Deep Learning for Inverting Borehole Resistivity Measurements

Jon Ander Rivera González

Supervised by *David Pardo* and *Elisabete Alberdi*

November 2022

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

basque center for applied mathematics

Deep Learning for Inverting Borehole Resistivity Measurements

Jon Ander Rivera González

Supervised by *David Pardo* and *Elisabete Alberdi*

November 2022

This dissertation has been possible with the support of the University of the Basque Country (UPV/EHU) grant No. PIF18/017; the BCAM "Severo Ochoa" accreditation of excellence (SEV-2017-0718); the Basque Government through the BERC 2018-2021 program; and the Consolidated Research Group MATHMODE (IT1294-19;IT1456-22) given by the Department of Education.

Acknowledgements

The last four years of my life have been an adventure of learning and personal growth. In this journey, I have been fortunate to have David Pardo as my supervisor. The day I met him I was surprised by the naturalness with which he treated me. From then on, I got to know the rest of his multiple virtues: work discipline (always available for any questions or queries), enthusiasm for research (always looking for new challenges and new problems to solve), and intuition–the one that impresses me the most. I will be eternally grateful to him and he knows that, wherever he is, he will always have a friend here.

I would also like to thank my other supervisor Elisabete Alberdi for all the work she has done to help me. I would like to thank her for the trust she has placed in me when making me part of her projects and for all the advice she has given me during these four years. Finally, she has taught me to manage within the University and she has given me the opportunity to teach some classes and see how things look from the other side of the classroom.

I am deeply grateful to Mostafa Shahriari for all his help. He is a lovely person. From the first day I came to the group, he welcomed me and had the patience to help me day after day. He taught me everything he knew and helped me build the foundation of what today is my Dissertation. I also want to thank him for the opportunity he gave me to go to Austria to collaborate with him. I learned a lot from that experience. I expect to continue seeing each other in the coming years.

I would like to express my gratitude to Javier Omella for everything he has done for me. I have shared many hours of work with him and he taught me new things. He was always available to help and together we managed to get the job done. I have seen few experts advanced programmers like him and I thank him for transmitting all that knowledge to me.

I wish to thank all my colleagues from the MATHMODE group and BCAM. Especially, I would like to thank Judit Muñoz-Matute and Ana Fernandez-Navamuel. The first one for helping me with bureaucracy questions; the second one for sharing with me her enriching experiences while developing our Dissertations.

Finally, I want to thank my family and friends for their support. In particular, I want to offer my eternal gratitude to the three people that have been with me all this way long. First, to my parents Mari Carmen and Michel. Thank you for all your unconditional support and love, for teaching me the values that have made me who I am, and for encouraging me to pursue a Ph.D degree. Last but not least, to my partner Amaia. She has consistently supported me when I needed it: she was the serenity when I was stressed and the encouragement when I was frustrated. Thank you for trusting me and for never ceasing to remind me that I was capable of doing it. *Zaila bada ez da ezinezkoa, zaila bada badago lortzea.*

Abstract

The Earth's subsurface is formed by different materials, mainly porous rocks possibly containing minerals and filled with salty water and/or hydrocarbons. The formations that these materials create are often irregular, appearing geometrically abrupt forms with different properties that are mixed within the same layer.

One of the main objectives in geophysics is to determine the petrophysical properties of the Earth's subsurface. In this way, companies can discover hydrocarbon reservoirs and maximize the production, and determine optimal locations for hydrogen storage or CO_2-sequestration. To achieve these goals, companies often record electromagnetic measurements using Logging While Drilling (LWD) instruments, which are able to record data while drilling. The recorded data is processed to produce a map of the Earth's subsurface. Based on the reconstructed Earth model, the operator adjusts the well trajectory in real-time to further explore exploitation targets, including oil and gas reservoirs, and to maximize the posterior productivity of the available reserves. This real-time adjustment technique is called *geosteering*.

Nowadays, geosteering plays an essential role in geophysics. However, it requires the capability of solving inverse problems in real time. This is challenging since inverse problems are often ill-posed.

There exist multiple traditional methods to solve inverse problems, mainly, gradient-based or statistics-based methods. However, these methods have severe limitations. In particular, they often need to compute the forward problem hundreds of times for each set of measurements, which is computationally expensive in three-dimensional (3D) problems.

To overcome these limitations, we propose the use of Deep Learning (DL) techniques to solve inverse problems. Although the training stage of a Deep Neural Network (DNN) may be time-consuming, after the network is properly trained, it can forecast the solution in a fraction of a second, facilitating real-time geosteering operations. In the first part of this dissertation, we investigate appropriate loss functions to train a DNN when dealing with an inverse problem.

Additionally, to properly train a DNN that approximates the inverse solution, we require a large dataset containing the solution of the forward problem for many different Earth models. To create such dataset, we need to solve a Partial Differential Equation (PDE) thousands of times. Building a dataset may be time-consuming, especially for two and three-dimensional problems since solving

PDEs using traditional methods, such as the Finite Element Method (FEM), is computationally expensive. Thus, we want to reduce the computational cost of building the database needed to train the DNN. For this, we propose the use of refined Isogeometric Analysis (rIGA) methods.

In addition, we explore the possibility of using DL techniques to solve PDEs, which is the main computational bottleneck when solving inverse problems. Our main goal is to develop a fast forward simulator for solving parametric PDEs. As a first step, in this dissertation we analyze the quadrature problems that appear while solving PDEs using DNNs and propose different integration methods to overcome these limitations.

Resumen

El subsuelo terrestre está formado por diferentes materiales, principalmente por rocas porosas que posiblemente contienen minerales y están rellenas de agua salada y/o hidrocarburos. Por lo general, las formaciones que crean estos materiales son irregulares y con materiales de diferentes propiedades mezclados en el mismo estrato.

Uno de los principales objetivos en geofísica es determinar las propiedades petrofísicas del subsuelo de la Tierra. De este modo, las compañías pueden determinar la localización de las reservas de hidrocarburos para maximizar su producción o descubrir localizaciones óptimas para el almacenamiento de hidrógeno o el depósito de CO_2. Para este propósito, las compañías registran mediciones electromagnéticas utilizando herramientas de Medición Durante Perforación (MDP), las cuales son capaces de recabar datos mientras se lleva a cabo el proceso de prospección. Los datos obtenidos se procesan para producir un mapa del subsuelo de la Tierra. Basándose en el mapa generado, el operador ajusta en tiempo real la trayectoria de la herramienta de prospección para seguir explorando objetivos de explotación, incluidos los yacimientos de petróleo y gas, y maximizar la posterior productividad de las reservas disponibles. Esta técnica de ajuste en tiempo real se denomina *geo-navegación*.

Hoy en día, la geo-navegación desempeña un papel esencial en geofísica. Sin embargo, requiere la resolución de problemas inversos en tiempo real. Esto supone un reto, ya que los problemas inversos suelen estar mal planteados.

Existen múltiples métodos tradicionales para resolver los problemas inversos, principalmente, los métodos basados en el gradiente o en la estadística. Sin embargo, estos métodos tienen graves limitaciones. En particular, a menudo necesitan calcular el problema inverso cientos de veces para cada conjunto de mediciones, lo que es computacionalmente caro en problemas tridimensionales (3D).

Para superar estas limitaciones, proponemos el uso de técnicas de Aprendizaje Profundo (AP) para resolver los problemas inversos. Aunque la etapa de entrenamiento de una Red Neuronal Profunda (RNP) puede requerir mucho tiempo, una vez que la red está correctamente entrenada puede predecir la solución en una fracción de segundo, facilitando las operaciones de geo-navegación en tiempo real. En la primera parte de esta tesis, investigamos las funciones de pérdida apropiadas para entrenar una RNP cuando se trata de un problema inverso.

Además, para entrenar adecuadamente una RNP que se aproxime a la solución inversa, necesitamos un gran conjunto de datos que contenga la solución del problema directo para muchos modelos terrestres diferentes. Para crear dicho conjunto de datos, necesitamos resolver una Ecuación en Derivadas Parciales (EDPs) miles de veces. La creación de un conjunto de datos puede llevar mucho tiempo, especialmente para los problemas bidimensionales y tridimensionales, ya que la resolución de la EDPs mediante métodos tradicionales, como el Método de Elementos Finitos (MEF), es computacionalmente caro. Por lo tanto, queremos reducir el coste computacional de la construcción de la base de datos necesaria para entrenar la RNP. Para ello, proponemos el uso de métodos de Análisis Isogeométrico refinado (AIGr).

Además, exploramos la posibilidad de utilizar técnicas de AP para resolver EDPs, que es la limitación computacional principal al resolver problemas inversos. Nuestro objetivo principal es desarrollar un simulador rápido para resolver EDPs paramétricas. Como primer paso, en esta tesis analizamos los problemas de cuadratura que aparecen al resolver EDPs utilizando RNPs y proponemos diferentes métodos de integración para superar estas limitaciones.

Contents

Acronyms

DL	Deep Learning
AI	Artificial Intelligence
PDE	Partial Differential Equation
ODE	Ordinary Differential Equation
FEM	Finite Element Method
FDM	Finite Difference Method
IGA	Isogeometric Analysis
EM	electromagnetic
BC	Boundary Condition
DNN	Deep Neural Network
RNN	Residual Neural Network
NN	Neural Network
PINN	Physics-Informed Neural Network
VPINN	Variational Physics-Informed Neural Networks
DRM	Deep Ritz Method
DGM	Deep Galerkin Method
DLS	Deep Least Square
SGD	Stochastic Gradient Descent
LS	Least Square
LWD	Logging While Drilling

Acronyms

GPU	Graphics Processing Unit
CPU	Central Processing Unit
rIGA	refined Isogeometric Analysis
CAD	Computer-Aided Design

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Literature Review

The Earth's subsurface is formed by different materials, mainly porous rocks containing minerals and filled with salty water and/or hydrocarbons. The formations that these materials create are often irregular, appearing abrupt forms with peaks or breaks. Furthermore, each of the several layers that compose the Earth is composed of various materials with different material properties. Figure 1.1 shows an example of a laminar subsurface formation.

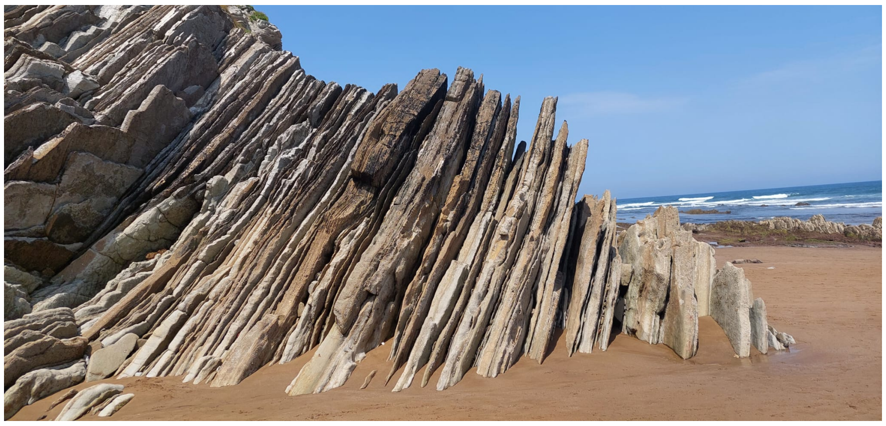


Figure 1.1: Earth subsurface formation with different layers. Photo taken in Sopela (Biscay, Spain).

