really similar but by looking at the receivers they are not. The U-Net model curve has not big peaks and the mean value remains really close to 0. The wave velocity is kept and only the amplitude when arriving at the receivers will be slightly bigger or smaller as shown on Figure 6.8. The errors shown in 6.9b shows on the contrary two peaks. This means that the wave received by the receivers is slightly delayed and for the best MultiScale model in Figure 6.9c the first part of the signal received is really similar to the one from the U-Net model but at around 0.27 [s] a large negative peak indicates that a new wave-front is recorded at one of the receivers created by the model as shown in Figure D.1a.

Once all the models are trained on the homogeneous domain, a new dataset is used for the space extrapolation. The source is switched from (x,y) = (750,750) [m] to (1000,1000) [m]. The wavefields are presented in Figure 6.6b for the U-Net model with the PINN RES training scenario and in Figure D.1b for the MultiScale model with the PINN MSE scenario. They

corresponds to the best model in average and the best model for this task based on the RMSE presented in Table 6.1 (b).

One can see the difference between the two models. The main difference is the amplitude of the wavefield and the range of the errors. First from 200 to -150 [Pa] where as the second only between 50 and - 80 [Pa]. U-NET PINN RES still predicts a wave-front but that is attenuated. The signal is present everywhere and the error starts to be visible when the wave-front is reflected against the edge after timestep 50.

Figure D.1c presents the wavefields with the L2 GDL loss and shows also good predictions until the wave-front reaches the edges of the domain. Afterwards, new waves are created which totally induces the larger RMSE than the U-Net and the MultiScale architectures. It creates a new wavefield whose propagation is really close to the behaviour from the homogeneous case. However the models show great capabilities of extrapolation despite having being trained on one simulation.

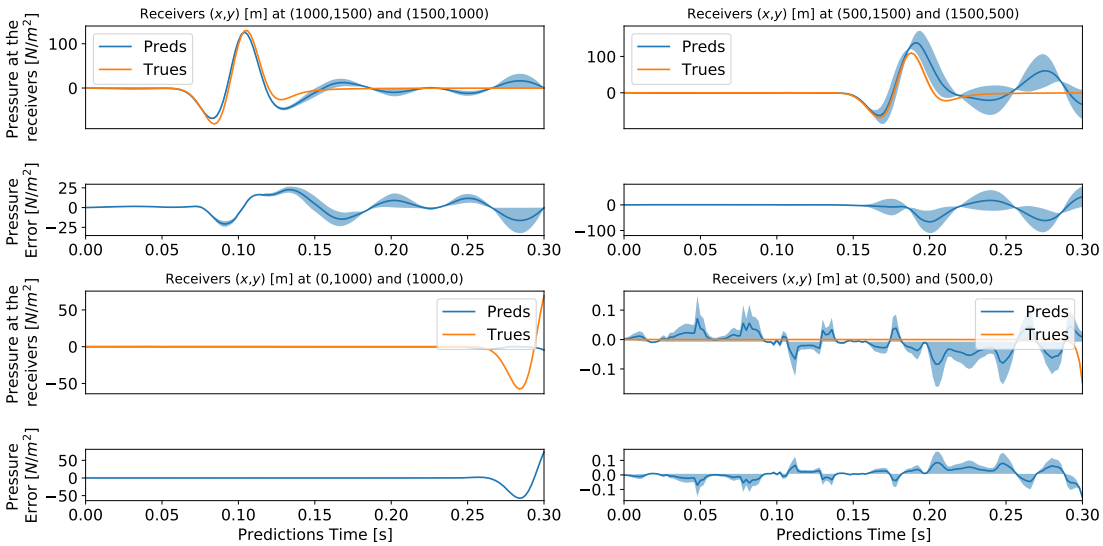


Figure 6.10 – Space U-Net Receivers - Each group of figure is plot for an equidistant pair of receivers when the source is located at (x,y) = (1000,1000) [m]

Looking at the errors for the receivers more difficult to interpret. Waves won't arrive at the same time anymore since the distance from the source to each receivers is different. The receivers located at (1000,1500 and 1500, 1000) [m] will receive the wave first, then (500,1500 and 1500,500) [m], (0,1000 and 1000,0) and finally (0,500 and 500,0) [m]. The predictions presented in Figure. 6.10 looks less precise than the case where the source is located in the middle. But keep in mind that it is unseen data for the model. The waves do not have time to

propagate to the furthest receivers hence the really small error for them.

The wavefront reaches the closest receivers at around the 50th timestep and the second around the 90th timestep. From the two, the wave amplitude has the time to decrease. Indeed in an homogeneous medium, according to Yilmaz, 2001 the energy density decays proportionately to $1/r^2$ where r is the radius of the wave-front. As the wave amplitude is proportional to the square root of energy density it decays proportionately to $1/r$ from the radius wave-front. We notice that the amplitude of the second receivers to predict is smaller than the first but the predicted value at the receivers has the same amplitude than the first which indicates that the model has trouble to implement this decay. However, the arrival time is perfectly matched.

The time extrapolation is the trickiest for the models. Indeed, one wants to see if the models are able to extend on larger timesteps. The models will have to predict the following 200 timesteps from the ones it was originally trained on. Since some errors already exist between the trues and predicted values, the model will have to predict 200 unseen timesteps with an existing bias. As expected, the worst RMSE and SSIM is observed between all the models for the results on the training data and space extrapolation. Encoder and U-Net models all have value between 10 and 20. For some MultiScale scenarios (L2 and PINN MSE) the predictions diverge.

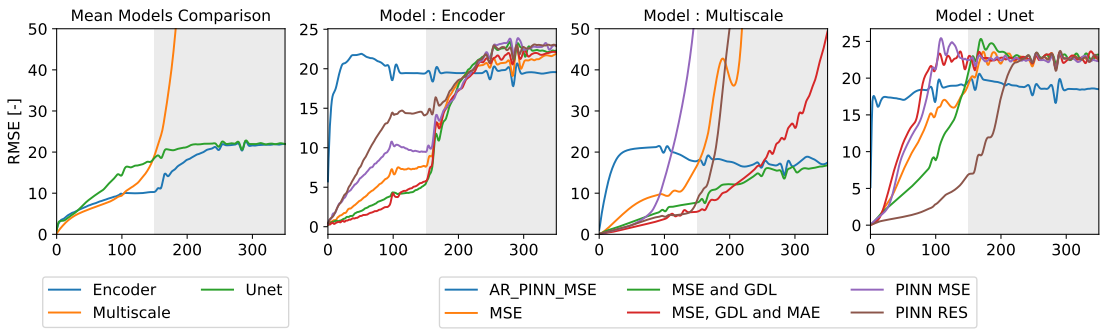


Figure 6.11 – RMSE Time with the mean RMSE for each model on the left and the detailed RMSE for all Loss tested. The shaded area represents all the timesteps predicted outside the training range (From 151 to 350)

A more detailed RMSE evolution presented in Figure 6.11. For the whole metrics for the time extrapolation, the interested reader can refer the the Figure D.4 presented in the appendix. We observe the average performance of the MultiScale model whose RMSE explodes. From Figure 6.11, each model architecture has a different behaviour. U-Net model either stabilises and keeps going up until a plateau is reached at around 23 [Pa] for all losses except the AR PINN RES scenario proposed by Geneva and Zabaras, 2019. An error is present already after

the first timestep and the amplitude of the wave-front decreases up to practically disappear for later timestep. However one notices that the PINN RES scenario takes longer time to reach the plateau hence being considered as the most stable model.

The encoder architecture shows clearly a shift between the predictions occurring on the training data and the time extrapolation. For all the losses, there is a clear increase of the RMSE once the first wavefield is predicted outside of the training data. The RMSE for all different losses tend also to converge to the same value around 300 to 350 timesteps.

We observe on Figure 6.6c the predictions for the U-Net model trained with the PINN RES scenario. Wavefield 150 is the same as in the Figure 6.6a. After the predictions are wrong and we can barely distinguish the shape of the wave-front.

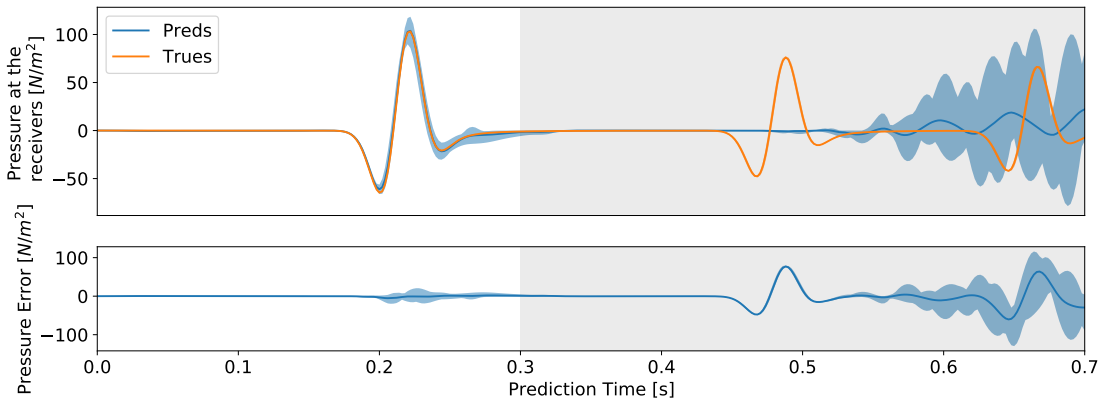


Figure 6.12 – Receivers errors for the time interpolation. The predictions on the available training data extends from 0.0 [s] to 0.3 [s] and the predictions on the extrapolated time data from 0.3 [s] to 0.7 [s] or the shaded area.

It is really interesting to see on Figure 6.12 that the true wavefields reach the receivers with a certain periodicity and the U-Net model is unable to predict the second arrival of the waves. It predicts the correct value of 0 between 0.3 [s] and 0.5 [s] (timesteps 150 and 250) and then starts to predict incorrect wavefields which corresponds to value reached by the RMSE on Figure 6.11.

One can observe the different models and losses predictions and we can state that the models in general has more generalisation capabilities for space extrapolation than the time extrapolation. With only one training scenario furthermore the most basic one, which is the source located in the middle, the model can extrapolate for an other source location and also predict the correct reflected wave-front against the edges.

6.3.2.2 Heterogeneous Domain

Perfectly homogeneous media are rarely present in nature or in lab experiments. We can consider them as so to calibrate the sources and receivers but all velocity are given with a deviation as the authors presented in Liu et al., 2020 due to imperfection in the materials.

Table 6.3 – Average RSME on the heterogeneous domain for the model selection

(a) Training RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.56e+01	1.51e+01	1.97e+01
L2	7.05e-01	6.68e+00	1.87e+03
L2 GDL	8.03e-01	7.59e+00	8.73e+00
L2 GDL MAE	2.61e+00	5.73e+00	inf
PINN MSE	3.17e+00	6.44e+00	1.85e+01
PINN RES	**1.01e+00**	9.96e+00	7.62e+00

(b) Space Extrapolation RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.33e+01	1.59e+01	1.86e+01
L2	1.30e+01	1.71e+01	1.20e+03
L2 GDL	1.43e+01	1.79e+01	1.17e+01
L2 GDL MAE	1.18e+01	1.71e+01	inf
PINN MSE	1.48e+01	1.75e+01	1.78e+01
PINN RES	**9.30e+00**	1.68e+01	1.00e+01

(c) Time Extrapolation RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.64e+01	1.68e+01	4.74e+01
L2	8.93e+00	1.3e+01	inf
L2 GDL	8.92e+00	1.45e+01	inf
L2 GDL MAE	9.16e+00	1.33e+01	inf
PINN MSE	1.02e+01	1.35e+01	inf
PINN RES	**8.28e+00**	1.44e+01	inf

(d) Average RMSE over training, space and time

	U-Net	Encoder	MultiScale
AR PINN MSE	1.51e+01	1.59e+01	2.86e+01
L2	7.54e+00	1.23e+01	inf
L2 GDL	8.01e+00	1.33e+01	inf
L2 GDL MAE	7.86e+00	1.20e+01	inf
PINN MSE	9.39e+00	1.25e+01	inf
PINN RES	**6.20e+00**	1.37e+01	inf

Table 6.4 – Average SSIM on the heterogeneous domain for the model selection

(a) Training SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	1.95e-01	5.96e-01	1.19e-01
L2	9.92e-01	8.68e-01	4.23e-01
L2 GDL	9.91e-01	8.48e-01	4.82e-01
L2 GDL MAE	9.43e-01	8.87e-01	2.81e-01
PINN MSE	9.31e-01	8.89e-01	4.24e-01
PINN RES	9.75e-01	8.07e-01	8.23e-01

(b) Space Extrapolation SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	4.17e-01	6.32e-01	1.55e-01
L2	7.10e-01	5.79e-01	5.18e-01
L2 GDL	7.14e-01	6.66e-01	5.16e-01
L2 GDL MAE	7.45e-01	5.34e-01	2.99e-01
PINN MSE	6.99e-01	5.29e-01	5.10e-01
PINN RES	7.99e-01	5.25e-01	8.03e-01

(c) Time Extrapolation SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	1.62e-01	4.37e-01	6.31e-02
L2	6.67e-01	5.72e-01	2.66e-01
L2 GDL	6.67e-01	5.29e-01	2.17e-01
L2 GDL MAE	6.58e-01	5.63e-01	2.07e-01
PINN MSE	6.34e-01	5.65e-01	1.69e-01
PINN RES	6.68e-01	5.52e-01	4.66e-01

(d) Average SSIM over training, space and time

	U-Net	Encoder	MultiScale
AR PINN MSE	2.58e-01	5.55e-01	1.12e-01
L2	7.89e-01	6.73e-01	4.02e-01
L2 GDL	7.90e-01	6.81e-01	4.05e-01
L2 GDL MAE	7.82e-01	6.61e-01	2.62e-01
PINN MSE	7.54e-01	6.61e-01	3.67e-01
PINN RES	8.14e-01	6.28e-01	6.97e-01

Moreover, when a fluid is injected in crack, there are at least two phases. For theses reasons, all the models and the losses are trained and their performances are assessed on a layered

domain supposed to represent an earth realistic model. Indeed, such models have many different layers with different densities and velocities and at each boundary, reflected and diffracted waves will be generated due to the change of those parameters.

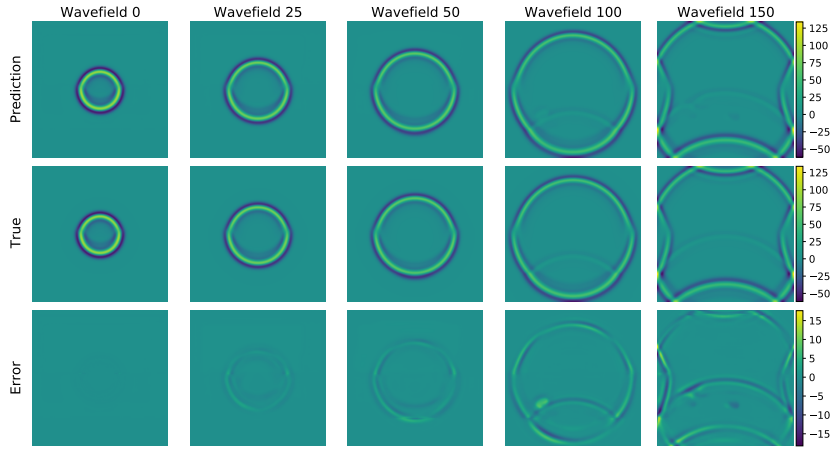
For the second model selection in order to obtain the surrogate model, the velocity field is always available and will be given to the model for all the recursive predictions. The Algorithm 4 is adapted and the first initial inputs change from $\mathbf{u}_{inputs} = \{\mathbf{u}_{t_0-4}, \mathbf{u}_{t_0-3}, \mathbf{u}_{t_0-2}, \mathbf{u}_{t_0-1}\}$ to $\mathbf{u}_{inputs} = \{\mathbf{u}_{field}, \mathbf{u}_{t_0-4}, \mathbf{u}_{t_0-3}, \mathbf{u}_{t_0-2}, \mathbf{u}_{t_0-1}\}$. The wavefields are also updated and changed from $\mathbf{u}_{inputs} = \{\mathbf{u}_{t-3}, \mathbf{u}_{t-2}, \mathbf{u}_{t-1}, \mathbf{u}_t\}$ to $\mathbf{u}_{inputs} = \{\mathbf{u}_{field}, \mathbf{u}_{t-3}, \mathbf{u}_{t-2}, \mathbf{u}_{t-1}, \mathbf{u}_t\}$.

Table 6.3 and Table 6.3.2.2 present the average RMSE and SSIM for the different model architecture, training scenarios and dataset. The metrics for the predictions are presented in Figure D.5, D.6 and D.7 in the appendix.

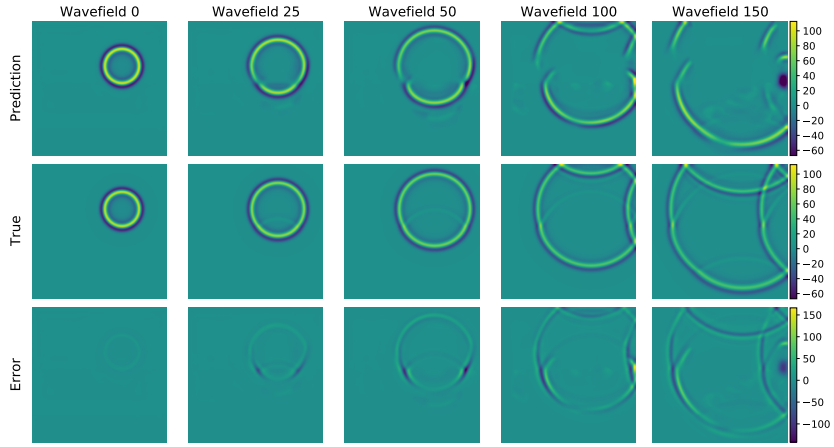
One clearly see from Table 6.3 and Table 6.3.2.2 that surprisingly only one model yields the lowest average RMSE and the higher SSIM on the three different dataset. This was not the case on the homogeneous domain. The predictions generated with the MultiScale model diverge in most of the training scenarios. They can only predict the wavefields for around 25 timesteps and then diverges. Five out of six scenarios yield a RMSE below 10 [Pa] for the U-Net model and 20 [Pa] for the Encoder but the latest does not diverge as the MultiScale. Only the predictions of the U-Net model trained with the PINN RES scenario will be presented since this model performs clearly better than the others. From Table. 6.2 and Table. 6.3.2.2 we see that the model produce better predictions for the heterogeneous domain than for the homogeneous. The SSIM value is 8.14e-01 [-] for the heterogeneous dataset where it was 7.66e-01 [-] on the homogeneous dataset.

From Figure 6.13a we observe that the predictions are almost perfect. The errors are mostly located around the wave-front and they increase once the edges have been reached. The maximum error recorded is 15 [Pa] or - 15 [Pa] and only present at few locations. Some errors also appears when the wave is reflected at the lower layer (y = 300) [m] when it reaches a boundary between two layers. But we notice that the reflected wave is also predicted by the neural network.

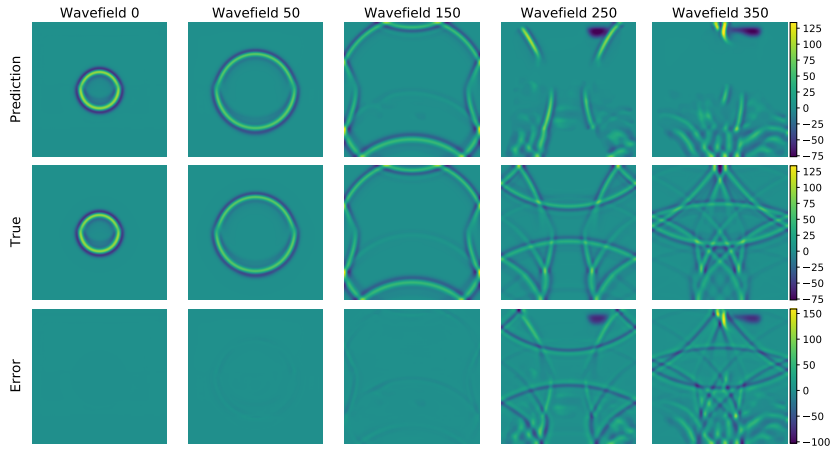
When the same model is tested on the dataset with a different source, the predictions are good until the 100th wavefield. At this point, one part of the wavefield simply disappears. The parts that are predicted remains at the correct location but there is not diffracted wavefield at the boundaries.



(a) Training Predictions



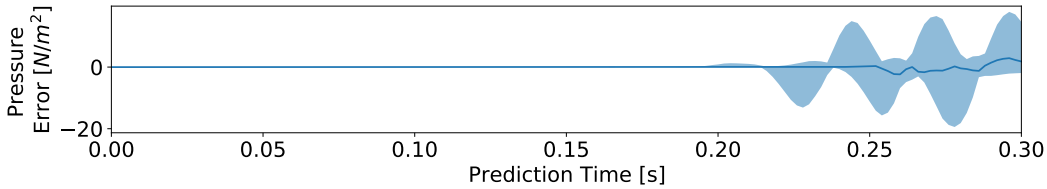
(b) Space Extrapolation Predictions



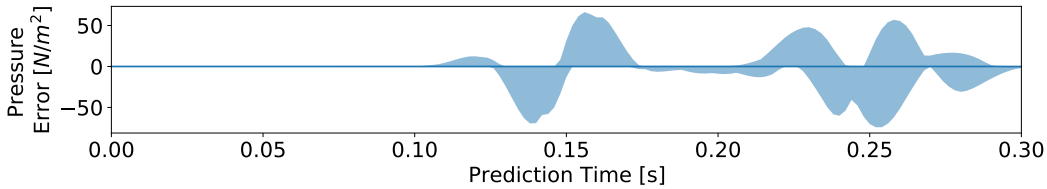
(c) Time Extrapolation Predictions

Figure 6.13 – Heterogeneous domain - Wavefields predictions for U-Net PINN RES

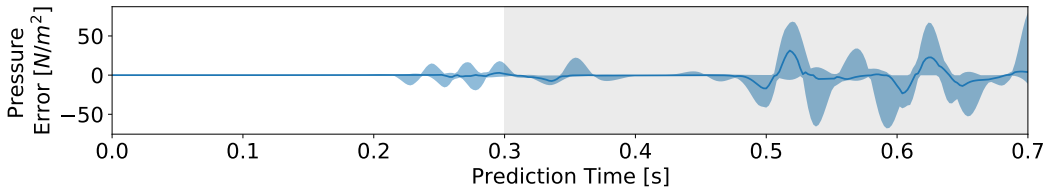
The same behaviour occurs when the model predict the wavefield propagation longer than it was originally planned. The waves simply disappear. One notices from Table that the average SSIM is the lowest between the training and the space extrapolation dataset. However the few waves that remain at the top and at the bottom are located at the correct location. Hence the model is still able to predict the location with the correct velocity at the 250th timestep. For the 350th timestep which is the last timestep to predict at t = 0.7 [s], the solution is not reliable at all.



(a) Training Data - U-Net receivers errors



(b) Space Extrapolation - U-Net receivers errors



(c) Time Extrapolation - receivers errors

Figure 6.14 – U-Net PINN RES receivers errors on the Heterogeneous Layered domain

When looking at the receivers errors for the three dataset (Fig. 6.14), we observe a different behaviour than the receivers for the homogeneous dataset (Fig. 6.9). Despite that the receivers are located at the same locations, they exhibit really different values recorded. Indeed the median value is now around zero which at first sight can be seen as perfect predictions. But one has to keep in mind that because of the different velocities, the wave wont arrive at the same time. It's possible for one receiver to record the wave and later an other. Since predictions are not perfect, oscillations can be seen in Fig. 6.14a. It's even more flagrant when the source is located at (x,y) = (1000,1000) (See Fig. 6.14b). Two receivers record first the wavefields and then 0.1 [s] later for the receivers further away. The furthest receivers don't record the wavefield at

all. The same pattern is observed on this figure as the one in Fig. 6.10. The velocity is slightly delayed at the first receivers, so the model has trouble to adapt to the change of velocity from its source term. Indeed at (x,y) = (750,750) [m] the medium velocity is 2'200 [m/s] and at (x,y) = (1000,1000) [m] it is 2'000 [m/s].

The current best model according to the results presented in Table 6.3 allows to select the U-Net model with the PINN RES scenario as the best performing model. The two will be combined to create a surrogate model.

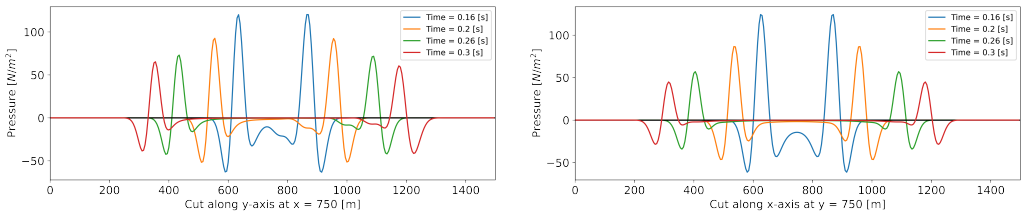
From the whole model selection, we notice that AR PINN MSE scenario from Geneva and Zabaras, 2019 did not yield satisfying results. This difference can by explained by two factors. First of all the authors used a lot of initial states to train their model. On a much powerful GPU, the training lasts for around 9 hours where in this master project the models were trained in 2 hours on one initial state. Second, the acoustic wave equation is of the second order in time and requires two timesteps to compute the next one. From Geneva and Zabaras, 2019, their most complicated case was the coupled Burgers' 2D equation which is of the first order in time. To resolve this issue some manipulations could be performed to obtain an alternative form of the linear acoustic equation.

$$\begin{bmatrix} p \\ u \end{bmatrix}_t + \begin{bmatrix} u_0 & K_0 \\ 1/\rho_0 & u_0 \end{bmatrix} \begin{bmatrix} p \\ u \end{bmatrix}_x = 0 \tag{6.3}$$

With K_0 the bulk modulus of compressibility of the materials and ρ_0 its density.