Several fields demand a map of the subsurface in order to carry out their activities, needed for: (a) minimize earthquake-induced damage, (b) enhance the production of geothermal energy, (c) store different materials such as hydrogen in subsurface reservoirs, and (d) maximize hydrocarbon recovery.

In this last application, companies often record electromagnetic (EM) measurements using a Logging While Drilling (LWD) instrument. These tools incorporate different transmitters that generate an EM field. In the same way, several receivers are placed along the tool in order to receive the emitted wave after rebounding in the surroundings of the borehole. Depending on the materials and/or the formation of the surroundings, the received waves exhibit different properties. Figure 1.2 shows an example of a conventional LWD instrument.

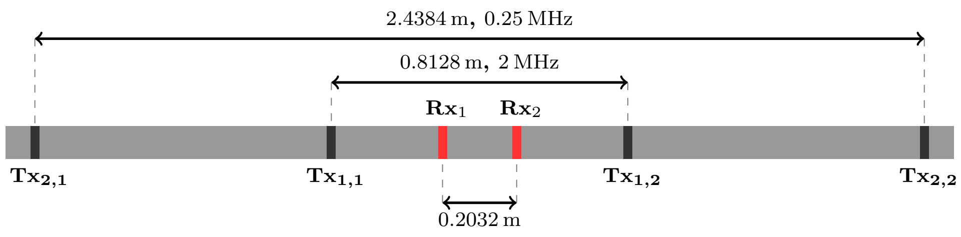


Figure 1.2: Example of a conventional LWD instrument. This tool is equipped with a pair of receivers (red) and two pairs of transmitters (black).

These tools have gained importance in the oil and gas industry in the last decades due to their capability to record logging data during drilling. The recorded data is processed to produce a map of the Earth's subsurface nearby the well. Based on the reconstructed Earth model, the operator adjusts the well-trajectory in real-time to further explore exploitation targets, including oil and gas reservoirs, and to maximize the posterior productivity of the available reserves. This real-time navigation technique is called *geosteering*. As a consequence of the tremendous productivity increase achieved with this technique, nowadays geosteering plays an essential role in the oil and gas industry [40].

The main difficulty one faces when dealing with geosteering problems is to obtain a map of the Earth's subsurface. We must solve the following inverse problem: given the measurements $\mathcal{M}$ recorded by the tool and the well trajectory $\mathcal{T}$, we want to obtain the subsurface properties ρ. In contrast, the forward problem is the one that given the subsurface properties ρ and the well trajectory $\mathcal{T}$, it produces the measurements $\mathcal{M}$ recorded by the tool. Figure 1.3 presents a schematic description of the forward and inverse problems.

Unfortunately, traditional inversion methods have severe limitations, which force geophysicists to continuously look for new solutions to this problem (see, e.g., [28, 41, 49, 64, 99, 128, 131, 156]). In particular, inverse problems are not well-defined, that is, there may exist multiple outputs for a given input [141, 145]. Gradient-based methods require simulating the forward problem dozens of times for each set of measurements. Moreover, these methods also estimate the derivatives of the measurements with respect to the inversion variables, which is often challenging and time consuming [141]. To alleviate the high computational costs associated with these inversion methods, simplified 1.5-dimensional (1.5D) methods are common (see, e.g., [64, 99, 133]). For the inversion of borehole resistivity measurements, an alternative is to apply statistics-based methods [53,

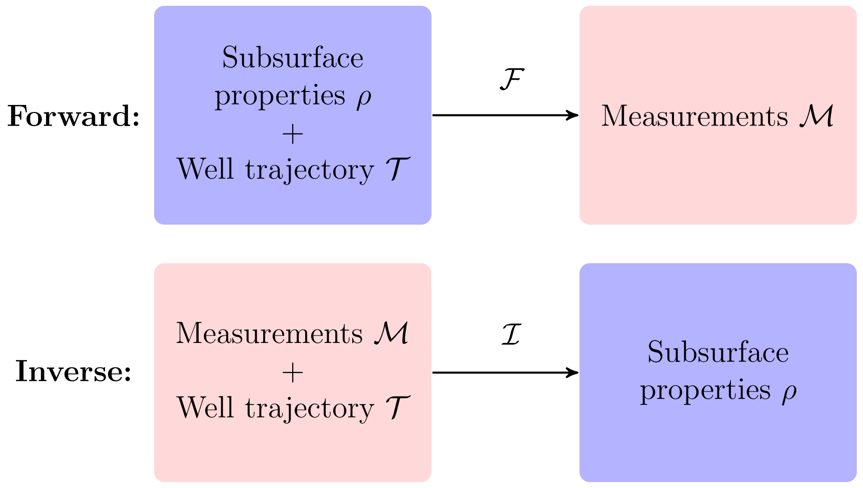

Forward:
Subsurface properties ρ + Well trajectory $\mathcal{T}$
$\mathcal{F}$
Measurements $\mathcal{M}$
Inverse:
Measurements $\mathcal{M}$ + Well trajectory $\mathcal{T}$
$\mathcal{I}$
Subsurface properties ρ

Figure 1.3: Schematic description of forward and inverse problems.

85, 147]. The statistical methods also perform forward simulations hundreds of times for each set of measurements. Both gradient and statistics-based methods *only evaluate* the inverse operator. Thus, the entire inversion process is repeated at each new logging position.

Deep Learning (DL) techniques seem apropiate to overcome the limitations of traditional methods while solving inversion problems. The large amount of research articles and industrial applications of DL algorithms in different areas – computer vision [82], speech recognition [4, 5, 154], biometrics [16], self-driving cars [54, 114], and healthcare [43, 108] to mention a few – are exponents of their high performance and capability to solve all kind of problems.

In addition, in recent years there have been significant advances in the field of DL, with the appearance of Residual Neural Networks (RNNs) [59], which prevent gradient degeneration during the training stage, and Encoder-Decoder (sequence-to-sequence) Deep Neural Networks (DNNs), which have improved the DL work capability in computer vision applications [10]. Due to the high demand from industry to use DNNs, dedicated libraries and packages such as Tensorflow [82], Keras [29], and Pytorch [103] have been developed. These libraries facilitate the use of DNNs across different industrial applications [39, 66, 83, 117, 132, 144, 157]. All these advances make DNNs one of the most powerful and fast-growing Artificial Intelligence (AI) tools presently. **The first main contribution of this dissertation is to design a fast inversion method using DL techniques to solve borehole measurement problems that allows the application of geosteering techniques.**

However, DNNs also face important challenges when applied to the inversion of borehole resistivity problems. In particular, the training stage can be time-consuming. However, this is an *offline* cost incurred during the training stage. Then, after the network is properly trained, it can forecast a solution in a fraction of a second [131]. This feature allows real-time inversion, which facilitates geosteering operations. Another limitation of DNN is that they require a large dataset (also known as *ground truth*). In our case, it consists of the solution of the forward problem for different Earth models [60, 130, 131].

To generate the database for DL inversion, we must solve the forward problem. Our forward problem – simulation of borehole measurements – is governed by Partial Differential Equations (PDEs). In our case, we consider resistivity measurements governed by a set of four time-dependent first-order PDE named Maxwell's equations [45]. Here, knowing some electrical properties (i.e., electrical conductivities of the subsurface materials), we can obtain the corresponding electric and magnetic fields (i.e., recorded measurements).

We solve the forward problem using numerical simulation methods such as the Finite Element Method (FEM) [6, 11, 58, 69, 98, 115, 133] or the Finite Difference Method (FDM) [35, 36, 77, 140]. Moreover, we need to optimally sample the parameter space describing relevant Earth models. This process may be time-consuming, especially for two and three-dimensional problems. In those cases, it is common to reduce the Earth model dimensionality to two or one spatial dimensions using a Fourier or a Hankel transform. These transformations lead to the so-called 2.5D [3, 48, 92, 101, 135] and 1.5D [12, 99, 129] formulations, respectively. In particular, 1.5D simulations are inaccurate when dealing with geological faults.

Galerkin methods are effective for simulating well-logging problems (see, e.g., [24, 27, 84, 95, 97, 120, 146]). Isogeometric Analysis (IGA), introduced by [61], is a widely used Galerkin method for solving PDEs. IGA has been successfully employed in various EM [20, 21, 93, 137, 138] and geotechnical [56, 134] applications. IGA uses spline basis functions introduced in Computer-Aided Design (CAD) as basis functions of FEM. These basis functions exhibit high continuity (up to C^{p-1}, being p the polynomial order of spline bases) across the element interfaces.

When comparing IGA and FEM, the former provides smoother solutions for wave propagation problems with a lower number of unknowns [31, 61]. However, in contrast to the minimal interconnection of elements in FEM, high-continuity IGA discretizations strengthen the interconnection between elements, leading to an increase of the cost of matrix LU factorization per degree of freedom when using sparse direct solvers [30]. In order to avoid this degradation and also benefit from the recursive partitioning capability of multifrontal direct solvers, [47] developed a new method called refined Isogeometric Analysis (rIGA). This discretization technique conserves desirable properties of high-continuity IGA discretizations, while it partitions the computational domain into blocks of macroelements weakly interconnected by low-continuity separators. As a result, the computational cost required for performing LU factorization decreases. The applicability of the rIGA framework to general EM problems was studied by [46]. Compared to high-continuity IGA, rIGA produces solutions of EM problems up to $\mathcal{O}(p^2)$ faster on large domains and close to $\mathcal{O}(p)$ faster on small domains. rIGA also improves the approximation errors with respect to IGA since the continuity reduction of basis functions increases the number of degrees of freedom [31] and enriches the Galerkin space. Thus, **as another contribution of this dissertation, we propose the use of rIGA discretizations to generate databases for DL inversion of 2.5D geosteering EM measurements.**

Apart from the traditional methods, another option for solving PDEs is the use of DL techniques. In the last years, DL algorithms have become popular for solving PDEs – see, e.g., [15, 18, 81, 104, 118, 121, 123, 152]. DL techniques present several advantages and limitations with respect to traditional PDE solvers based on FEM [58], FDM [77], or IGA [94]. Among the advantages of DL, we encounter the nonnecessity of generating a grid. In general, DL uses a dataset in which each datum is independent from others. In contrast, in the linear system that produces the FEM, there exists a connectivity between the nodes of the mesh. In addition, DL allows the parallelization into Graphics Processing Units (GPUs) for fast computations. Furthermore, DL provides the possibility of solving certain problems that cannot be solved via traditional methods, like high-dimensional PDEs [42, 57], some fractional PDEs [9, 96], and multiple nonlinear PDEs [62, 113]. Moreover, using DL techniques opens the door to rapidly solve parametric PDEs.

However, DL also presents limitations when solving PDEs. For example, in [148] they show that Fully-Connected Neural Networks suffer from spectral bias [111] and exhibit different convergence rates for each loss component. In addition, the convergence of the method is often assumed, under reasonable hypothesis (see, e.g., [88]), since it cannot be rigorously guaranteed due to the non-convexity of the loss function. Another notorious problem of DL methods for solving PDEs is due to quadrature errors. In traditional mesh-based methods, such as FEM, we first select an approximating solution space and then compute the integrals over each element required to produce the stiffness matrix. With DL, we first set a quadrature rule and then construct the approximated function. Due to this, there is no proper quadrature rule for DNNs, as remarked in [71].

The most common method used in DL to approximate definite integrals is based on Monte Carlo integration. This family of methods compute definite integrals using randomly sampled points from a given distribution. Monte Carlo integration is suitable for high-dimensional integrals. However, for low-dimensional integrals (1D, 2D, and 3D), convergence is slow in terms of the number of integration points in comparison to other quadrature rules. This produces elevated computational costs. Examples of existing DL works that follow a Monte Carlo approach are [149] using a Deep Ritz Method (DRM), [139] using a Deep Galerkin Method (DGM), and [86], where they employ the so-called Physics-Informed Neural Networks (PINNs) [112] that can be interpreted as a collocation method or as a variational method using Dirac delta test functions. From the practical point of view, at each iteration of the Stochastic Gradient Descent (SGD) [63], one considers a mini-batch of randomly selected points to discretize the domain.

Another existing method to compute integrals in DNNs is the so-called *automatic integration* [78]. In this method, the author approximates the integrand by its high order Taylor series expansion around a given point within the integration domain. Then, the integrals are computed analytically. The derivatives needed in the Taylor series expansion are computed via automatic differentiation (a.k.a *autodiff*) [52]. Since the information of the derivatives is local, overfitting may easily occur.

Another alternative quadrature rule is to use adaptive integration methods. Examples where authors have selected these methods are the Deep Least Square (DLS) method [23], where they employ an adaptive mid-point quadrature rule using local error indicators. These indicators are based on the value of the residual at randomly selected points, and the resulting quadrature error is unclear.

One can also use Gauss-type quadrature rules to evaluate integrals, as they do in Variational Physics-Informed Neural Networkss (VPINNs) [71]. One limitation of this method is the impossibility of selecting *a priori* an adequate quadrature order because properties of the Neural Network approximation are unknown. In addition, the use of fixed quadrature rules increases the chances of performing overfitting. Therefore, **as the third and last main contribution of this dissertation, we analyze the problems associated with quadrature rules in DL methods when solving PDEs, and we propose several alternatives to overcome the quadrature problems.**

1.2 Outline

The remainder of the dissertation is organized as follows. In Chapter 2 we study adequate loss functions to solve inverse problems using DL and test them with real examples. Chapter 3 proposes the use of rIGA to generate a database for solving borehole inverse problems using DL. Chapter 4 analyzes the quadrature problems that may arise when using inadequate integration methods in DL for solving PDEs, and propose suitable alternatives. Chapter 5 is devoted to the conclusions and future works and Chapter 6 provides a list of main achievements.

2 Solving Inverse Problems using Deep Learning

In this Section, we solve a borehole inversion problem using Deep Learning (DL) techniques. We denote as inverse problem the one where having some measurements recorded using a Logging While Drilling (LWD) instrument, we want to recover a map of the Earth's subsurface. In opposition, we have the forward problem, where given the properties of the Earth's subsurface, we want to obtain the set of recorded measurements. Often, the forward problem is *well-posed* in the sense of Hadamard [55].

Definition 2.1. Let X and Y be two normed spaces and $A : X \to Y$ a mapping. The problem defined by the equation $Ax = y$ is said to be *well-posed* if the following conditions are satisfied:

- The solution of the problem exists: $\forall y \in Y \Rightarrow \exists x \in X : Ax = y$.
- The solution is unique: $\forall y \in Y \Rightarrow \exists! x \in X : Ax = y$.
- The solution is stable: $\forall (x_n) \subset X$ with $Ax_n \to Ax$ for $n \to \infty$, it follows that $x_n \to x$ for $n \to \infty$.

The problems that do not satisfy one of the above properties are called *ill-posed.* Inverse problems are usually *ill-posed.* As we will see in this Section, borehole inverse problems lack the uniqueness of the solution, which makes solving those problems using DL a difficult task.

2.1 Problem Formulation

2.1.1 Forward Problem

We fix the measurement acquisition system $\tilde{\mathbf{s}}$. Then, for a well trajectory $\tilde{\mathbf{t}}$, and an Earth model $\tilde{\mathbf{p}}$, the forward problem consists of finding the corresponding borehole resistivity measurements $\tilde{\mathbf{m}}$. We denote by $\tilde{\mathcal{F}}$ the associated forward function. That is:

$$\tilde{\mathcal{F}}(\tilde{\mathbf{t}}, \tilde{\mathbf{p}}) = \tilde{\mathbf{m}}, \qquad \text{where } \tilde{\mathbf{t}} \in \tilde{\mathbb{T}}, \tilde{\mathbf{p}} \in \tilde{\mathbb{P}}, \tilde{\mathbf{m}} \in \tilde{\mathbb{M}}. \tag{2.1}$$

In the above, we omit for convenience the explicit dependence of the function $\tilde{\mathcal{F}}$ upon the fixed input variable $\tilde{\mathbf{s}}$. $\tilde{\mathbb{P}} = \{\tilde{\mathbf{p}} = \tilde{\mathbf{p}}(x, y, z) \in \mathbb{R}^{3x3} : \forall(x, y, z) \in \Omega \subset \mathbb{R}^3\}$ – the set of all possible resistivity tensors – and $\tilde{\mathbb{M}} = \{\tilde{m} \in \mathbb{R}^m$, being m the number of measurements $\}$ – the set of all possible measurements – are normed vector spaces equipped with norms $||\cdot||_{\tilde{\mathbb{P}}}$ and $||\cdot||_{\tilde{\mathbb{M}}}$, respectively. $\tilde{\mathbb{T}} = \{\mathbf{t} = \mathbf{t}(s) : \mathbf{t}(s) \in \Omega \ \forall s \in (a, b) \subset \mathbb{R}\}$ – the set of all possible logging trajectories – is also a vector space. Function $\tilde{\mathcal{F}}$ consists of a boundary value problem governed by Maxwell's equations (see [133] for details).

2.1.2 Inverse Problem

In the inversion of borehole resistivity measurements, the objective is to determine the subsurface properties $\tilde{\mathbf{p}}$ corresponding to a set of measurements $\tilde{\mathbf{m}}$ recorded over a given trajectory $\tilde{\mathbf{t}}$. Again, the measurement acquisition system $\tilde{\mathbf{s}}$ is fixed. We denote that inverse operator as $\tilde{\mathcal{I}}$. Mathematically, we have:

$$\tilde{\mathcal{I}}(\tilde{\mathbf{t}}, \tilde{\mathbf{m}}) = \tilde{\mathbf{p}}, \qquad \text{where } \tilde{\mathbf{t}} \in \tilde{\mathbb{T}}, \tilde{\mathbf{m}} \in \tilde{\mathbb{M}},\ \tilde{\mathbf{p}} \in \tilde{\mathbb{P}}. \tag{2.2}$$

Again, we omit for convenience the explicit dependence of function $\tilde{\mathcal{I}}$ upon input variable $\tilde{\mathbf{s}}$. The governing physical equation of operator $\tilde{\mathcal{I}}$ is unknown. However, we know that a given input may have multiple associated outputs. Thus, such inverse operator is not well-defined.

2.1.3 Parameterization

We select a finite dimensional subspace of $\tilde{\mathbb{T}}$ parameterized with n_t real-valued numbers. The corresponding vector representation of an element from that subspace is $\mathbf{t} \in \mathbb{R}^{n_t}$. We similarly parameterize a finite dimensional subspace of $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{M}}$ with n_p and n_m real-valued numbers, respectively. The corresponding vector representations of an element from those subspaces are denoted as $\mathbf{p} \in \mathbb{R}^{n_p}$ and $\mathbf{m} \in \mathbb{R}^{n_m}$, respectively.

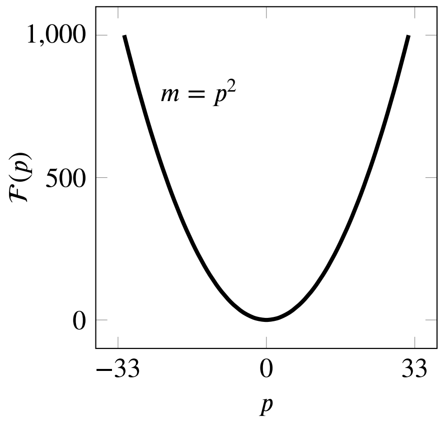
The span of vector representations $\mathbf{p}$ and $\mathbf{m}$ constitute two subspaces of $\mathbb{R}^{n_p}$ and $\mathbb{R}^{n_m}$ with norms $||\cdot||_{\mathbb{P}}$ and $||\cdot||_{\mathbb{M}}$, respectively. Ideally, these norms should be inherited from those associated with the original infinite dimensional spaces. However, this is often a challenging task and an open area of research. We directly employ some existing (typically l_1 or l_2) finite dimensional norms.

The function $\mathcal{F}$ associates a pair $(\mathbf{t},\ \mathbf{p})$ (vector representations of $(\tilde{\mathbf{t}},\tilde{\mathbf{p}})$) with $\mathbf{m}$ (vector representation of $\tilde{\mathbf{m}}$) such that $\mathcal{F}(\mathbf{t}, \mathbf{p}) = \mathbf{m}$. We employ a similar notation for its inverse $\mathcal{I}$ acting on vector representations.

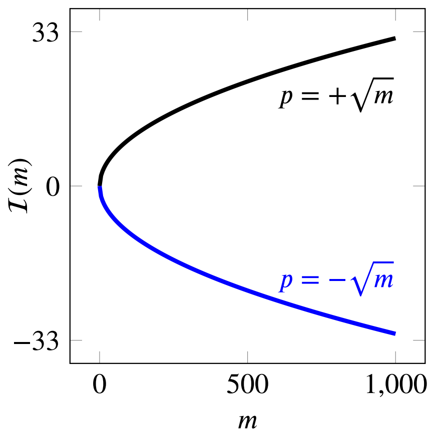
To provide context and guidance for future developments, we introduce simple examples that illustrate some of the shortcomings of the standard techniques when applied to these problems, and we explain how we seek to overcome the associated challenges. The first problem seeks to predict the inverse of squaring a number. The second example focuses on geosteering applications.

2.1.4 Example A: Model Problem with Known Analytical Solution

We select $n_t = 0$, $n_p = n_m = 1$. The forward function is given by $\mathcal{F}(p) = p^2$, while the inverse problem has two solutions (branches): $\mathcal{I}(m) = +\sqrt{m}$, and $\mathcal{I}(m) = -\sqrt{m}$, as described in Figure 2.1.



(a) Forward function



(b) Inverse operator with two branches

Figure 2.1: Model problem with known analytical solution.

This simple example contains a key feature exhibited by most inverse problems: it has multiple solutions. Thus, it is useful to illustrate the behaviour of Deep Neural Networks (DNNs) when considering different loss functions. Results are enlightening and, as we show below, they provide clear guidelines to construct proper loss functions for approximating inverse problems.

2.1.5 Example B: Inversion of Borehole Resistivity Measurements

In geosteering applications, multiple oil and service companies perform inversion assuming a piecewise 1D layered model of the Earth. In this case, there exist semi-analytic methods that can simulate the forward problem in a fraction of a second. Herein, we use the same approach. Thus, the evaluation of $\mathcal{F}$ is performed with a 1.5D semi-analytic code (see [80, 133]). As a result, at each logging position, our inversion operator recovers the formation properties of a 1D layered medium [64, 99].