$$q(x,t) = \begin{bmatrix} p(x,t) \\ u(x,t) \end{bmatrix}$$

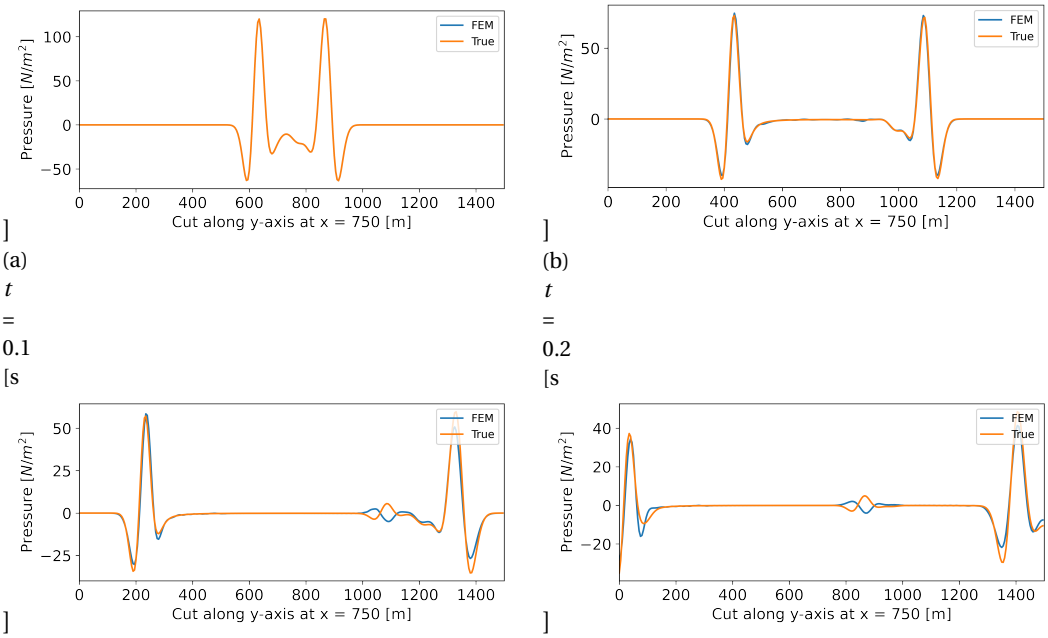
This could increase the reliability of the Finite-Difference scheme. Since the source term was neglected, the initial conditions have to be silent when the first wavefield is computed with FDM (Eq. 3.12 or Eq. 3.14). If the computed wavefields diverge then the physics constrained scenarios will not train the networks correctly. To analyse the reliability of the FDM, a Salvus simulation is analysed and the outputs are compared with the wavefields computed with the FDM. Different timesteps from the layered heterogeneous experiment are plotted along the x and the y to visualise the wavefields. In short, the initial conditions should be silent i.e. the source is located in the middle (x,y) = (750,750) [m] should be zero. If this is not the case, then the source term $f^n_{i,j}$ presented in Eq. 3.11 cannot be simplified and the finite different time stepping method cannot predict the correct evolution.



(a) Initial wavefield along the y-axis for Salvus (b) Initial wavefield along the x-axis for Salvus

Figure 6.15 – Initial wavefield along the x and the y axis for Salvus

From Figure 6.15, one can observe the initial conditions at different time of the Salvus simulation for the heterogeneous layered case along the x and the y axis. In addition of seeing how each layer impact the wavefield, one notice that at $t = 0.16$ [s], the source is clearly not silent. One must wait at least $t = 0.2$ [s] to have silent initial conditions.



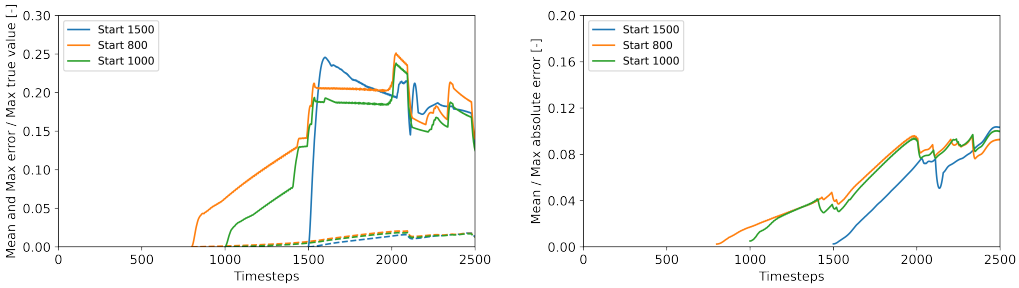
] (a) t = 0.1 [s] (b) t = 0.2 [s] (c) t = 0.3 [s] (d) t = 0.4 [s

Figure 6.16 – Salvus and Finite Time Difference for different time steps for a cut along the y-axis and x = 750

The first predicted wavefield at $t = 0.1$ [s] is the same as the Salvus output and is considered as the true values for the acoustic wave equation. However when the next wavefield are predicted with the FDM, an offset appears around the wave-front with values computed with Salvus.

It will dampen the wave-front since the values predicted with the FDM are smaller than the values from Salvus. Despite all that, one notice on Figure 6.17 that independently of the first initial wavfield, the FDM error relative to Salvus will be at its maximum around 25% but the mean error really low at around 2-3 %. The maximum relative error is that high because the FDM has difficulty in predicting the wave-front perfectly and since it is really sharp, this will lead to the high relative error.

The plot with the ratio of the mean over the maximum relative error for each timesteps can provide insights at the error distribution. The maximum mean error is around 10 % of the maximum error. This means that the most of the errors are really small and that they are located along the wave-front. If the ratio was larger, then one could expect to see large error spread over the entire domain. So we can conclude that the FDM predict well over the domain expect at the wave-front. This can be problematic and improving the method that constrained the training of the neural networks could improve their results, especially if we train recursively over long period such as with the AR PINN MSE scenario.



(a) The mean and the max error over the max true value

(b) The mean over the max error

Figure 6.17 – Error propagation between Salvus and the Finite Difference Method (FDM) when different timesteps from Salvus are considered as the initial conditions. The bold line represents the max error and the dashed line the mean.

The Convolutional Neural Networks (CNN) uses Batch-Norm (BN) layers. There is a well-acknowledged drawback of batch-norm layers is that they require sufficiently large mini-batches. If small mini-batchs are used then model performance is drastically reduced (Singh & Shrivastava, 2019). The models were trained with mini-batches of 4 samples. By increasing the batch size, the samples will be closer to the population for inner activation (Bjorck et al., 2018). Maybe Batch-Norm layers are not suited for physics informed neural networks.

6.3.3 Surrogate Modelling

For the surrogate modelling, we aim at simulating the wavefields that would occur in lab experiments. Hence the sources and receivers are only located on the edges of the samples. The goal is to analyse how the model predict the wavefield for a set of test receivers not seen during the training. In this way, we could artificially increase the number of receivers placed on the edge of the samples and model the crack propagation with greater accuracy for example. As training data must be similar to test data, the model is trained with 8 sources placed on the sides and 4 sources are used to assess the performances of the models. The results for the three surrogate models created (Homogeneous, Layered Heterogeneous and Crack Heterogeneous) are presented in Table 6.5 for the U-Net model trained with the PINN-RES scenario. Training one surrogate model with wavefields from eight different sources requires around 20 [h] on a Nvidia Quattro P2000.

		RMSE	SSIM	Mean Abs Error	Median True/ Max True
Homogeneous	Train	2.36e+01	7.67e-01	8.59e+00	3.27e-03
	Test	2.44e+01	7.96e-01	8.49e+00	2.53e-03
Layered	Train	1.08e+01	8.73e-01	3.87e+00	9.57e-04
	Test	2.08e+01	7.86e-01	7.6e+00	2.18e-03
Crack 1	Train	9.41e+00	7.77e-01	4.01e+00	4.39e-03
	Test	1.64e+01	7.63e-01	5.96e+00	3.44e-03
Crack 2	Train	1.51e+01	7.89e-01	5.62e+00	3.82e-03
	Test	1.71e+01	8.22e-01	5.77e+00	2.99e-03

Table 6.5 – Mean Metrics over the 150 predicted timesteps for the train and test simulations

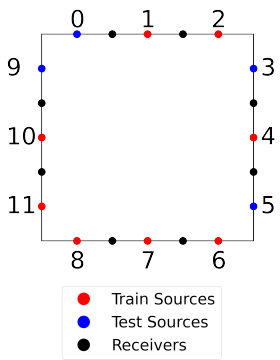
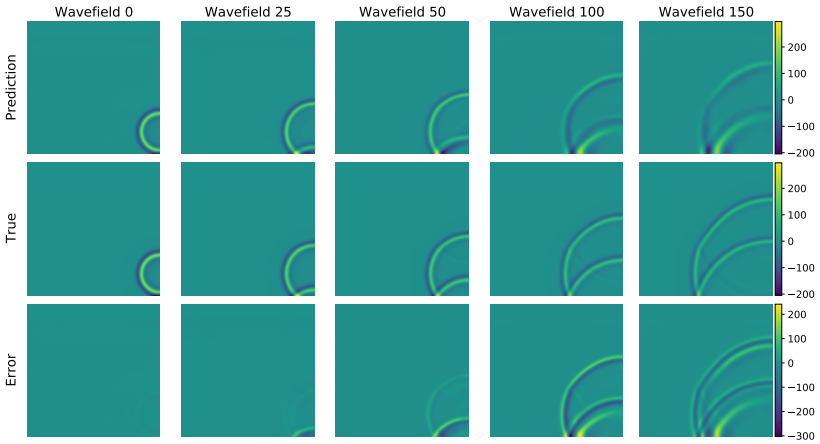


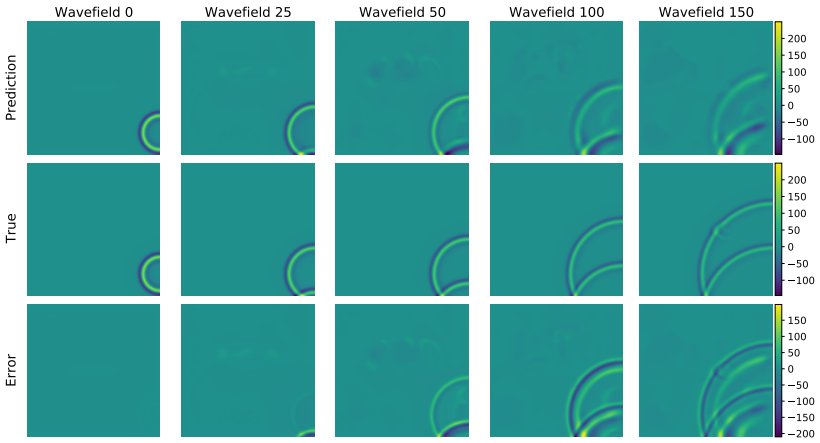
Figure 6.18 – Train and test sources for the simulations

On Table. 6.5 the results shown for the Homogeneous, Layered and Crack 1 are trained on 8 receivers as shown on Fig 6.18 and tested on the 4 remaining. The training and test sources are selected randomly but they are the same for all the surrogate models. However for Crack 2 the same model used in Crack 1 is used to compute the metrics for both train and test data.

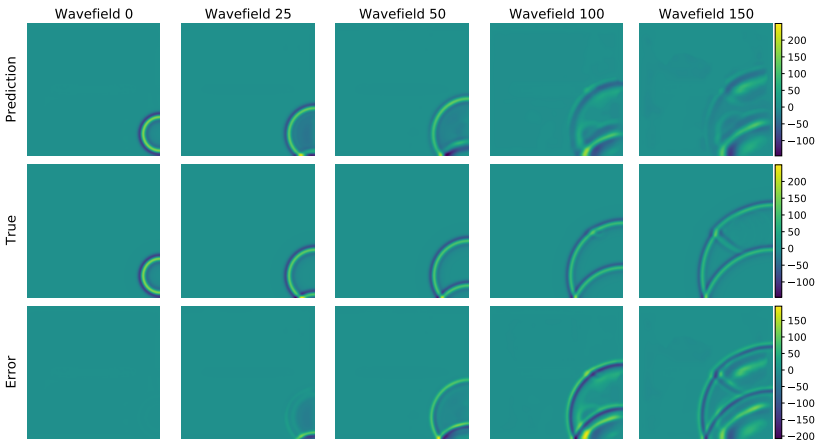
From the SSIM all values are around 0.8 [-] which indicates that the predicted images are clear and does not present to much noise. All RMSE values computed on the training sources are lower than on the test dataset. There is an increase of 3 % for the Homogeneous dataset and 73 % for the Crack 1 dataset. However the RMSE only increases by 13 % on the Crack 2 between the train receivers and the test receivers. There is also an increase in the RMSE between the Crack 1 and Crack 2, but this was expected since the model trained in Crack 1 is used to predict



(a) Test dataset on the heterogeneous layered domain



(b) Test dataset on the heterogeneous small crack domain



(c) Test dataset on the heterogeneous small crack domain

Figure 6.19 – Predictions for Event 5 for the surrogate models for the layered, small crack and big crack

Crack 2. However, the SSIM between the Crack 1 and Crack 2 are higher on the latest dataset even though it is totally new dataset. The really small ratio of the median tells us that the predicted wavefield are not diverging as it can be observed in the Figure 6.19.

On Figure 6.19 one can see the wavefields predictions and the errors for the 5th event (receiver #5) on the heterogeneous dataset. Unfortunately longer predictions are required if we want to record receiver from sources located at the other side of the sample. The same number of timesteps was used as the original homogeneous case but here the wave-front has just the time to be diffracted from the crack or the first layer. The pressure recorded by the receivers along the edges does not record diffracted wavefield and there are too many wavefield predicted in the beginning where nothing important occurs. We could increase the number of predicted timesteps to at least 350 or train the model with bigger timesteps to reduce the training time. Fig. 6.20 illustrates the 100th recursively predicted wavefield when the U-Net PINN RES is trained for a different number of epochs. Since we added more data to train the surrogate model (more initial conditions as Geneva and Zabaras, 2019) suggested), it will be logical to train more the model.