In this work, as measurement acquistion system, we first consider a co-axial LWD instrument equipped with two transmitters and two receivers (see Figure 2.2). H^1_{zz} and H^2_{zz} are the zz-couplings of the magnetic field measured at the first and second receivers, respectively (the first and second subscripts denote the orientation of the transmitter and receiver, respectively). Then, we define the attenuation and phase difference as follows:

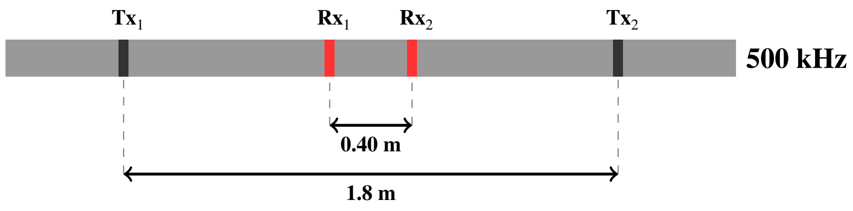


Figure 2.2: Conventional LWD logging instrument. Tx$_i$ and Rx$_i$ are the transmitters and the receivers, respectively.

$$\ln \frac{H^1_{zz}}{H^2_{zz}} = \underbrace{\ln \frac{\mid H^1_{zz} \mid}{\mid H^2_{zz} \mid}}_{\times 20\log(e) =: \text{attenuation } (dB)} + i \underbrace{\left(ph(H^1_{zz}) - ph(H^2_{zz})\right)}_{\times \frac{180}{\pi} =: \text{phase difference (degree)}} , \tag{2.3}$$

where ph denotes the phase of a complex number. We then record the average of the attenuations and phase differences associated with the two transmitters, and we denote these values as *LWD coaxial*.

Then, we consider a short-spacing configuration corresponding to a deep azimuthal instrument equipped with one transmitter and one receiver, as shown in Figure 2.3. In this logging instrument, the distance between transmitter and

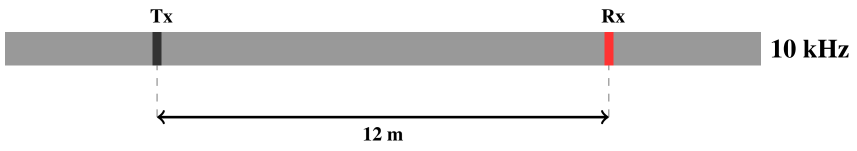


Figure 2.3: Short-spacing of a deep azimuthal logging instrument. Tx and Rx are the transmitter and the receiver, respectively.

receiver is significantly larger than that of the previously considered LWD instrument. It also employs tilted receivers that are sensitive to the presence of bed boundaries. We record several measurements with this logging instrument: (a) the attenuation and phase differences, denoted as *deep coaxial*, computed using Equation (2.3) with $H_{zz}^2 = 1$, and (b) the attenuation and phase differences of a directional measurement expressed as:

$$Geosignal = \ln \frac{H_{zz} - H_{zx}}{H_{zz} + H_{zx}} = \underbrace{\ln \frac{\mid H_{zz} - H_{zx} \mid}{\mid H_{zz} + H_{zx} \mid}}_{\times 20 \log(e) =: \text{attenuation } (dB)} + i \underbrace{\left(ph(H_{zz} - H_{zx}) - ph(H_{zz} + H_{zx})\right)}_{\times \frac{180}{\pi} =: \text{phase difference (degree)}} . \tag{2.4}$$

These measurements exhibit a discontinuity as a function of the dip angle at 90 degrees. Indeed, such discontinuity is essential in the measurements if one wants to discern between top and bottom of the logging instrument (see Figure 2.4).

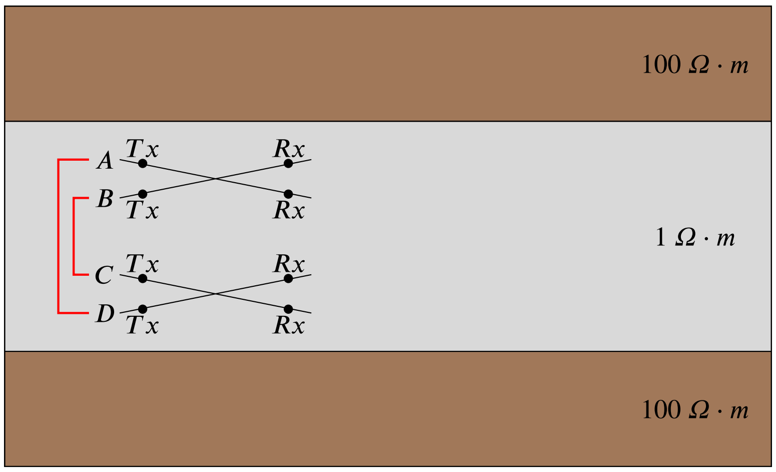


Figure 2.4: Illustration with four logging trajectories. By symmetry, measurements recorded with trajectories A and D are identical. The same occurs with trajectories B and C. If these measurements are continuous with respect to the dip angle, then they coincide at 90 degrees, which disables the possibility of identifying if a nearby bed boundary is located above or below the logging instrument.

For our borehole resistivity applications, we consider a zero-thickness borehole embedded in a three-layer medium (see Figure 2.5). A common practice in the field is to characterize this medium with seven parameters, as described in Figure 2.5. In this work, to simplify the problem, we consider only five of them by restricting the search to isotropic formations ($\rho_v = \rho_h$) with zero dip angle ($\beta = 0$), as illustrated in Figure 2.6. Thus, $n_p = 5$.

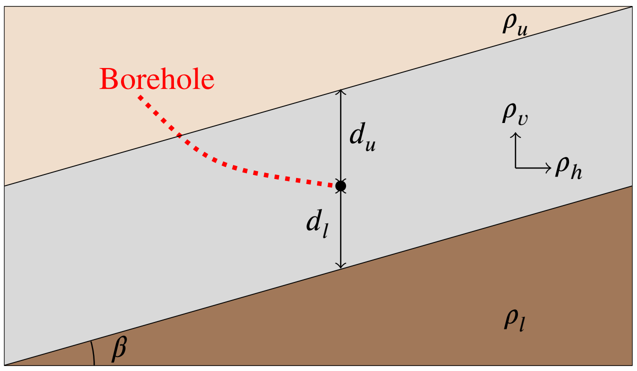
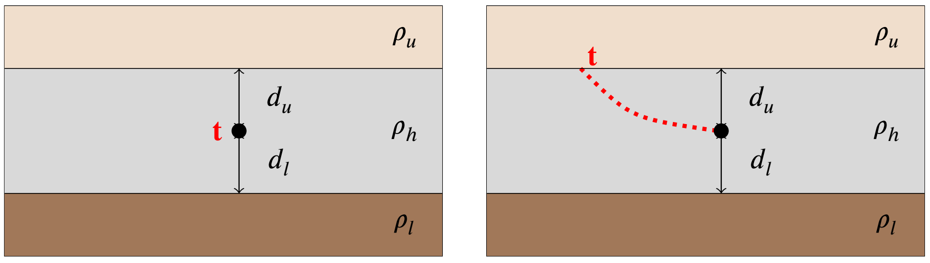


Figure 2.5: Well trajectory in a 1D medium. The black circle indicates the last trajectory position. ρ_h and ρ_v are the horizontal and vertical resistivities of the host layer corresponding to the final logging position, respectively. ρ_u and ρ_l are the resistivity values of the upper and lower layers to the host layer, respectively. d_u and d_l show the distance from the final logging position to the upper and lower bed boundaries, respectively.



(a) Example B.1: trajectory with 1 logging positions

(b) Example B.2: trajectory with 65 logging positions

Figure 2.6: Model problems corresponding to examples B.1 and B.2, respectively.

In this example, we consider two cases (see Figure 2.6) according with the different numbers of logging positions we consider per data sample.

2.1.5.1 Example B.1: One Logging Position

In this case, each trajectory consists of a single logging position. Therefore, for each sample, we record six real numbers (three attenuations and three phases), i.e., $n_m = 6$. At each logging position, the trajectory is described by one number: the trajectory dip angle. Thus, $n_t = 1$.

2.1.5.2 Example B.2: Sixty-Five Logging Positions

In this case, the logging trajectory of each sample is formed by 65 logging positions with a logging step size of 0.3048 m (see [130, 131] for further details). Thus, for each Earth model $\mathbf{p}$, we parametrize $\mathbf{m}$ with $6 \times 65 = 390$ real numbers ($n_m = 390$). For this example, we assume that the variation of the dip angle at a given logging position with respect to the previous one is constant. We denote that constant dip angle variation as α_v. Then, at the i-th logging position, the trajectory dip angle is $\alpha_i = \alpha_{ini} + (i-1)\alpha_v$, where α_{ini} is the initial dip angle. Hence, we have $n_t = 2$.

2.2 Data Space and Ground Truth

In this work, we employ a DNN to approximate the discrete inverse operator $\mathcal{I}$. Given a supervised database of n-pairs $(\mathbf{m}_i, \mathcal{I}(\mathbf{t}_i, \mathbf{m}_i))$, $i = 1, ..., n$, the DNN builds an approximation of the unknown function $\mathcal{I}$. This section describes the construction of the supervised database.

We first select the number of samples, n, and two subspaces of $\mathbb{R}^{n_p}$ and $\mathbb{R}^{n_t}$, respectively. Then, we select the n samples in those subspaces, namely, $((\mathbf{t}_1, \mathbf{p}_1), ..., (\mathbf{t}_n, \mathbf{p}_n))$. To each of these samples, we apply the operator $\mathcal{F}$. That is, we compute $(\mathcal{F}(\mathbf{t}_1, \mathbf{p}_1), ..., \mathcal{F}(\mathbf{t}_n, \mathbf{p}_n))$. Finally, the n-pairs $(\mathbf{m}_i, \mathcal{I}(\mathbf{t}_i, \mathbf{m}_i)) := (\mathcal{F}(\mathbf{t}_i, \mathbf{p}_i), \mathbf{p}_i)$, $i = 1, ..., n$ form our supervised database.

We denote by $\mathbf{T} \in \mathbb{R}^{n_t \times n}$ to the set of all trajectory samples $(\mathbf{t}_1, ..., \mathbf{t}_n)$. In other words, $\mathbf{T}$ is a matrix with $\mathbf{t}_i$ being its i-th column. Similarly, we define $\mathbf{M} = (\mathbf{m}_1, ..., \mathbf{m}_n) \in \mathbb{R}^{n_m \times n}$ and $\mathbf{P} = (\mathbf{p}_1, ..., \mathbf{p}_n) \in \mathbb{R}^{n_p \times n}$.

Example A: Simple model problem with known analytical solution. We select $n = 10^3$ uniformly spaced samples within the subspace $[-33, 33] \subset \mathbb{R}$.

Example B: Inversion of borehole resistivity measurements. We select $n = 10^6$. Then, for the five parameters described in Section 2.1.5, we select random samples of the following rescaled variables over the corresponding intervals forming a subspace of $\mathbb{R}^5$:

$$\begin{aligned} \log(\rho_l), \log(\rho_u), \log(\rho_h) &\in [0, 3] \\ \log(d_l), \log(d_u) &\in [-2, 1]. \end{aligned} \tag{2.5}$$

We consider arbitrary high-angle trajectories. For each model problem, we randomly select the trajectory parameters within the following intervals:

$$\begin{array}{lll} \alpha_{ini} & \in [83^\circ, 97^\circ] & \\ \alpha_v & \in [-0.045^\circ, 0.045^\circ] & \text{(only for Example B.2).} \end{array} \tag{2.6}$$

2.3 Data Preprocessing

Notation. For each output parameter of $\mathcal{F}$ and $\mathcal{I}$, we denote by $\mathbf{x} = (x_1, ..., x_n)$ the n-samples associated with that parameter. These x_i are real scalar values for $i = 1, ..., n$. For example, in the borehole resistivity example, each variable $\mathbf{x}$ contains n samples of each particular geophysical quantity such as resistivities, distances, or given measurements (attenuations, phases, etc.). Each dimension corresponds to a particular value (sample) of that variable, for example, the geosignal attenuation recorded at a specific logging position. From the algebraic point of view, the variable $\mathbf{x}$ denotes a row of either matrix $\mathbf{M}$ or $\mathbf{P}$.