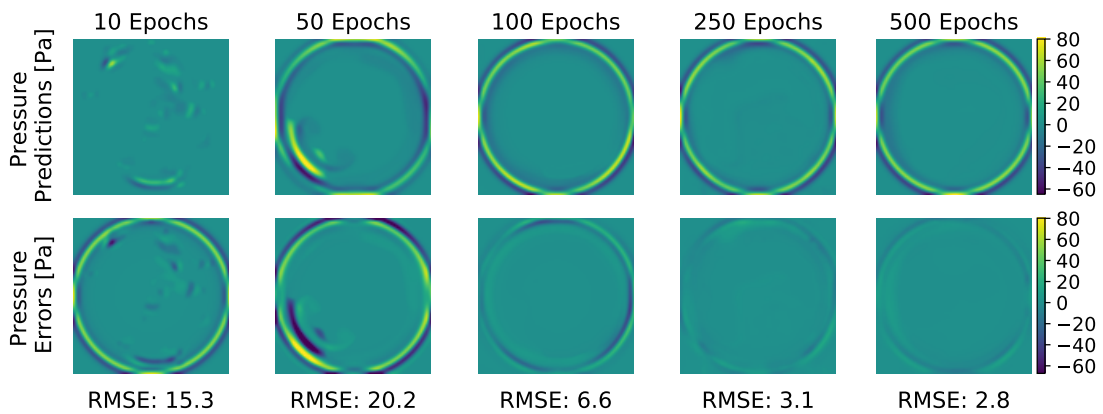


Figure 6.20 – Evolution of the RMSE between the predicted and the true 100th wavefields for the U-Net PINN RES training scenarios trained for different number of epochs.

We observe on Fig. 6.20 that the RMSE is lower when the model is trained for 10 epochs than 50. This can be explained since the model is not trained at all for 10 epochs and will tend to predict only zero values since more of the inputs seen by the network are composed of zeros. When 50 epochs are used, then the model understands that it should predict the wavefield but it does with the wrong velocity. A negative pressure is predicted where positive pressure is expected for example. One need at least 100 epochs to obtain acceptable predictions. The learning rate μ of the model can also be adjusted to reach a better local minimum in the loss

function.

When the metrics are computed for the Crack 1 and Crack 2, we observe that the RMSE for the training data is the lowest for the test data on the Crack 1 and finally the predictions for Crack 2 are overlaying each other independently of the receivers locations. The same behaviour is observed for the Mean absolute error. However for the SSIM there is not a clear gap, predictions are better for Crack 1 but after timestep 100 the Crack 2 yields a higher SSIM.

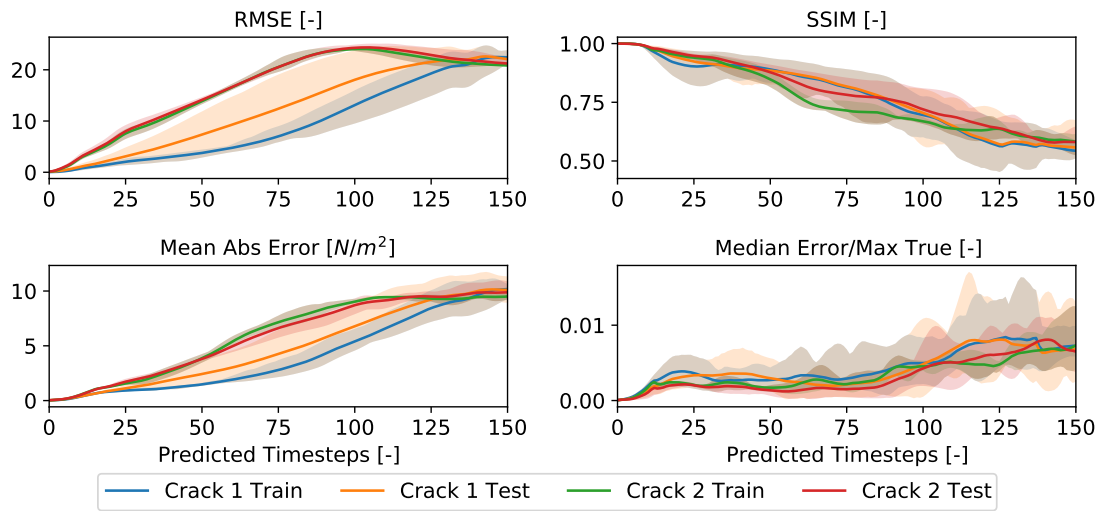
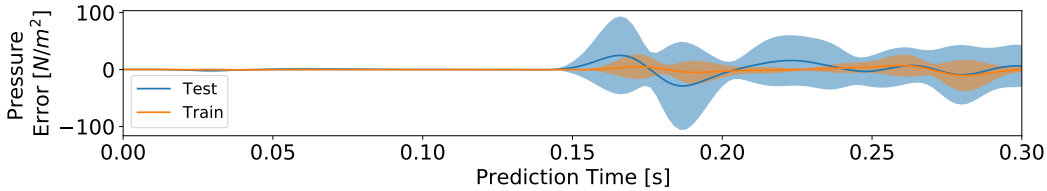


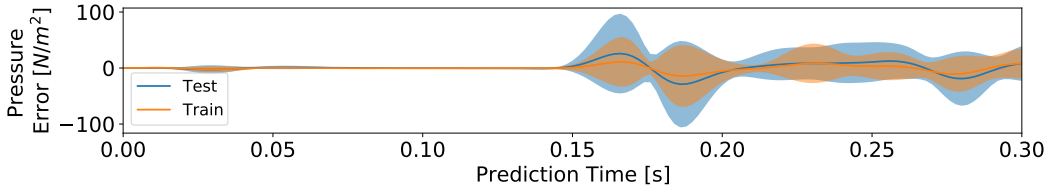
Figure 6.21 – Metrics for the heterogeneous small and big cracks domain

The errors at the receivers presents the same behaviour as shown on figure **??**. There is a really small error on the training set compared to the test set on the Crack 1 dataset. For the Crack 2, both train and test error interval are bigger, resembling like the test set of the Crack 1 errors predictions. Considering the arrival time of wavefields at their closest receiver from the source. The wave-front should arrive at approximately 0.1 [s] (250 [m]/2500[m/s]), and the errors are 0 which mean a really good prediction. The errors appear exactly at t = 0.2 [s] which corresponds to the arrival of wave reflected on the lower edge of the domain (See Wavefield 100 on Fig 6.19c).

The networks predictions are far quicker than FEM computation, but still a drawback is present and one must between the speed and the accuracy at the moment. FEM is still required to generate the initial conditions and wavefield silent initial conditions has to be extracted. Despite this, if we consider the time taken then the predictions performed by the neural are 20 times faster than Salvus. If the metrics are computed then it is only 10 times, where most of the time taken to update the inputs.



(a) Error at the receivers for the train and test events for the heterogeneous small crack domain



(b) Error at the receivers for the train and test events for the heterogeneous big crack domain

Figure 6.22 – Error at the receivers locations for the experiments with sources placed on the edges

7 Conclusion

This master project looked at physics informed neural networks (PINN) as surrogate models and their ability to replace Finite Element Methods (FEMs) to simulate physical problems. FEMs are resource intensive and the simulations need to be rerun if some parameters changes in the problem.

The ability of neural network to approximate any function was first tested, which led to the implementation of an multi-layer perceptron (MLP) to predict the Burgers' equation in 1D. For the acoustic wave equation, the choice of working with convolutional neural networks (CNN) was taken. Indeed, this kind of models are particularly suited when working with large images. They decrease the number of parameters compared to MLP and the training times. They also lead in better extrapolation results since the convolutional layers used in those models can detect some patterns in the images independently from their locations. They are the logical approach to use when working with wavefields if we represent them as images.

The workflow implemented to obtain surrogate models that are able to plot the pressure at certain locations was to predict the wavefield propagation given initial conditions. The wavefield propagation is recursively predicted from the initial wavefield and the pressure extracted. Indeed, in laboratory experiments some sources and receivers are placed along the edges being and generate some waves. With the signals recorded, inversion methods are employed to compute the material properties or the location of a crack for example.

When the whole wavefield propagation is predicted, one can extract the pressure at any location on the domain. This kind of methodology provides a much greater generalisation than a model that predicts only the receivers pressure at specific locations. With the retained methodology we can add or remove receivers as our will, change their locations without having

to change anything in the model or to retrain it.

Three different model architectures, respectively the MultiScale, Encoder-Decoder and the U-Net, and six different training scenarios were tested in order to create the best surrogate model. Three of the scenarios were data driven and trained in a supervised scheme. This means that the target values are provided to the model during its training and three were physics constrained. For the physics constrained scenarios applied to CNN, a finite difference method (FDM) was used to enforce the physics as proposed by Geneva and Zabaras, 2019. Indeed since it was not possible to use the automatic differentiation to compute the partial derivatives the residuals of the wave equation are minimised to enforce physics.

The U-Net model trained with the semi-supervised PINN-RES scenario yields the best results on the layered heterogeneous domain and was chosen as a surrogate model. In this training scenario, the predictions of the neural networks are used to compute the residuals of the wave equation with the FDM. To select this model, the average RMSE, SSIM, average absolute error and the ratio between the median error and the maximum true value are computed on an homogeneous and an heterogeneous dataset. The models are trained with the wavefields when a source located and the middle. From this training, we test if the same model can predict the propagation for a new source or a longer time than the original training data (space and time extrapolation). If the performances are similar then the model will have learn the physics underlying the problem. The MulstiScale architecture implemented in Weiqiang et al., 2017 and Alguacil et al., n.d. diverges for the most complex experiments. Indeed in their papers, the model did not predict the propagation for so many timesteps. Between the Encoder-Decoder architecture and U-Net, the latest yields better results in average as shown in Fotiadis et al., 2020. Some relevant benchmarks results and metrics are provided which were missing in this field of research. They are also reproducible since relatively simple simulations are performed.

The surrogate model shows great potential to predict the wavefields for heterogeneous media. With a SSIM of 0.822 [-] and an average error of 5.77 [Pa] obtained on the large crack test set the results seem promising. The pressures at the receivers could be compared with real laboratory values to see if they could be used to effectively locate the crack in 2D before extending the surrogate modelling for 3D simulations.

The models were trained only for a small range of variables. This includes a slightly different domain and source location. However the sources generating the wavefields have the same frequency and the domain's size is the same. Further researches could be conducted to analyse

the sensitivity of the results for different domain's sizes and shapes or different frequencies.

To improve the existing models, more training data could be used with more different initial conditions. Generative Auto-Encoder Networks (GANs) have also proven to be fairly successful in generating samples from high-dimensional distributions in tasks such as images. Used in Weiqiang et al., 2017 to train their model, they led to really small improvements in the wavefields predictions. Such models could be used to generate the initial conditions given a media a source function which would totally remove the need of FEMs.

Further work can be conducted from the signal recorded by the receivers and neural networks can be trained to predict the source location of an earthquake for example with the Time Reversal Method (Larmat et al., 2011) or the physical parameters such as the velocity with inversion methods (Shukla et al., 2020).

A Theory

In Geneva and Zabaras, 2019 the authors presents this theorem to prove that the L_2 minimisation between the model's prediction and a numerical time integration is similar to the L_2 minimisation of a discritsed PDE residuals. The method is formulated by discretising the temporal derivative as shown in this example for a single time step $n \to n+1$ considering the forward Euler time integration scheme.

$$\frac{u^{n+1} - u^n}{\Delta t} = -F_{\Delta x}\left(x, u^n\right) \qquad \hat{u}^{n+1} = T_{\Delta t}\left(u^n, F_{\Delta x}\right) = u^n - \Delta t F_{\Delta x}\left(x, u^n\right) \tag{A.1}$$

u^{n+i} is the prediction from the neural network.

$$\begin{aligned}
L &= \frac{1}{M}\sum_{i=1}^{M} ||\hat{u}_i - u_i||_2^2, \qquad u_i = T_{\Delta t}\left(u^n, F_{\Delta x}\right) \\
&= \frac{1}{M}\sum_{i=1}^{M} \left|\left|u_i^n - \Delta t F\left(x_i u_i^n\right) - u_i^{n+1}\right|\right|_2^2 \\
&= \frac{1}{M}\sum_{i=1}^{M} \left|\left|u_i^{n+1} - u_i^n + \Delta t F\left(x_i u_i^n\right)\right|\right|_2^2 \\
&= \frac{\Delta t^2}{M}\sum_{j=1}^{M} \left|\left|\frac{u_i^{n+1} - u_i^n}{\Delta t} + F\left(x_i u_i^n\right)\right|\right|_2^2 \\
&= \frac{\Delta t^2}{M}\sum_{i=1}^{M} \left|\left|R(u_i^{n+1})\right|\right|_2^2 = \frac{\Delta t^2}{M}\sum_{j=1}^{M} \left|\left|R\left(f\left(u, w\right)\right)\right|\right|_2^2
\end{aligned} \tag{A.2}$$

B Experiments

This section contains all the specific tables to reproduce the Salvus experiments.