Data preprocessing algorithm. This algorithm consists of three steps.

1. **Logarithmic change of coordinates**. We introduce the following change of variables:
$$\mathcal{R}_{\ln}(\mathbf{x}) := (\ln x_1, ..., \ln x_n). \tag{2.7}$$
For some geophysical variables (e.g., resistivity), this change of variables ensures that equal-size relative errors correspond to similar-size absolute errors. Thus, this change of variables allows us to perform *local* (within a variable) comparisons.

2. **Remove outlier samples**. In practice, often outlier measurements are present in the sample database. These outliers appear due to measurement error or the physics of the problem. For example, in borehole resistivity measurements, some apparent resistivity measurements approach infinity, producing “horns” in the logs. When outlier measurements exists in any particular variable of the i-th sample x_i, then the entire sample should be removed. Otherwise, outlier measurements affect the entire minimization problem, leading to poor numerical results. The removal process may be automated using statistical indicators, or decided by the user based on a priori physical knowledge about the problem. We follow this second approach in this work.

3. **Linear change of coordinates**. We now introduce a linear rescaling mapping into the interval $[0.5, 1.5]$. We select this interval since it has unit length and the mean of a normal (or a uniform) distribution variable $\mathbf{x}$ is equal to one. Let $x_{\min} := \min_i x_i$, $x_{\max} := \max_i x_i$. We define
$$\mathcal{R}_{lin}(\mathbf{x}) := \left(\frac{x_1 - x_{\min}}{x_{\max} - x_{\min}} + 0.5, ..., \frac{x_n - x_{\min}}{x_{\max} - x_{\min}} + 0.5 \right), \tag{2.8}$$
where the limits $x_{\min}$ and $x_{\max}$ are fixed for all possible approximations $\mathbf{x}^{app}$. This change of variables allows us to perform a *global* comparison between

errors corresponding to different variables since they all take values over the same interval.

Remark: $x_{\min}$ and $x_{\max}$ could also be selected based on the physically valid interval of each particular variable rather than on the training samples.

Variables classification. We categorize each input and output geophysical variable $\mathbf{x}$ into two types: either linear (A) or log-linear (B). When necessary, we shall indicate that a particular variable belongs to a specific category by adding the corresponding symbol as subindex of the variable, e.g., $\mathbf{x}_A$. Table 2.1 describes the domain of those variables as well as the rescaling employed for each of them. Variables of type A only require a global rescaling while those of type B require both a local and a global change of variables.

Geophysical Variables	Category	Domain	Rescaling
Angles, attenuations, phases, and geosignals	A	$\mathbb{R}^n$	$\mathcal{R}_{lin}(\mathbf{x})$
Apparent resistivities, resistivities, and distances	B	$(a, \infty)^n$ $a > 0$	$\mathcal{R}_{lin}(\mathcal{R}_{\ln}(\mathbf{x}))$

Table 2.1: Categories for geophysical variables: types A or B. We apply a different rescaling to each of them.

For simplicity, we denote by $\mathcal{R}$ the result of the above rescalings, i.e., $\mathcal{R}(\mathbf{x}_A) := \mathcal{R}_{lin}(\mathbf{x}_A)$, and $\mathcal{R}(\mathbf{x}_B) := \mathcal{R}_{lin}(\mathcal{R}_{\ln}(\mathbf{x}_B))$. In general, given a variable $\mathbf{x}$ (of category A or B), we represent $\mathbf{x}_{\mathcal{R}} := \mathcal{R}(\mathbf{x})$. Given a matrix $\mathbf{X} \in \mathbb{R}^{n_x \times n}$, we abuse notation and denote by $\mathbf{X}_{\mathcal{R}} := \mathcal{R}(\mathbf{X}) \in \mathbb{R}^{n_x \times n}$ to the matrix that results from applying operator $\mathcal{R}$ row-wise.

Remark: Substituting in Equation 2.7 the natural logarithm by the base ten logarithm does not affect the definition of $\mathcal{R}$. Results are identical.

2.4 Norms and Errors

We first introduce both the vector and the matrix norms that we use during the training process.

Norms. We introduce a norm $||\cdot||_{\mathbb{X}}$ associated with the variable $\mathbf{x}$. In general, we employ the l_1 or l_2 vector norms and, for matrices, the l_1 and Frobenius norms.

Absolute and relative errors. Let $\mathbf{x}^{app} = (x_1^{app}, ..., x_n^{app})$ be an approximation of $\mathbf{x}$. We define the absolute error A_e between $\mathbf{x}^{app}$ and $\mathbf{x}$ in the $|| \cdot ||_{\mathbb{X}}$ norm as

$$A_e^{\mathbb{X}}(\mathbf{x}^{app}, \mathbf{x}) := ||\mathbf{x}^{app} - \mathbf{x}||_{\mathbb{X}}. \tag{2.9}$$

This error measure has limited use since it is challenging to select an absolute error threshold that distinguishes between a *good* and a *bad* quality approximation. To overcome this issue, practitioners often employ relative errors. We define the relative error R_e in percent between $\mathbf{x}^{app}$ and $\mathbf{x}$ in the $|| \cdot ||_{\mathbb{X}}$ norm as:

$$R_e^{\mathbb{X}}(\mathbf{x}^{app}, \mathbf{x}) := 100 \frac{||\mathbf{x}^{app} - \mathbf{x}||_{\mathbb{X}}}{||\mathbf{x}||_{\mathbb{X}}}. \tag{2.10}$$

Error control. For a variable $\mathbf{x}$ and its approximation $\mathbf{x}^{app}$, we want to control the relative error of the rescaled variable, that is:

$$R_e^{\mathbb{X}}(\mathbf{x}_{\mathcal{R}}^{app}, \mathbf{x}_{\mathcal{R}}). \tag{2.11}$$

The value $B = ||\mathbf{x}_{\mathcal{R}}||_{\mathbb{X}}$ is expected to be similar for all variables $\mathbf{x}$. Thus:

$$\sum_{\mathbf{x}_{\mathcal{R}}} A_e^{\mathbb{X}}(\mathbf{x}_{\mathcal{R}}^{app}, \mathbf{x}_{\mathcal{R}}) = \sum_{\mathbf{x}_{\mathcal{R}}} ||\mathbf{x}_{\mathcal{R}}^{app} - \mathbf{x}_{\mathcal{R}}||_{\mathbb{X}} \approx B \sum_{\mathbf{x}_{\mathcal{R}}} \frac{||\mathbf{x}_{\mathcal{R}}^{app} - \mathbf{x}_{\mathcal{R}}||_{\mathbb{X}}}{||\mathbf{x}_{\mathcal{R}}||_{\mathbb{X}}} = \frac{B}{100} \sum_{\mathbf{x}_{\mathcal{R}}} R_e^{\mathbb{X}}(\mathbf{x}_{\mathcal{R}}^{app}, \mathbf{x}_{\mathcal{R}}). \tag{2.12}$$

Therefore, the minimum of the first and last terms of the above equation coincide.

2.5 DNN Architectures

To approximate the forward and inverse problems, we use DNN architectures based on residual-type blocks [59, 110] with convolutional operators [60, 67, 74, 151]. This work does not discuss optimal data sampling techniques nor the decision-making for the optimal selection of DNN architectures [89, 109]. In the following, we first define the main operators of our DNN architectures, followed by a description of the forward and inverse DNN architectures.

We denote by $\mathcal{N}$ to our nonlinear *activation function*. In our case, we employ the *rectified linear unit (ReLU)*, defined component-wise for each entry x as $\max(0, x)$ [60]. We now introduce a 1D convolutional operator $\mathbf{C}_{\psi}^{c,k}$, where c is the filter size (output dimensionality), k the kernel size, and ψ the weights [59, 60]. Then, we define the following block:

$$\mathbf{B}_{\psi}^{c,k} := \left(\mathcal{N} \circ \mathbf{C}_{\psi^1}^{c,k} \circ \mathcal{N} \circ \mathbf{C}_{\psi^2}^{c,k} + \mathbf{C}_{\psi^3}^{c,k} \right), \tag{2.13}$$

where now $\psi = (\psi^1, \psi^2, \psi^3)$ are all weights associated to block $\mathbf{B}_{\psi}^{c,k}$.

2.5.1 Forward Problem DNN Architecture

Each input sample has dimension $n_p + n_t$, and contains the variables representing the material properties and the trajectory. We define our DNN architecture as:

$$\mathcal{F}_{\mathcal{R},\phi} := \mathcal{N} \circ \mathbf{C}_{\phi_6}^{c_6,1} \circ \mathbf{L} \circ \mathbf{U} \circ \mathbf{B}_{\phi_5}^{c_5,3} \circ \mathbf{U} \circ \mathbf{B}_{\phi_4}^{c_4,3} \circ \cdots \circ \mathbf{U} \circ \mathbf{B}_{\phi_1}^{c_1,3}, \tag{2.14}$$

where

- $\mathbf{U}$ is a 1D upsampling operator with upsampling factor equal to two (using the *TensorFlow* routine upsampling1D [2, 102, 142]).
- $\mathbf{L}$ is a bilinear resampling operator with resampling factor equal to the number of logging positions [102, 142], i.e., 1 for Example B.1 and 65 for Example B.2.
- $c_i := 40i$, for $i = 1, \cdots, 5$ and $c_6 = n'_m = 6$, where n'_m is the number of evaluated measurements per logging position.
- $\phi = \{\phi_i : \ i = 1, \cdots, 6\}$ is a set of all weights associated to the forward DNN architecture.

$\mathbf{L}$ expands (in case of 65 logging positions) or shrinks (in case of one logging position) its input dimension. The output of the mentioned bilinear resampling is a matrix in which its first dimension is equal to the number of logging positions [102, 142]. All the resampling operators considered in the Equation 2.14 raise/shrink the dimension of their input gradually to avoid missing information due to a sudden dimension change. The output is a matrix of dimension (n_l, n'_m), where n_l is the number of logging positions.

2.5.2 Inverse Problem DNN Architecture

The input of the DNN is a matrix of dimension (n_l, n'_m+2), where n_l is the number of logging positions. The first two columns of the aforementioned matrix are the sine and cosine of the trajectory dip angle at each logging position. Analogously to the forward problem, we consider the following architecture:

$$\mathcal{I}_{\mathcal{R},\theta} := \mathcal{N} \circ \mathbf{D}_{\theta_7}^{n_p} \circ \mathbf{S} \circ \mathbf{B}_{\theta_6}^{c_6,3} \circ \mathbf{B}_{\theta_5}^{c_5,3} \circ \mathbf{B}_{\theta_4}^{c_4,3} \circ \cdots \circ \mathbf{B}_{\theta_1}^{c_1,3}, \tag{2.15}$$

- $\mathbf{D}_\theta^n$ is a fully-connected layer with n being its number of units and θ its weights [2, 60].
- $\mathbf{S}$ is a flattening layer that receives a 2D matrix and outputs a 1D vector [2].