Table B.1 – Salvus - Homogeneous domain parameters

Source Location 1 & 2 [m]	x	750											
(Training and Time Extrapolation)	y	750											
Source Location 3 [m]	x	1000											
(Space extrapolation)	y	1000											
Source Location 4 [m]	x	0	0	0	250	750	1250	1500	1500	1500	1250	750	250
(Surogate Model)	y	250	750	1250	1500	1500	1500	250	750	1250	0	0	0
Material properties 1, 2, 4 and 4 $v_s[m/s]$; $\rho[kg/m^3]$		2500; 2200											
Simulation properties 1, 3 and 4 $T_0[s]$; $T_{max}[s]$; $T_s[s]$		-0.1; 0.4; 0.0002											
Simulation properties 2 $T_0[s]$; $T_{max}[s]$; $T_s[s]$		-0.1; 0.8; 0.0002											

Table B.2 – Salvus - Layered heterogeneous domain parameters

Source Location 1 & 2 [m]	x	750											
(Training and Time Extrapolation)	y	750											
Source Location 3 [m]	x	1000											
(Space extrapolation)	y	1000											
Source Location 4 [m]	x	0	0	0	250	750	1250	1500	1500	1500	1250	750	250
(Surogate Model)	y	250	750	1250	1500	1500	1500	250	750	1250	0	0	0
Boundaries between layers 1, 2 and 3 [m]		0; 300; 800; 1450; 1500											
Boundaries between layers 4 [m]		0; 550; 800; 1100; 1350; 1500											
v_s values 1, 2 and 3 $[m/s]$		2600; 2200; 2000; 2300											
v_s values 4 $[m/s]$		2200; 2000; 2100; 2300; 2600											
ρ values 1, 2 and 3 $[kg/m^3]$		2213; 2123; 2073; 2146											
ρ values 4 $[kg/m^3]$		2123; 2073; 2098; 2146; 2213											
Simulation properties 1, 3 and 4 $T_0[s]$; $T_{max}[s]$; $T_s[s]$		-0.1; 0.4; 0.0002											
Simulation properties 2 $T_0[s]$; $T_{max}[s]$; $T_s[s]$		-0.1; 0.8; 0.0002											

Table B.3 – Salvus - Cracked heterogeneous domain parameters

Source Location 1 and 2 [m]	x	0	0	0	250	750	1250	1500	1500	1500	1250	750	250
(Surogate Model for Crack 1 & 2)	y	250	750	1250	1500	1500	1500	250	750	1250	0	0	0
Fracture center location 1 and 2 [m]	x	750											
	y	750											
Fracture dimension 1	a = 250 [m], b = 40 [m]												
Fracture dimension 2	a = 500 [m], b = 40 [m]												
v_s values 1 and 2 $[m/s]$	2500 (Background); 2000 (Crack)												
ρ values 1 and 2 $[kg/m^3]$	2200 (Background); 1800 (Crack)												
Simulation properties 1 and 2 $T_0[s]$; $T_{max}[s]$; $T_s[s]$	-0.1; 0.4; 0.0002												

C Methodology

C.1 MultiScale parameters

Table C.1 – MultiScale neural network parameters

Layer Name		Features maps (Inputs)	Features Maps (Outputs)	Kernel Size
Block (n/4 x n/4)	Conv Layer 1	4 x 75 x 75	32 x 75 x 75	3 x 3
	Conv Layer 2	32 x 75 x 75	64 x 75 x 75	3 x 3
	Conv Layer 3	64 x 75 x 75	32 x 75 x 75	3 x 3
	Conv Layer 4	32 x 75 x 75	1 x 75 x 75	3 x 3
Block (n/2 x n/2)	Conv Layer 1	5 x 150 x 150	32 x 150 x 150	3 x 3
	Conv Layer 2	32 x 150 x 150	64 x 150 x 150	3 x 3
	Conv Layer 3	64 x 150 x 150	128 x 150 x 150	3 x 3
	Conv Layer 4	128 x 150 x 150	64 x 150 x 150	3 x 3
	Conv Layer 5	64 x 150 x 150	32 x 150 x 150	3 x 3
	Conv Layer 6	32 x 150 x 150	1 x 150 x 150	3 x 3
Block (n x n)	Conv Layer 1	5 x 300 x 300	32 x 300 x 300	3 x 3
	Conv Layer 2	32 x 300 x 300	64 x 300 x 300	3 x 3
	Conv Layer 3	64 x 300 x 300	128 x 300 x 300	3 x 3
	Conv Layer 4	128 x 300 x 300	64 x 300 x 300	3 x 3
	Conv Layer 5	64 x 300 x 300	32 x 300 x 300	3 x 3
	Conv Layer 6	32 x 300 x 300	1 x 300 x 300	3 x 3

C.2 Encoder - Decoder parameters

Table C.2 – Encoder neural network parameters

Layer Name		Features maps (Inputs)	Features Maps (Outputs)	Kernel Size	Stride
Encoding	Conv Layer 1	4 x 300 x 300	32 x 300 x 300	3 x 3	-
	Max-Pooling 1	32 x 300 x 300	32 x 150 x 150	-	2x2
	Conv Layer 2	32 x 150 x 150	64 x 150 x 150	3x3	-
	Max-Pooling 2	64 x 75 x 75	64 x 75 x 75	-	2 x 2
	Conv Layer 3	64 x 75 x 75	128 x 75 x 75	3 x 3	-
	Max-Pooling 3	128 x 75 x 75	128 x 37 x 37	-	2 x 2
	Conv Layer 4	128 x 37 x 37	256 x 37 x 37	3 x 3	-
	Max-Pooling 4	256 x 37 x 37	256 x 18 x 18	-	2 x 2
Bottleneck		256 x 18 x 18	512 x 18 x 18	3 x 3	
Decoding	Up-Sample 1	512 x 18 x 18	256 x 37 x 37	-	2 x 2
	Conv Layer 5	256 x 37 x 37	256 x 37 x 37	3 x 3	-
	Up-Sample 2	256 x 37 x 37	128 x 75 x 75	-	2 x 2
	Conv Layer 6	128 x 75 x 75	128 x 75 x 75	3 x 3	-
	Up-Sample 3	128 x 75 x 75	64 x 150 x 150	-	2 x 2
	Conv Layer 7	64 x 150 x 150	64 x 150 x 150	3 x 3	-
	Up-Sample 4	64 x 150 x 150	32 x 300 x 300	-	2 x 2
	Conv Layer 8	32 x 300 x 300	32 x 300 x 300	3 x 3	-
	Conv Layer 9	32 x 300 x 300	1 x 300 x 300	3 x 3	-

C.3 U-Net parameters

Table C.3 – U-Net neural network parameters

Layer Name		Features maps (Inputs)	Features Maps (Outputs)	Kernel Size	Stride
Encoding	Conv Layer 1	4 x 300 x 300	32 x 300 x 300	3 x 3	-
	Max-Pooling 1	32 x 300 x 300	32 x 150 x 150	-	2x2
	Conv Layer 2	32 x 150 x 150	64 x 150 x 150	3x3	-
	Max-Pooling 2	64 x 75 x 75	64 x 75 x 75	-	2 x 2
	Conv Layer 3	64 x 75 x 75	128 x 75 x 75	3 x 3	-
	Max-Pooling 3	128 x 75 x 75	128 x 37 x 37	-	2 x 2
	Conv Layer 4	128 x 37 x 37	256 x 37 x 37	3 x 3	-
	Max-Pooling 4	256 x 37 x 37	256 x 18 x 18	-	2 x 2
Bottleneck		256 x 18 x 18	512 x 18 x 18	3 x 3	
Decoding	Up-Sample 1	512 x 18 x 18	256 x 37 x 37	-	2 x 2
	Conv Layer 5	(256+256) x 37 x 37	256 x 37 x 37	3 x 3	-
	Up-Sample 2	256 x 37 x 37	128 x 75 x 75	-	2 x 2
	Conv Layer 6	(128+128) x 75 x 75	128 x 75 x 75	3 x 3	-
	Up-Sample 3	128 x 75 x 75	64 x 150 x 150	-	2 x 2
	Conv Layer 7	(64+64) x 150 x 150	64 x 150 x 150	3 x 3	-
	Up-Sample 4	64 x 150 x 150	32 x 300 x 300	-	2 x 2
	Conv Layer 8	(32+32) x 300 x 300	32 x 300 x 300	3 x 3	-
	Conv Layer 9	32 x 300 x 300	1 x 300 x 300	3 x 3	-

D Results

D.1 Acoustic - Homogeneous Domain

Table D.1 – Average metrics for the homogeneous domain on 150 timesteps predictions

(a) Average RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.79e+01	1.98e+01	1.85e+01
L2	1.05e+01	4.74e+00	7.88e+00
L2 GDL	7.72e+00	2.88e+00	4.34e+00
L2 GDL MAE	1.46e+01	2.56e+00	2.80e+00
PINN MSE	1.4e+01	7.18e+00	1.29e+01
PINN RES	2.39e+00	9.85e+00	3.28e+00

(b) Average SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	2.83e-01	2.70e-01	3.63e-01
L2	6.93e-01	9.08e-01	8.32e-01
L2 GDL	7.92e-01	9.56e-01	7.87e-01
L2 GDL MAE	5.42e-01	9.59e-01	9.36e-01
PINN MSE	5.37e-01	8.62e-01	8.04e-01
PINN RES	9.38e-01	8.41e-01	8.21e-01

(c) Average Absolute Error

	U-Net	Encoder	MultiScale
AR PINN MSE	9.96e+00	1.19e+01	1.12e+01
L2	5.44e+00	2.04e+00	3.79e+00
L2 GDL	3.71e+00	1.13e+00	2.31e+00
L2 GDL MAE	8.7e+00	1.04e+00	1.16e+00
PINN MSE	7.98e+00	2.99e+00	5.29e+00
PINN RES	1.23e+00	3.79e+00	2.03e+00

(d) Median Error over Max True

	U-Net	Encoder	MultiScale
AR PINN MSE	6.19e-02	7.36e-02	6.01e-02
L2	1.16e-02	1.69e-03	6.68e-03
L2 GDL	8.01e-03	9.3e-04	8.32e-03
L2 GDL MAE	4.31e-02	9.25e-04	1.66e-03
PINN MSE	2.50e-02	1.37e-03	1.03e-02
PINN RES	3.62e-03	1.54e-03	1.45e-02

Table D.2 – Average metrics for the homogeneous domain and a different source location on 150 timesteps predictions

(a) Average RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.74e+01	1.96e+01	2.07e+01
L2	1.12e+01	1.03e+01	1.51e+01
L2 GDL	1.22e+01	9.90e+00	4.70e+00
L2 GDL MAE	1.44e+01	1.11e+01	4.76e+00
PINN MSE	1.47e+01	1.1e+01	3.86e+00
PINN RES	9.16e+00	1.19e+01	6.53e+00

(b) Average SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	3.33e-01	3.38e-01	4.36e-01
L2	6.33e-01	8.25e-01	8.44e-01
L2 GDL	7.03e-01	8.23e-01	8.62e-01
L2 GDL MAE	5.79e-01	8.13e-01	9.18e-01
PINN MSE	5.65e-01	7.94e-01	9.12e-01
PINN RES	7.76e-01	7.79e-01	8.43e-01

(c) Average Absolute Error

	U-Net	Encoder	MultiScale
AR PINN MSE	9.74e+00	1.12e+01	1.19e+01
L2	6.1e+00	4.31e+00	5.29e+00
L2 GDL	6.41e+00	4.24e+00	2.34e+00
L2 GDL MAE	8.42e+00	4.69e+00	1.85e+00
PINN MSE	8.35e+00	4.94e+00	1.79e+00
PINN RES	4.70e+00	5.4e+00	2.7e+00

(d) Median Error over Max True

	U-Net	Encoder	MultiScale
AR PINN MSE	4.33e-02	4.49e-02	3.86e-02
L2	1.50e-02	3.81e-03	4.73e-03
L2 GDL	1.34e-02	3.73e-03	5.06e-03
L2 GDL MAE	3.22e-02	3.52e-03	1.5e-03
PINN MSE	2.54e-02	5.19e-03	2.54e-03
PINN RES	1.16e-02	5.69e-03	9.54e-03

Table D.3 – Results Tables Time Extrapolation - Homogeneous Domain

(a) Average RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.85e+01	1.97e+01	1.79e+01
L2	1.74e+01	1.28e+01	5.38e+04
L2 GDL	1.65e+01	1.23e+01	9.72e+00
L2 GDL MAE	1.92e+01	1.2e+01	1.28e+01
PINN MSE	1.89e+01	1.45e+01	9.18e+04
PINN RES	1.21e+01	1.59e+01	1.22e+03

(b) Average SSIM

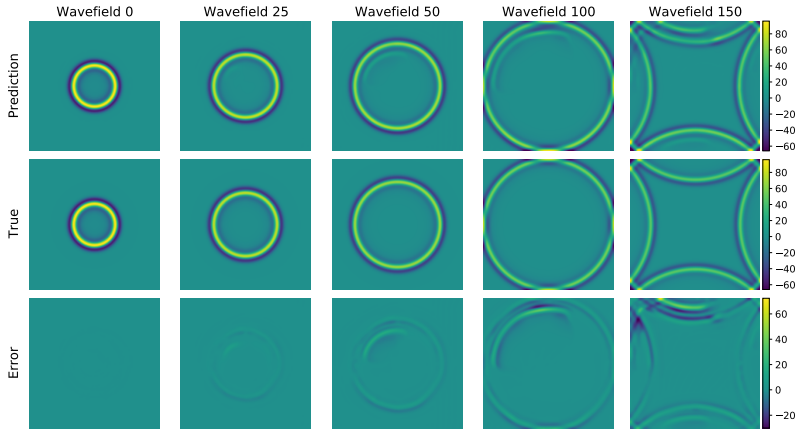
	U-Net	Encoder	MultiScale
AR PINN MSE	2.05e-01	1.81e-01	3.13e-01
L2	4.14e-01	6.32e-01	4.65e-01
L2 GDL	4.80e-01	6.46e-01	5.68e-01
L2 GDL MAE	3.66e-01	6.47e-01	6.25e-01
PINN MSE	3.48e-01	5.93e-01	4.18e-01
PINN RES	5.86e-01	5.83e-01	4.97e-01