- $c_i := 40i$, for $i = 1, \cdots, 5$.
- $\theta = \{\theta_i : \ i = 1, \cdots, 7\}$ is a set of all the weights associated to each block and layer

$\mathbf{D}_{\theta_7}^{n_p}$ performs the ultimate feature extraction and down-sampling. The output of this DNN is a vector consisting of material properties.

2.6 Loss Function

In this section, we consider a set of weights $\theta \in \Theta$ and a function $\mathcal{I}_{\mathcal{R},\theta}$ that depends upon the selected DNN architecture. Then, we introduce a loss function $L(\mathcal{I}_{\mathcal{R},\theta})$. We define the minimizer of the loss function over all possible weight sets θ as:

$$\mathcal{I}_{\mathcal{R},\theta*} := \arg\min_{\theta\in\Theta} L(\mathcal{I}_{\mathcal{R},\theta}). \tag{2.16}$$

Function $\mathcal{I}_{\theta*} := \mathcal{R}^{-1} \circ \mathcal{I}_{\mathcal{R},\theta*} \circ \mathcal{R}$ is the final DNN approximation of $\mathcal{I}$. In the following, we analyze the advantages and limitations associated with the use of different loss functions.

2.6.1 Data Misfit

A simple loss function based on the data misfit is given by:

$$L(\mathcal{I}_{\mathcal{R},\theta}) := ||\mathcal{I}_{\mathcal{R},\theta}(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{P}_{\mathcal{R}}||_P. \tag{2.17}$$

In the above equation, symbol $|| \cdot ||_P$ indicates l_1 or Frobenius norms introduced in Section 2.4.

Example A: Model problem with known analytical solution. In this example, $n_p = 1$. Thus, matrix norms reduce to vector norms. Figure 2.7 illustrates the results we obtain using the l_1 and l_2 norms, respectively. These disappointing results are expected. In the case of l_2-norm, for a sufficiently flexible DNN architecture the exact solution is $\mathcal{I}_{\theta*} = 0$. We want to minimize

$$\sum_{i\in I} \left(\mathcal{I}_{\mathcal{R},\theta}(m_i) - p_i\right)^2, \tag{2.18}$$

where $I = \{1, ..., n\}$ denotes the training dataset. For every sample of the form $(m_i, \sqrt{m_i})$, there exists another one $(m_i, -\sqrt{m_i})$, which is satisfied in our dataset by construction (see Section 2.2). Then, for a specific sample m_i, the solution that minimizes the loss must satisfy

$$(\mathcal{I}_{\mathcal{R},\theta}(m_i) - \sqrt{m_i})^2 + (\mathcal{I}_{\mathcal{R},\theta}(m_i) - (-\sqrt{m_i}))^2. \tag{2.19}$$

Taking the derivative of Eq. (2.19) with respect to $\mathcal{I}_{\mathcal{R},\theta}(m_i)$ and equaling it to zero, we obtain

$$4 \cdot \mathcal{I}_{\mathcal{R},\theta}(m_i) = 0. \tag{2.20}$$

Thus, for any sample m_i, the function is minimized when the approximated value is $\mathcal{I}_{\mathcal{R},\theta}(m_i) = 0$. We thus conclude that for l_2-norm, the approximated solution must be $\mathcal{I}_{\theta*} = 0$.

In the case of l_1-norm, any solution in between the two square root branches is valid. We want to minimize

$$\sum_{i \in I} |\mathcal{I}_{\mathcal{R},\theta}(m_i) - p_i|, \tag{2.21}$$

where $I = \{1, ..., n\}$ denotes the training dataset. Then, for a specific sample m_i the solution that minimizes the loss must satisfy

$$|\mathcal{I}_{\mathcal{R},\theta}(m_i) - \sqrt{m_i}| + |\mathcal{I}_{\mathcal{R},\theta}(m_i) - (-\sqrt{m_i})|. \tag{2.22}$$

By analyzing each possible case, we can express Eq. (2.22) as follows

$$\begin{cases} -2 \cdot \mathcal{I}_{\mathcal{R},\theta}(m_i), & \text{if } \mathcal{I}_{\mathcal{R},\theta}(m_i) < -\sqrt{m_i}, \\ 2\sqrt{m_i}, & \text{if } -\sqrt{m_i} \leqslant \mathcal{I}_{\mathcal{R},\theta}(m_i) \leqslant \sqrt{m_i}, \\ 2 \cdot \mathcal{I}_{\mathcal{R},\theta}(m_i), & \text{if } \mathcal{I}_{\mathcal{R},\theta}(m_i) > \sqrt{m_i}. \end{cases} \tag{2.23}$$

We see that the loss function for the exact solution attains its minimum at every point $\mathcal{I}_{\theta*}(m_i) \in [-\sqrt{m_i}, \sqrt{m_i}]$. Moreover, our numerical solutions in Figure 2.7 confirm these simple mathematical observations. Thus, the data misfit loss function is unsuitable for inversion purposes.

2.6.2 Misfit of the Measurements

To overcome the aforementioned limitation, we consider the following loss function that measures the misfit of the measurements (see [68]):

$$L(\mathcal{I}_{\mathcal{R},\theta}) := \|(\mathcal{F}_{\mathcal{R}} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M, \tag{2.24}$$

where $\mathcal{F}_{\mathcal{R}} := \mathcal{R} \circ \mathcal{F} \circ \mathcal{R}^{-1}$, and $|| \cdot ||_M$ indicates a matrix norm of the type introduced in Section 2.4.

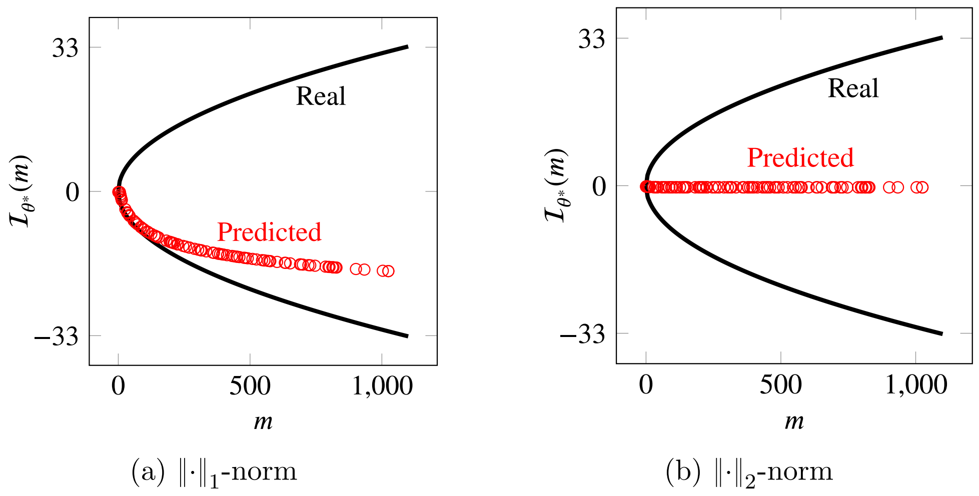


Figure 2.7: Analytical solution vs DNN predicted solution evaluated over the test dataset using the loss function based on the data misfit.

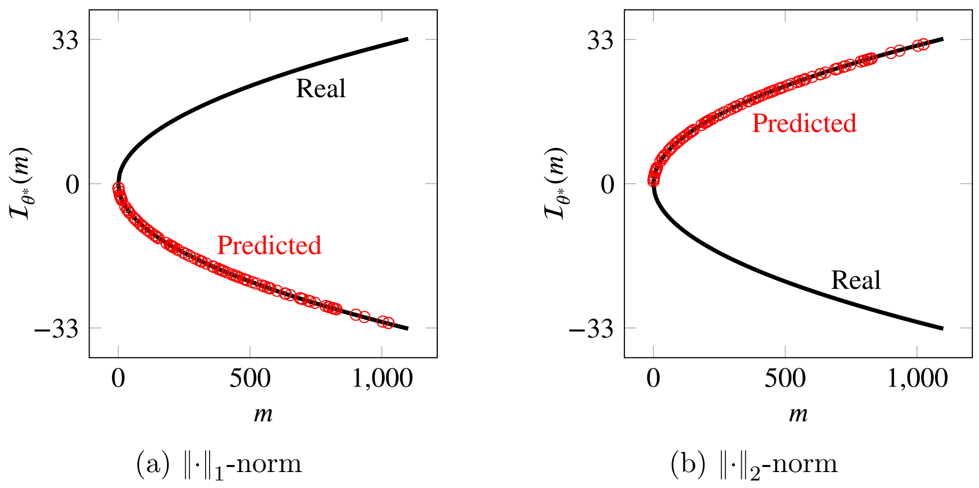


Figure 2.8: Analytical solution vs DNN predicted solution evaluated over the test dataset using the loss function based on the measurements misfit.

Example A: Model problem with known analytical solution. Figure 2.8 shows the inversion results when using the misfit of the measurements. We recover one of the possible solutions of the inverse operator. A regularization term could be introduced to select one solution branch over the other.

Despite the accurate results exhibited for the above example, the proposed loss function has some critical limitations that affect its performance. Namely, during training, it is necessary to evaluate the forward problem multiple times. Depending upon the size of the training dataset and number of iterations required to converge, this may lead to millions of forward function evaluations. Solving the forward problem for such large number of times is time-consuming even with a 1.5D semi-analytic simulator. Moreover, most forward solvers are implemented for Central Processing Unit (CPU) architectures, while the training of the DNN normally occurs on Graphics Processing Units (GPUs). This requires a permanent communication between GPU and CPU, which further slows down the training process. Additionally, porting the forward solver $\mathcal{F}$ to a GPU may be complex to implement and bring additional numerical difficulties.

2.6.3 Encoder-Decoder

To overcome the aforementioned implementation challenges, we propose to approximate the forward function using another DNN $\mathcal{F}_{\phi*}$, where $\phi^* \in \Phi$ are the parameters associated to the trained DNN. With this approach, we simultaneously train the forward and inverse operators solving the following optimization problem:

$$\begin{aligned}(\mathcal{F}_{\mathcal{R},\phi*}, \mathcal{I}_{\mathcal{R},\theta*}) := \arg \min_{\phi \in \Phi, \theta \in \Theta} \{ &\|(\mathcal{F}_{\mathcal{R},\phi} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M \\ &+ \|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M \},\end{aligned} \tag{2.25}$$

Function $\mathcal{F}_{\phi*} := \mathcal{R}^{-1} \circ \mathcal{F}_{\mathcal{R},\phi*} \circ \mathcal{R}$ is the final DNN approximation to $\mathcal{F}$. The first term in the above loss function constitutes an Encoder-Decoder DNN architecture [10] and ensures that function $\mathcal{I}_{\mathcal{R},\theta*}$ shall be a pseudo-inverse of $\mathcal{F}_{\mathcal{R},\phi*}$. The second term imposes that the forward DNN approximates the ground truth data. In particular, it prevents situations in which both $\mathcal{I}_{\mathcal{R},\theta*}$ and $\mathcal{F}_{\mathcal{R},\phi*}$ approximate the identity operator.

Example A: Model problem with known analytical solution. Figure 2.9 shows the results obtained with the Encoder-Decoder loss function. We recover accurate inversion results.

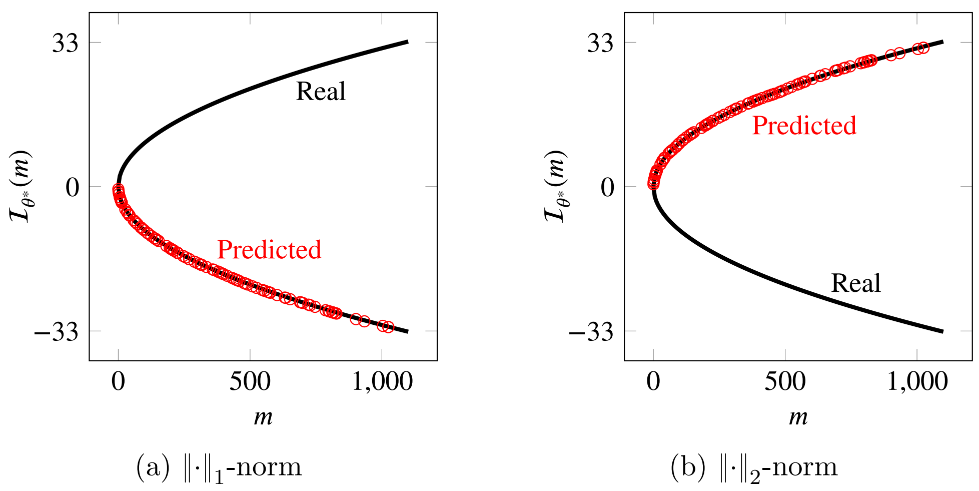


Figure 2.9: Analytical solution vs DNN predicted solution evaluated over the test dataset using the Encoder-Decoder loss function.

2.6.4 Two-step Approach

It is possible to decompose the above Encoder-Decoder based loss function into two steps: the first optimization problem intends to approximate the forward function, and the second one determines the inverse operator:

$$\mathcal{F}_{\mathcal{R},\phi^*} := \arg\min_{\phi\in\Phi} \|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M, \tag{2.26a}$$

$$\mathcal{I}_{\mathcal{R},\theta^*} := \arg\min_{\theta\in\Theta} \|(\mathcal{F}_{\mathcal{R},\phi^*} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M. \tag{2.26b}$$

Example A: Model problem with known analytical solution. Figure 2.10 shows the results of the inversion using the two-steps approach. We recover a faithful approximation of the inverse operator.

Remark A: Based on the above discussion, it may seem that optimization problems given by either Equations 2.25 or 2.26 are ideal to solve inverse problems. However, there is a critical issue that needs to be addressed. In Equation 2.26a, the forward DNN $\mathcal{F}_{\mathcal{R},\phi}$ is trained only for the given dataset samples. However, the output of the DNN approximation of the inverse operator $\mathcal{I}_{\mathcal{R},\theta}$ will often deliver data far away from the data space used to produce the training samples. This may lead to catastrophic results. To illustrate this, we consider our model problem with known analytical solution. If we consider a dataset with only positive values of p, then the following approximations will lead to a zero loss function:

$$\mathcal{F}_{\phi^*}(p) = \begin{cases} p^2 & \text{if } p > 0 \\ ap^2 & \text{if } p < 0 \end{cases} \qquad \mathcal{I}_{\theta^*}(m) = -\sqrt{m/a}, \tag{2.27}$$

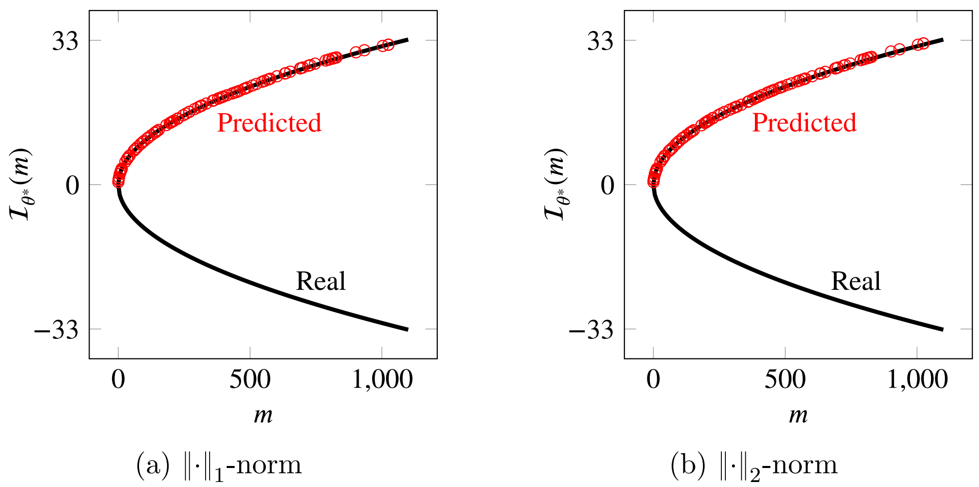


Figure 2.10: Analytical solution vs DNN predicted solution evaluated over the test dataset using the two-step based loss function.

for any $a > 0$. However, if $a \neq 1$, this approximation is far away from both our original forward and inverse solutions. To prevent these undesired situations, one should ensure that the output space of $\mathcal{F}_{\mathcal{R},\theta*}$ is sufficiently close to the space from which we obtain the training samples. However, this is often difficult to control.

2.6.5 Regularization Term

Inverse problems often exhibit non-unique solutions. Thus, in numerical methods, one introduces a regularization term to select a particular preferred solution out of all the existing ones.

In DL applications, standard regularization techniques seek to optimize the model architecture (e.g., by penalizing high-valued weights). Herein, we regularize the system by adding the loss function of Equation 2.17 measured in the l_1-norm to either the optimization problem given by Equation 2.25 or 2.26b. This extra term guides the solution towards the ones considered in the training dataset, which may be convenient. Nevertheless, such a regularization term often hides the fact that other different solutions of the inverse problem may coexist. We study the advantages and limitations of including this regularization term in detail in Section 2.8.

2.7 Implementation

To solve the forward problem, we employ a semi-analytic method [80] implemented in Fortran 90. It employs a Hankel transform to reduce the original 3D

Maxwell system to a sequence of uncoupled 1D problems, whose solutions are analytical in the Hankel domain [143]. Then, we perform a numerical inverse Hankel transform with an adaptive Andersson quadrature rule [8]. With it, we produce a dataset containing one million samples (*ground truth*). Each sample consists of a randomly selected 1D layered model (see Section 2.2 for details). We use 80% of the samples for training the DNNs, 10% for validating them, and the remaining 10% for testing.

We consider two DNN architectures to approximate $\mathcal{F}$ and $\mathcal{I}$, respectively. The forward function $\mathcal{F}$ is well-posed and continuous, while the inverse operator $\mathcal{I}$ is not even well-defined. Thus, we employ a simpler DNN architecture to approximate $\mathcal{F}$ than to approximate $\mathcal{I}$. See Section 2.5 for details. We use the l_1 norm for the loss function.

We implement our DNNs using *Tensorflow 2.0* [2] and *Keras* [29] libraries. To train the DNNs, we use a *NVIDIA Quadro GV100* GPU. Using this hardware device, we require almost 70 hours to simultaneously train $\mathcal{F}_{\mathcal{R},\phi*}$ and $\mathcal{I}_{\mathcal{R},\theta*}$. While the training process is time-consuming, it is performed *offline*. Then, the *online* part of the process consists of simply evaluating the DNN, which can deliver an inverse model for thousands of logging positions in a few seconds. This low *online* computational cost makes the DNN approach an excellent candidate to perform inversion during geosteering operations in the field.

2.8 Numerical Results

We perform a three step evaluation process of the results:

1. We first study the evolution of each term in the loss function during the training process. This analysis assesses the overall performance of the training process and, in particular, shows if any particular term of the loss function is driving the optimization procedure in detriment of other terms.

2. Second, we produce multiple cross-plots, which provide essential information about the adequacy of the selected loss function and dataset. These cross-plots indicate the possible non-uniqueness of the inverse problem at hand.

3. Finally, we apply the trained networks to invert three realistic synthetic models and analyze the overall success of the proposed DNN algorithm as well as its limitations.

The above evaluation process provides a step-by-step assessment of the adequacy of the proposed strategy for solving inverse problems.

In most cases, we observe similar results when we consider the Encoder-Decoder loss function given by Equation 2.25 and the two-step based loss function given by Equation 2.26. For brevity, we mostly focus on the Encoder-Decoder results. Additionally, we include one set of results using the two-step based loss function, for which the observed behavior is essentially different from that of the Encoder-Decoder process.

2.8.1 Evolution of the Loss Function

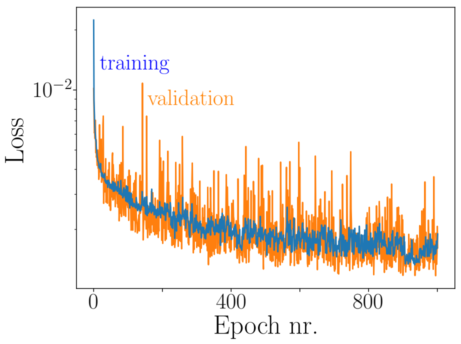
Figure 2.11 displays the evolution of the terms composing the Encoder-Decoder loss function described in Equation 2.25 for Example B.1. Figure 2.12 displays the corresponding results when we add the regularization term based on Equation 2.17. In both figures, we observe: (a) a proper reduction of the total loss function, indicating that the overall minimization process is successful; (b) an adequate balance between the loss contribution of the different terms composing each loss function, suggesting that all terms of the loss functions are simultaneously minimized; and (c) a satisfactory match between the loss functions corresponding to the training and the validation data samples, which indicates we avoid overfitting. We observe a similar behavior with Example B.2, which we skip for brevity. We do not detail the results per variable since the applied rescaling of Section 2.3 guarantees a good balance between different variables.