(c) Average Absolute Error

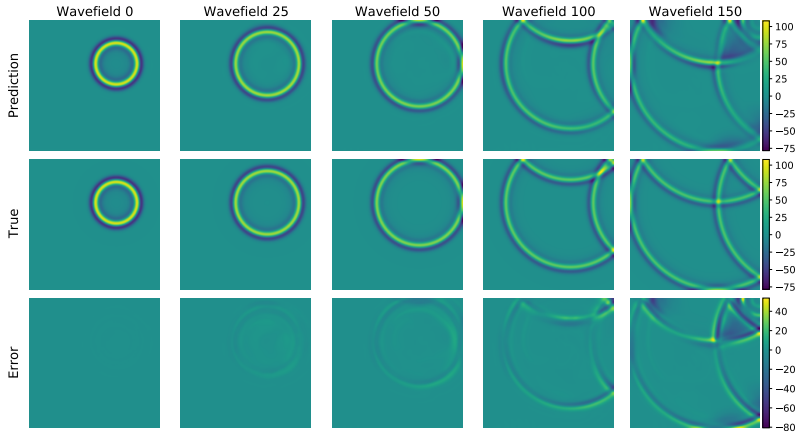
	U-Net	Encoder	MultiScale
AR PINN MSE	1.18e+01	1.29e+01	1.12e+01
L2	1.09e+01	7.39e+00	5.31e+03
L2 GDL	1.03e+01	7.22e+00	5.97e+00
L2 GDL MAE	1.21e+01	7.03e+00	7.49e+00
PINN MSE	1.20e+01	8.24e+00	5.09e+04
PINN RES	7.88e+00	8.59e+00	2.4e+02

(d) Median Error over Max True

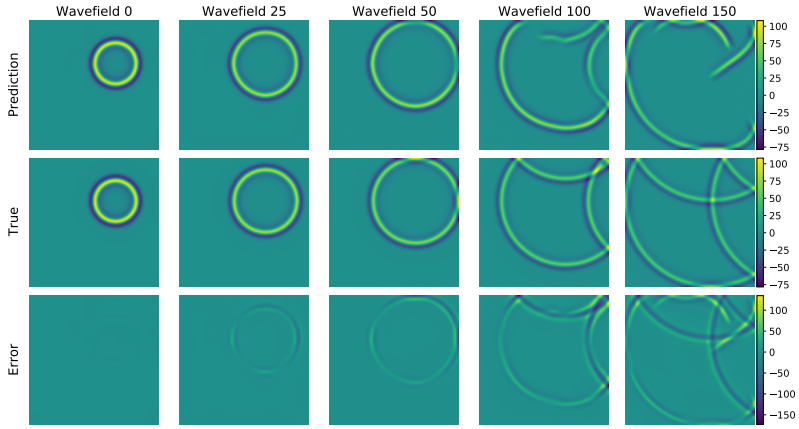
	U-Net	Encoder	MultiScale
AR PINN MSE	9.38e-02	9.31e-02	6.78e-02
L2	7.1e-02	3.7e-02	1.03e-01
L2 GDL	6.57e-02	4.08e-02	3.44e-02
L2 GDL MAE	7.62e-02	3.88e-02	4.96e-02
PINN MSE	7.07e-02	4.20e-02	3.33e+02
PINN RES	5.92e-02	4.05e-02	2.79e-01



(a) Wavefields predictions for MultiScale L2 GDL MAE training scenario for the homogeneous domain on the training data



(b) Wavefields predictions for the MultiScale model with the PINN MSE training scenario for the homogeneous domain with the source located at (x,y) = (1000,1000)



(c) Wavefields predictions for the Encoder model with the L2 GDL training scenario for the homogeneous domain with the source located at (x,y) = (1000,1000)

Figure D.1 – Homogeneous domain - Wavefields predictions

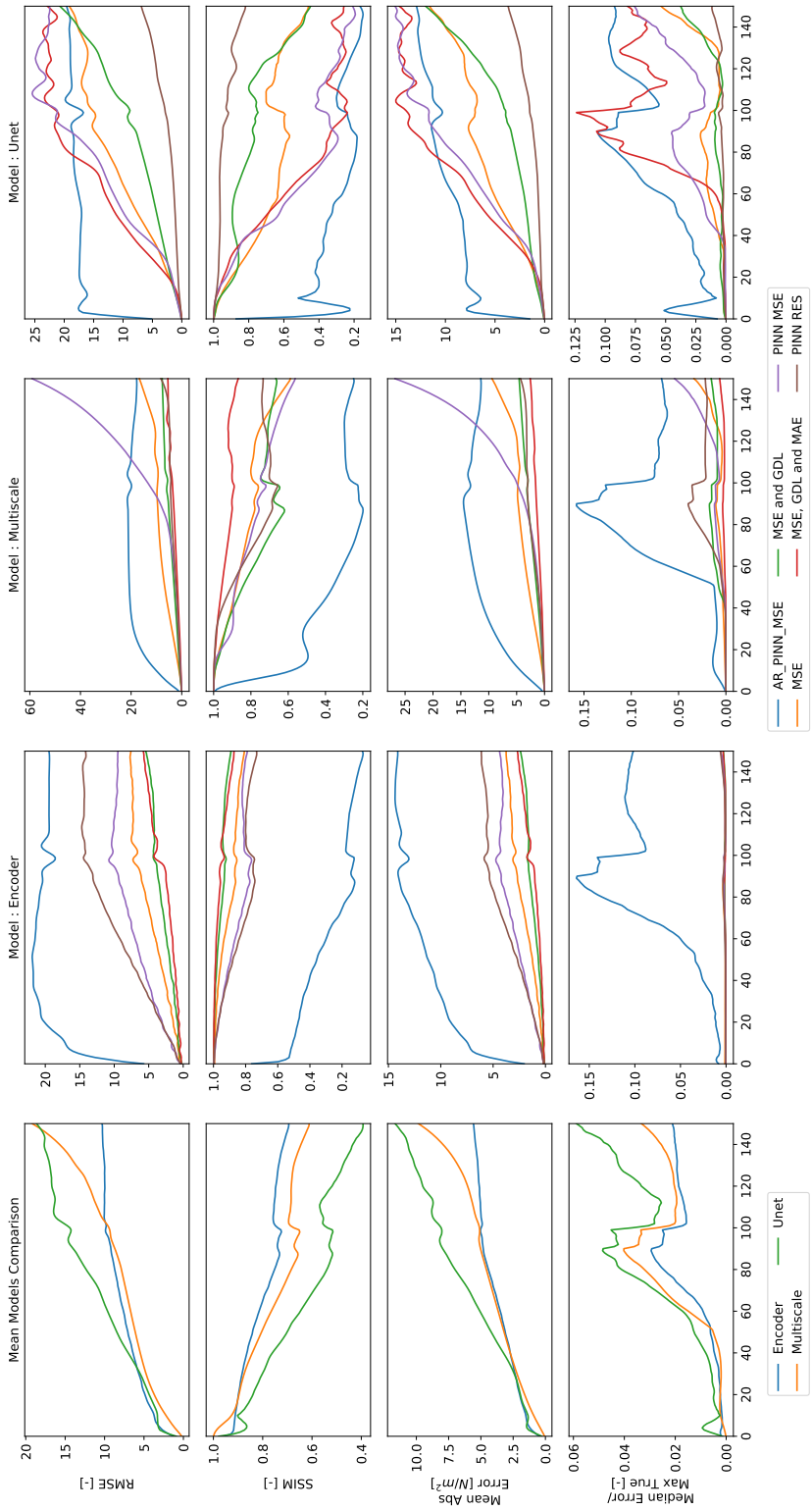


Figure D.2 – Metrics computed on recursive predictions for the homogeneous domain

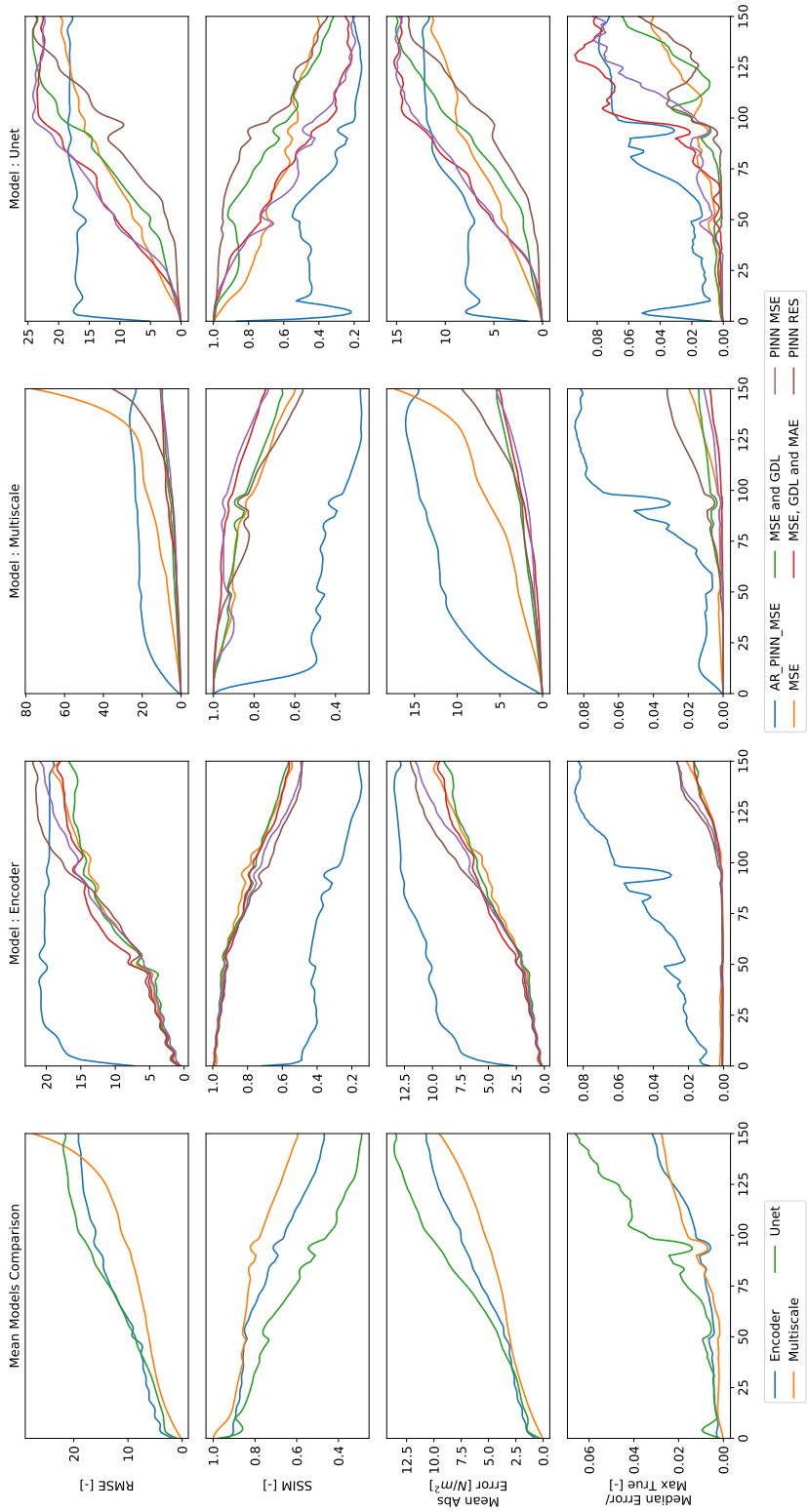


Figure D.3 – Relevant metrics for the homogeneous domain with the space extrapolation

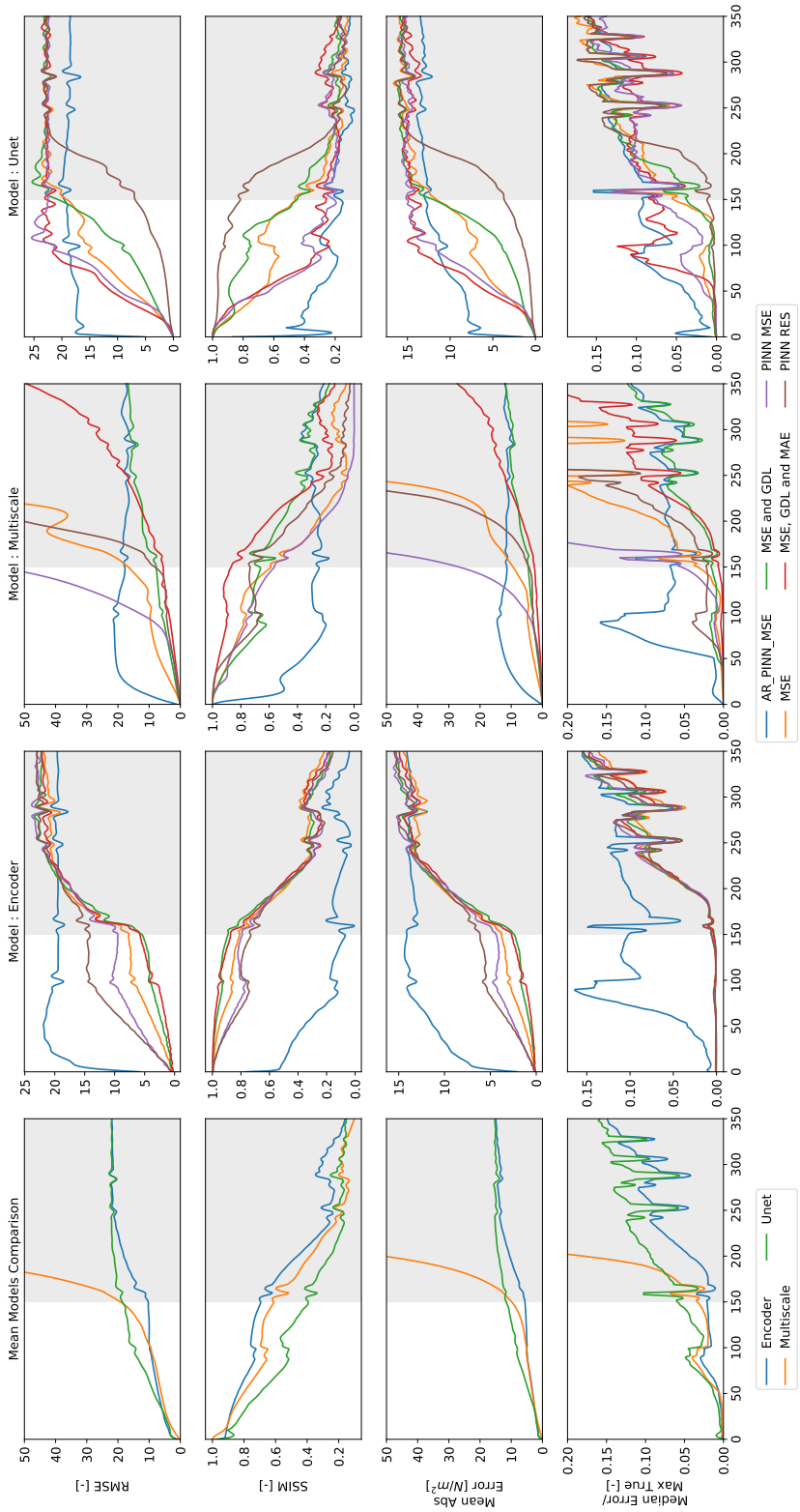


Figure D.4 – Relevant metrics for the homogeneous domain with the time extrapolation

D.2 Acoustic - Heterogeneous Layered Domain

Table D.4 – Results on the layered training data

(a) Average RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.56e+01	1.51e+01	1.97e+01
L2	7.05e-01	6.68e+00	1.87e+03
L2 GDL	8.03e-01	7.59e+00	8.73e+00
L2 GDL MAE	2.61e+00	5.73e+00	inf
PINN MSE	3.17e+00	6.44e+00	1.85e+01
PINN RES	1.01e+00	9.96e+00	7.62e+00

(b) Average SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	1.95e-01	5.96e-01	1.19e-01
L2	9.92e-01	8.68e-01	4.23e-01
L2 GDL	9.91e-01	8.48e-01	4.82e-01
L2 GDL MAE	9.43e-01	8.87e-01	2.81e-01
PINN MSE	9.31e-01	8.89e-01	4.24e-01
PINN RES	9.75e-01	8.07e-01	8.23e-01

(c) Average Absolute Error

	U-Net	Encoder	MultiScale
AR PINN MSE	9.5e+00	6.56e+00	1.45e+01
L2	2.89e-01	2.42e+00	1.20e+03
L2 GDL	3.26e-01	2.87e+00	6.07e+00
L2 GDL MAE	1.06e+00	2.12e+00	2.21e+14
PINN MSE	1.23e+00	2.26e+00	1.19e+01
PINN RES	4.47e-01	3.88e+00	3.23e+00

(d) Median Error over Max True

	U-Net	Encoder	MultiScale
AR PINN MSE	6.42e-02	9.87e-03	1.31e-01
L2	4.40e-04	1.11e-03	1.08e+01
L2 GDL	6.24e-04	1.28e-03	5.38e-02
L2 GDL MAE	7.4e-04	8.82e-04	5.07e+05
PINN MSE	1.02e-03	1.14e-03	8.86e-02
PINN RES	5.51e-04	1.41e-03	3.11e-03

Table D.5 – Results on the layered space extrapolation data

(a) Average RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.33e+01	1.59e+01	1.86e+01
L2	1.30e+01	1.71e+01	1.20e+03
L2 GDL	1.43e+01	1.79e+01	1.17e+01
L2 GDL MAE	1.18e+01	1.71e+01	inf
PINN MSE	1.48e+01	1.75e+01	1.78e+01
PINN RES	9.30e+00	1.68e+01	1.00e+01

(b) Average SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	4.17e-01	6.32e-01	1.55e-01
L2	7.10e-01	5.79e-01	5.18e-01
L2 GDL	7.14e-01	6.66e-01	5.16e-01
L2 GDL MAE	7.45e-01	5.34e-01	2.99e-01
PINN MSE	6.99e-01	5.29e-01	5.10e-01
PINN RES	7.99e-01	5.25e-01	8.03e-01

(c) Average Absolute Error

	U-Net	Encoder	MultiScale
AR PINN MSE	6.76e+00	6.54e+00	1.37e+01
L2	5.37e+00	8.42e+00	8.13e+02
L2 GDL	5.61e+00	7.82e+00	6.94e+00
L2 GDL MAE	4.59e+00	8.8e+00	3.38e+14
PINN MSE	6.01e+00	9.13e+00	1.02e+01
PINN RES	3.52e+00	8.78e+00	3.93e+00

(d) Median Error over Max True

	U-Net	Encoder	MultiScale
AR PINN MSE	2.82e-02	5.55e-03	1.07e-01
L2	3.79e-03	1.36e-02	4.50e+00
L2 GDL	5.85e-03	4.25e-03	3.72e-02
L2 GDL MAE	2.23e-03	1.65e-02	1.64e+07
PINN MSE	6.25e-03	1.79e-02	4.65e-02
PINN RES	3.16e-03	1.91e-02	2.27e-03

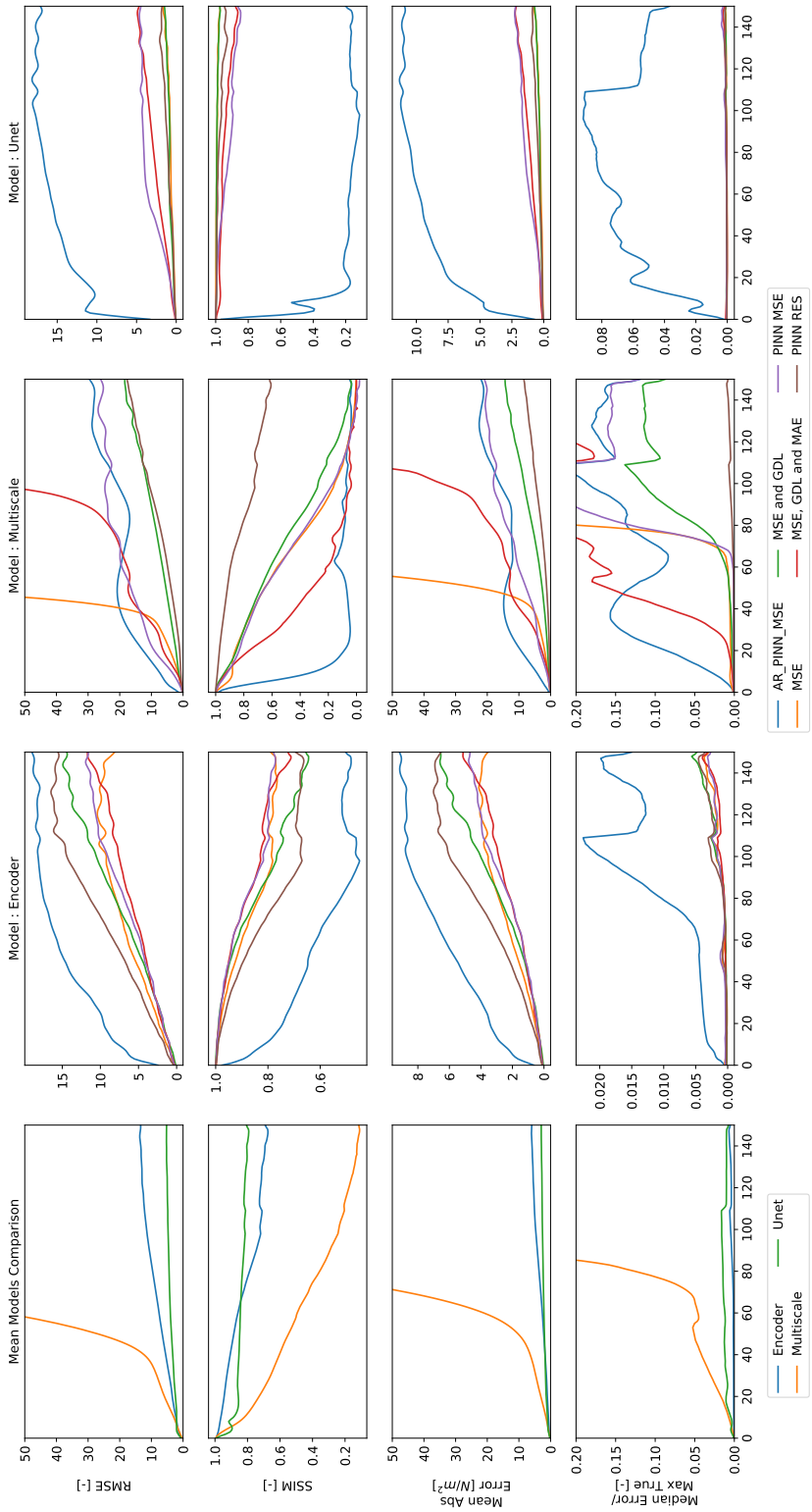


Figure D.5 – Relevant metrics for the layered training

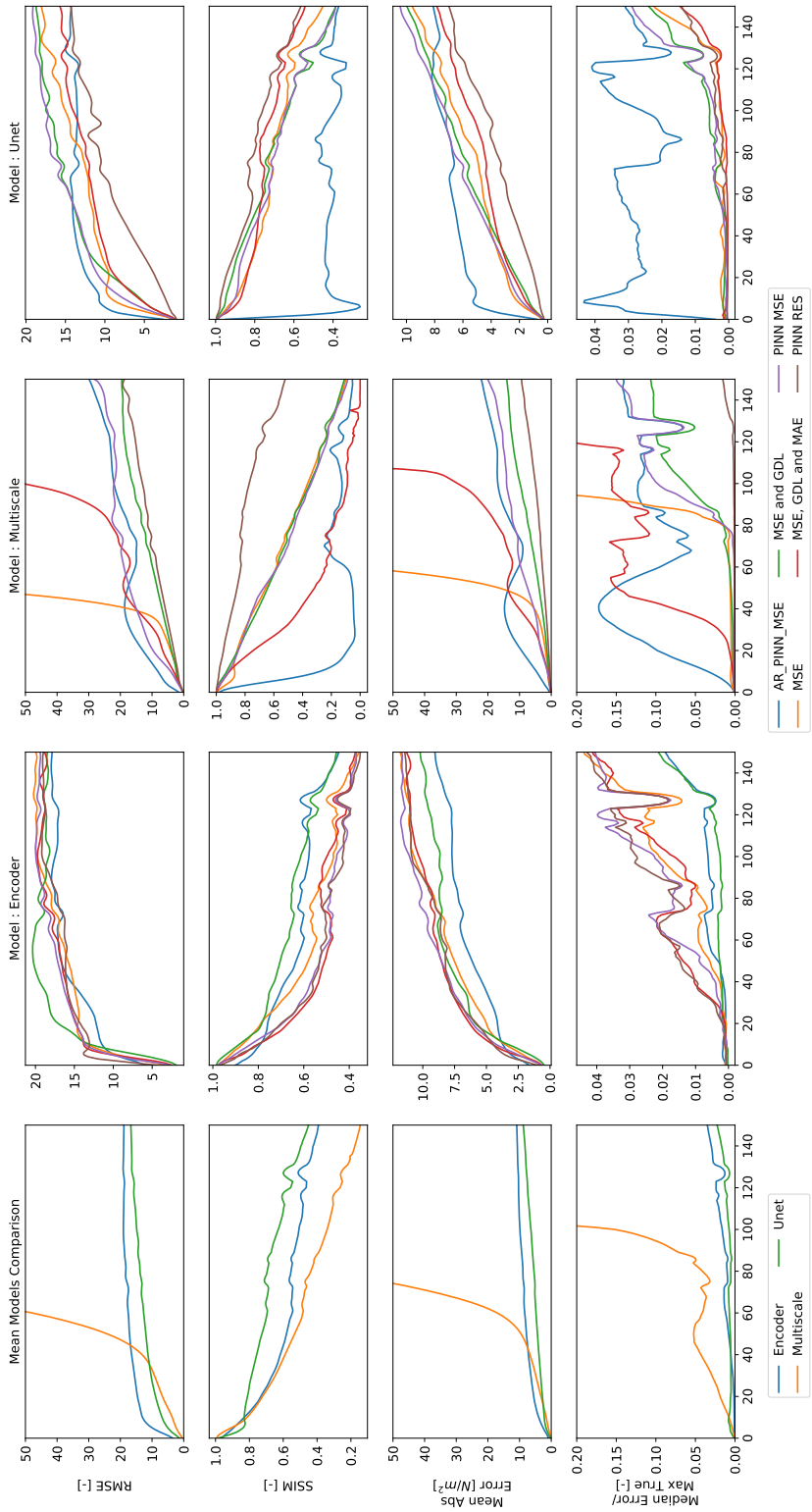


Figure D.6 – Relevant metrics for the layered domain with the space interpolation

Table D.6 – Results on the layered time extrapolation data

(a) Average RMSE

	U-Net	Encoder	MultiScale
AR PINN MSE	1.64e+01	1.68e+01	4.74e+01
L2	8.93e+00	1.3e+01	inf
L2 GDL	8.92e+00	1.45e+01	inf
L2 GDL MAE	9.16e+00	1.33e+01	inf
PINN MSE	1.02e+01	1.35e+01	inf
PINN RES	8.28e+00	1.44e+01	inf

(b) Average SSIM

	U-Net	Encoder	MultiScale
AR PINN MSE	1.62e-01	4.37e-01	6.31e-02
L2	6.67e-01	5.72e-01	2.66e-01
L2 GDL	6.67e-01	5.29e-01	2.17e-01
L2 GDL MAE	6.58e-01	5.63e-01	2.07e-01
PINN MSE	6.34e-01	5.65e-01	1.69e-01
PINN RES	6.68e-01	5.52e-01	4.66e-01

(c) Average Absolute Error

	U-Net	Encoder	MultiScale
AR PINN MSE	1.08e+01	8.54e+00	3.67e+01
L2	5.31e+00	7.28e+00	inf
L2 GDL	5.07e+00	8.25e+00	inf
L2 GDL MAE	5.2e+00	7.6e+00	inf
PINN MSE	5.71e+00	7.58e+00	inf
PINN RES	4.85e+00	7.72e+00	1.66e+23

(d) Median Error over Max True

	U-Net	Encoder	MultiScale
AR PINN MSE	6.88e-02	2.76e-02	3.01e-01
L2	2.28e-02	2.96e-02	8.29e+19
L2 GDL	2.18e-02	3.38e-02	1.09e+25
L2 GDL MAE	2.04e-02	3.12e-02	3.38e+26
PINN MSE	2.28e-02	3.06e-02	1.04e+25
PINN RES	2.20e-02	2.68e-02	4.77e+12

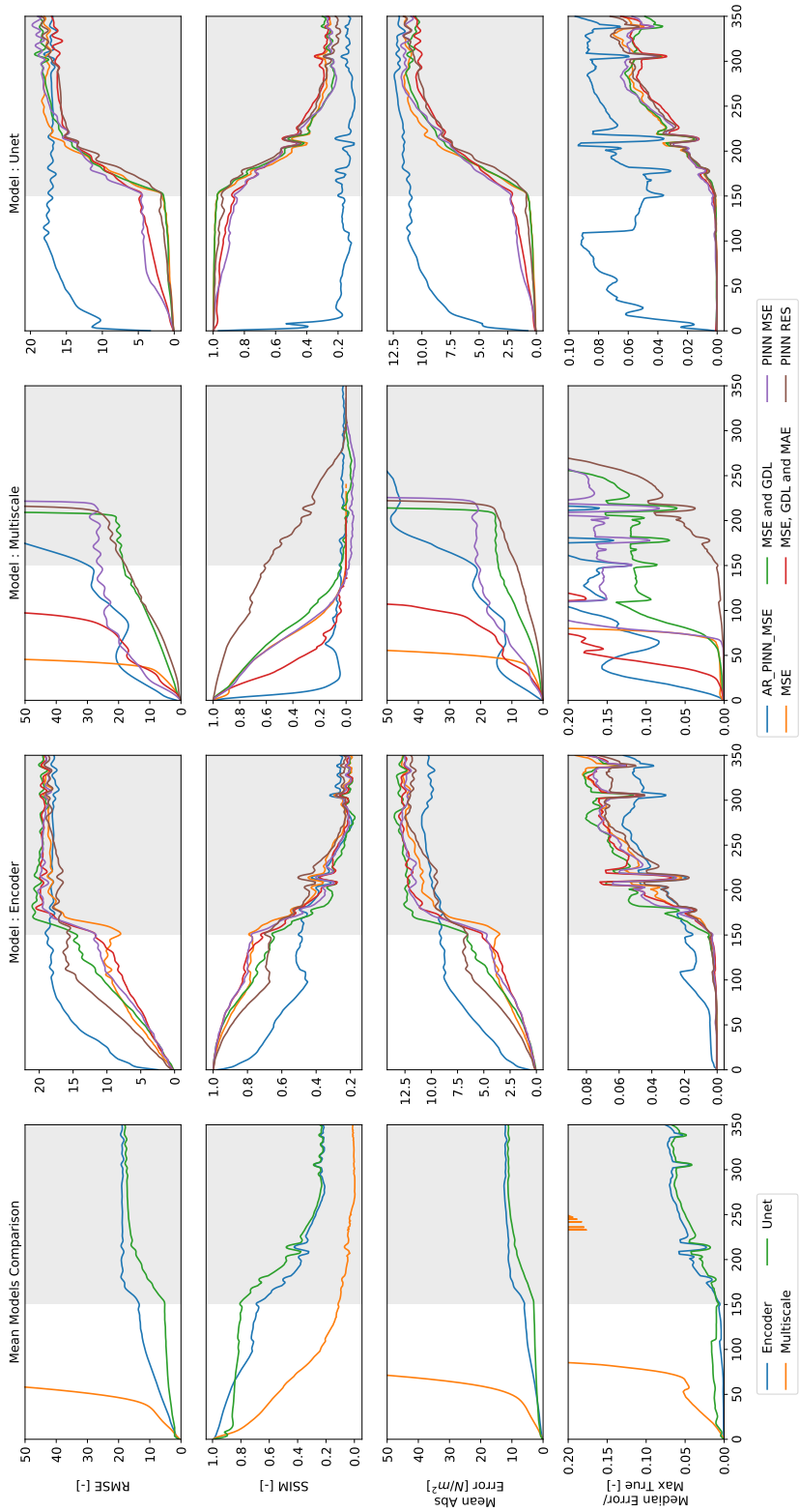


Figure D.7 – Relevant metrics for the layered domain with the time interpolation

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. *12th Symposium on Operating Systems Design and Implementation (16)*, 265–283.

Afanasiev, M., Boehm, C., van Driel, M., Krischer, L., Rietmann, M., May, D. A., Knepley, M. G., & Fichtner, A. (2019). Modular and flexible spectral-element waveform modelling in two and three dimensions. *Geophysical Journal International*, *216*(3), 1675–1692. https://doi.org/10.1093/gji/ggy469

Alguacil, A., Bauerheim, M., Jacob, M. C., & Moreau, S. (n.d.). Predicting the propagation of acoustic waves using deep convolutional neural networks. *Aiaa aviation 2020 forum*. https://doi.org/10.2514/6.2020-2513

Applied Technology Council. (1985). *Earthquake damage evaluation data for California* (tech. rep.). Seismic Safety Commission, Applied Technology Council (ATC). California.

Bengio, Y., & Lecun, Y. (1997). Convolutional networks for images, speech, and time-series.

bin Waheed, U., Haghighat, E., Alkhalifah, T., Song, C., & Hao, Q. (2021). Pinneik: eikonal solution using physics-informed neural networks.

Bjorck, J., Gomes, C. P., & Selman, B. (2018). Understanding batch normalization. *CoRR*, *abs/1806.02375*.

de Bézenac, E., Pajot, A., & Gallinari, P. (2017). Deep learning for physical processes: incorporating prior scientific knowledge. *CoRR*, *abs/1711.07970*.

Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., & Brox, T. (2015). Flownet: learning optical flow with convolutional networks. *CoRR*, *abs/1504.06852*.

Fotiadis, S., Pignatelli, E., Valencia, M. L., Cantwell, C. D., Storkey, A., & Bharath, A. A. (2020). Comparing recurrent and convolutional neural networks for predicting wave propagation. *CoRR*, *abs/2002.08981*.

Gardner, G. H. F., Gardner, L. W., & Gregory, A. R. (1974). Formation velocity and density—the diagnostic basics for stratigraphic traps. *GEOPHYSICS*, *39*(6), 770–780. https://doi.org/10.1190/1.1440465

Geneva, N., & Zabaras, N. (2019). Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Gulli, A., & Pal, S. (2017). *Deep learning with keras*. Packt Publishing Ltd.

Haghighat, E., & Juanes, R. (2021). Sciann: a keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, *373*, 113552. https://doi.org/10.1016/j.cma.2020.113552

Karimpouli, S., & Tahmasebi, P. (2020). Physics informed machine learning: seismic wave equation. *Geoscience Frontiers*, *11*(6), 1993–2001. https://doi.org/https://doi.org/10.1016/j.gsf.2020.07.007

Karniadakis, G., Kevrekidis, Y., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning, 1–19. https://doi.org/10.1038/s42254-021-00314-5

Kingma, D., & Ba, J. (2014). Adam: a method for stochastic optimization. *International Conference on Learning Representations*.

Langtangen, H. P., & Linge, S. (2018). *Finite difference computing with pdes: a modern software approach* (1st). Springer Publishing Company, Incorporated.

Larmat, C., Johnson, P., & Guyer, R. (2011). Time-reversal in seismology. *The Journal of the Acoustical Society of America*, *129*(4), 2531–2531. https://doi.org/10.1121/1.3588386

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–44. https://doi.org/10.1038/nature14539

Li, D., Xu, K., Harris, J. M., & Darve, E. (2020). Coupled time-lapse full waveform inversion for subsurface flow problems using intrusive automatic differentiation.

Liu, D., Lecampion, B., & Blum, T. (2020). Time-lapse reconstruction of the fracture front from diffracted waves arrivals in laboratory hydraulic fracture experiments*. *Geophysical Journal International*. https://doi.org/10.1093/gji/ggaa310

Loshchilov, I., & Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, *abs/1711.05101*.

Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2020). Deepxde: a deep learning library for solving differential equations.

Mathieu, M., Couprie, C., & LeCun, Y. (2016). Deep multi-scale video prediction beyond mean square error.

Mo, S., Zabaras, N., Shi, X., & Wu, J. (2020). Integration of adversarial autoencoders with residual dense convolutional networks for estimation of non-gaussian hydraulic conductivities. *Water Resources Research*, *56*(2). https://doi.org/10.1029/2019wr026082

Mo, S., Zhu, Y., Zabaras, N., Shi, X., & Wu, J. (2019). Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resources Research*, *55*(1), 703–728. https://doi.org/10.1029/2018WR023528

Moseley, B., Markham, A., & Nissen-Meyer, T. (2020). Solving the wave equation with physics-informed deep learning.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: an imperative style, high-performance deep learning library. *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: machine learning in python. *Journal of machine learning research*, *12*(Oct), 2825–2830.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. W. (2020). Learning mesh-based simulation with graph networks. *CoRR*, *abs/2010.03409*.

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045

Rao, C., Sun, H., & Liu, Y. (2020). Physics informed deep learning for computational elastodynamics without labeled data.

Ricker, N. (1953). The form and laws of propagation of seismic wavelets. *Geophysics*, *18*(1), 10–40. https://doi.org/10.1190/1.1437843

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: convolutional networks for biomedical image segmentation. *CoRR*, *abs/1505.04597*.

Ross, Z. E., Trugman, D. T., Hauksson, E., & Shearer, P. M. (2019). Searching for hidden earthquakes in southern california. *Science*, *364*(6442), 767–771. https://doi.org/10.1126/science.aaw6888

Shukla, K., Leoni, P. C. D., Blackshire, J., Sparkman, D., & Karniadakis, G. E. (2020). Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks.

Singh, S., & Shrivastava, A. (2019). Evalnorm: estimating batch normalization statistics for evaluation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 3632–3640. https://doi.org/10.1109/ICCV.2019.00373

Smith, J. D., Azizzadenesheli, K., & Ross, Z. E. (2020). Eikonet: solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 1–12. https://doi.org/10.1109/TGRS.2020.3039165

Song, C., Alkhalifah, T., & Waheed, U. B. (2021). Solving the frequency-domain acoustic vti wave equation using physics-informed neural networks. *Geophysical Journal International*, *225*(2), 846–859. https://doi.org/10.1093/gji/ggab010

Sorteberg, W. E., Garasto, S., Pouplin, A. S., Cantwell, C. D., & Bharath, A. A. (2018). Approximating the solution to wave propagation using deep neural networks.

Sun, L., Gao, H., Pan, S., & Wang, J.-X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, *361*, 112732. https://doi.org/10.1016/j.cma.2019.112732

Sun, L., & Wang, J.-X. (2020). Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters*, *10*(3), 161–169. https://doi.org/10.1016/j.taml.2020.01.031

Tartakovsky, A., Marrero, C., Perdikaris, P., Tartakovsky, G., & Barajas-Solano, D. (2020). Physics-informed deep neural networks for learning parameters and constitutive relationships

in subsurface flow problems. *Water Resources Research*, *56*, e2019WR026731. https://doi.org/10.1029/2019WR026731

Thuerey, N., Weißenow, K., Prantl, L., & Hu, X. (2020). Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, *58*(1), 25–36. https://doi.org/10.2514/1.J058291

van der Meer, R., Oosterlee, C., & Borovykh, A. (2021). Optimally weighted loss functions for solving pdes with neural networks.

Wang, Y. (2014). The Ricker wavelet and the Lambert W function. *Geophysical Journal International*, *200*(1), 111–115. https://doi.org/10.1093/gji/ggu384

Wang, Z., Bovik, A., Sheikh, H., & Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), 600–612. https://doi.org/10.1109/TIP.2003.819861

Weiqiang, Z., Sheng, Y., & Sun, Y. (2017). Wave-dynamics simulation using deep neural networks. https://doi.org/10.13140/RG.2.2.21994.77764

Yilmaz, O. (2001). *Seismic data analysis* (2 ed, Vol. 10). Society Of Exploration Geophysicists. https://doi.org/10.1190/1.9781560801580

Zhu, W., Xu, K., Darve, E., & Beroza, G. C. (2021). A general approach to seismic inversion with automatic differentiation. *Computers Geosciences*, *151*, 104751. https://doi.org/10.1016/j.cageo.2021.104751

Zhu, Y., & Zabaras, N. (2018). Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, *366*, 415–447. https://doi.org/10.1016/j.jcp.2018.04.018

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., & Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, *394*, 56–81. https://doi.org/10.1016/j.jcp.2019.05.024