2.8.2 Cross-plots

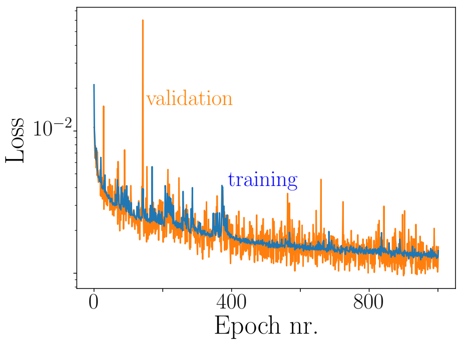
We consider the following types of cross-plots:

$$\begin{array}{lrcr} \text{Cross-plot 1:} & \mathcal{F}\circ\mathcal{I} & vs & \mathcal{F}_{\phi*}\circ\mathcal{I} \\ \text{Cross-plot 2:} & \mathcal{F}\circ\mathcal{I} & vs & \mathcal{F}_{\phi*}\circ\mathcal{I}_{\theta*} \\ \text{Cross-plot 3:} & \mathcal{F}\circ\mathcal{I} & vs & \mathcal{F}\circ\mathcal{I}_{\theta*} \\ \text{Cross-plot 4:} & \mathcal{I} & vs & \mathcal{I}_{\theta*} \end{array} \tag{2.28}$$

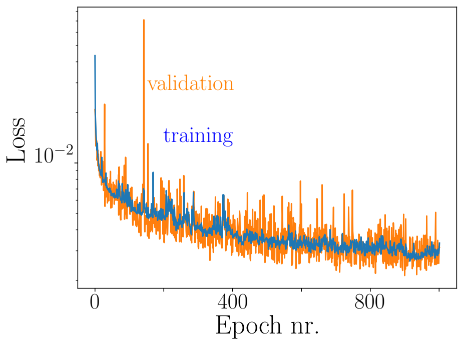
In the above, $\mathcal{F}$ and $\mathcal{I}$ are the exact functions and they define the *ground truth*, while the others are the *predictions* our DNNs deliver. In particular, in the first three types of cross-plots the ground truth is simply the identity mapping. We could display each type of cross-plot for the training, validation, and test data samples and for each variable. In our Example B, this makes a total of 69 cross-plots. In addition, we need to repeat them for each considered loss function. To compress this information, we quantify each cross-plot with a single number: the statistical measure R-squared (R^2), which represents how much variation of the ground truth is explained by the predicted value. When this value is close to 1, indicating a perfect matching between the predicted value and the ground truth,



(a) $\|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$

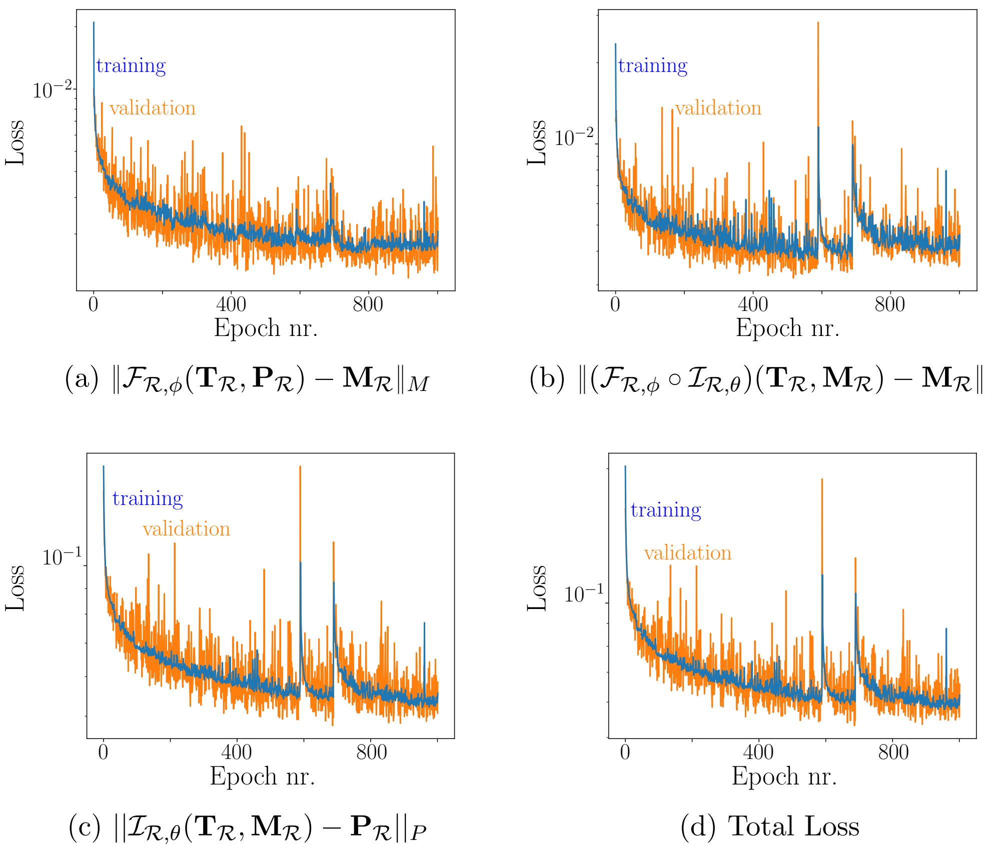


(b) $\|(\mathcal{F}_{\mathcal{R},\phi} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$



(c) Total Loss

Figure 2.11: Example B.1. Evolution of the different terms of the Encoder-Decoder loss function given by Equation 2.25 without regularization.



(a) $\|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}},\mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$

(b) $\|(\mathcal{F}_{\mathcal{R},\phi} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}},\mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$

(c) $||\mathcal{I}_{\mathcal{R},\theta}(\mathbf{T}_{\mathcal{R}},\mathbf{M}_{\mathcal{R}}) - \mathbf{P}_{\mathcal{R}}||_P$

(d) Total Loss

Figure 2.12: Example B.1. Evolution of the different terms of the Encoder-Decoder loss function given by Equation 2.25 with the regularization term prescribed by Equation 2.17.

we can safely omit these cross-plots. Otherwise, cross-plots display interesting information beyond what R^2 provides.

The proper interpretation of the cross-plots (or alternatively, R^2 factors) is of utmost importance. Cross-plots of type 1 (Equation 2.28_1) indicate how well the forward function is approximated over the given dataset. The cross-plots of type 2 (Equation 2.28_2) display how well the composition of the predicted forward and inverse mappings approximate the identity. These two types of cross-plots often deliver high R^2 factors, since the corresponding approximations are directly built into the Encoder-Decoder loss function given by Equation 2.25. Table 2.2 confirms those theoretical predictions for the most part.

An in-depth inspection of Table 2.2 reveals that for the the geosignal measurements (both attenuation and phase) corresponding to the Example B.1 without regularization, the cross-plots 2 exhibit significantly better R^2 factors than those corresponding to the cross-plots 1. Figure 2.13 shows the corresponding cross-plots. The anti-diagonal grey line shown in cross-plots of type 1 corresponds to dip angles of the logging instrument that are close to 90 degrees. At that angle, the geosignal is discontinuous. Thus, it is not properly approximated via DL algorithms, which approximate continuous functions. Cross-plots of type 2 seem to fix that issue by delivering higher R^2 factors and apparently nicer figures. However, they amplify the problem. In reality, the DL approximation of the inverse operator is inverting an incorrect forward approximation. Numerical results below illustrate this problem.

Obtaining high R^2 factors associated to cross-plots of type 3 (Equation 2.28_3) is a challenging task as we discuss in Remark A of Section 2.6. Equation 2.27 shows a simple example in which cross-plots of type 1 and 2 deliver perfect R^2 marks and results, while cross-plots of type 3 are disastrous. This is also the situation that occurs in Example B.2. (see Table 2.3). While the original training dataset is based on 1D Earth models, the one obtained after the predicted DNN inversion is a *piecewise* 1D Earth model, for which $\mathcal{F}_{\phi^*}$ is untrained for. When this occurs, the training database should be upgraded, either by increasing the space of the data samples or by selecting a different parameterization (e.g., measurements) for each sample. In our case, we choose to parametrize each sample independently (the later stategy) and we move to Example B.1.

Table 2.3 shows mixed results for the Example B.1. Results without regularization are unremarkable with the geosignal forecasts showing poor results. The DNN inverse approximation accurately inverts for the outcome predicted by the DNN forward approximation. Nevertheless, since the DNN predicts solutions far from the true forward function, the predictions are poor. Again, this poor forecasting occurs because the DNN inverse approximation encounters subsurface models for which the forward DNN approximation is untrained. As a result,

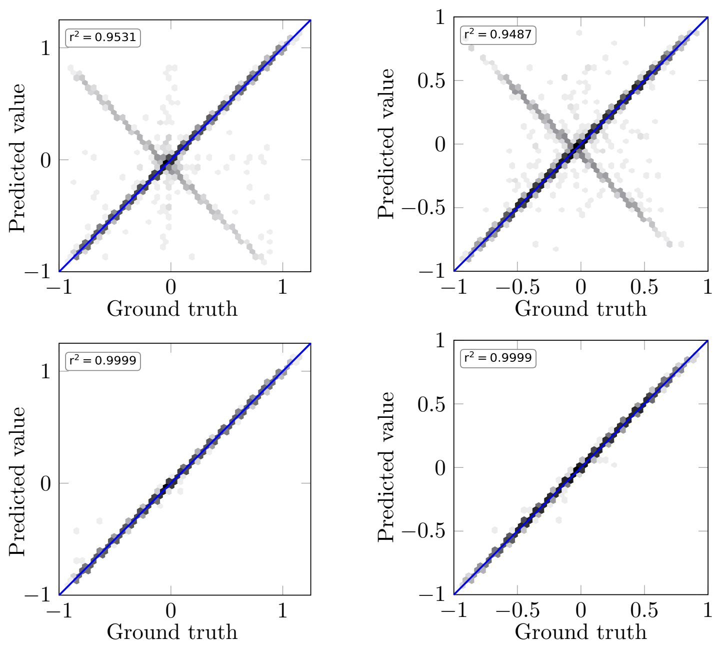


Figure 2.13: Geosignal cross-plots for the Example B.1 without regularization for the test dataset. First row: Cross-plots of type 1. Second row: Cross-plots of type 2. First column: Attenuation. Second column: Phase.

Cross-plots 1

R^2 factors	Atten. LWD Coaxial	Atten. Deep Coaxial	Atten. Deep Geosignal	Phase LWD Coaxial	Phase Deep Coaxial	Phase Deep Geosignal
Example B.1 Training Test Without Reg.	0.9997 0.9995	0.9992 0.9984	**0.9509** **0.9531**	0.9996 0.9990	0.9994 0.9991	**0.9468** **0.9487**
Example B.1 Training Test With Reg.	0.9998 0.9998	0.9998 0.9998	0.9897 0.9893	0.9998 0.9998	0.9998 0.9998	0.9893 0.9890
Example B.2 Training Test Without Reg.	0.9959 0.9924	0.9975 0.9960	0.9872 0.9775	0.9954 0.9920	0.9980 0.9974	0.9853 0.9765

Cross-plots 2

R^2 factors	Atten. LWD Coaxial	Atten. Deep Coaxial	Atten. Deep Geosignal	Phase LWD Coaxial	Phase Deep Coaxial	Phase Deep Geosignal
Example B.1 Training Test Without Reg.	0.9997 0.9997	0.9995 0.9994	0.9998 0.9999	0.9999 0.9999	0.9996 0.9996	0.9999 0.9999
Example B.1 Training Test With Reg.	0.9971 0.9970	0.9980 0.9979	0.9779 0.9785	0.9970 0.9970	0.9979 0.9978	0.9798 0.9803
Example B.2 Training Test Without Reg.	0.9931 0.9890	0.9958 0.9930	0.9800 0.9701	0.9933 0.9881	0.9967 0.9944	0.9821 0.9720

Table 2.2: R^2 factors for cross-plots of type 1 and 2 and Examples B.1 and B.2, with and without regularization, for training and test datasets. Numbers below 0.96 are marked in boldface.

both the forward and inverse DDN approximations depart strongly from the true solutions. In other words, the inverse can only comply with their composition to be close to the identity, which is not robust to deliver accurate and physically relevant approximations.

Cross-plots 3

R^2 factors	Atten. LWD Coaxial	Atten. Deep Coaxial	Atten. Deep Geosignal	Phase LWD Coaxial	Phase Deep Coaxial	Phase Deep Geosignal
Example B.1						
Without Reg.	0.9468	0.7406	0.0013	0.9383	0.9116	0.0167
With Reg.	0.9971	0.9979	0.9807	0.9969	0.9979	0.9856
Example B.2						
Without Reg.	0.5721	0.8383	0.0253	0.4546	0.8611	0.0284
With Reg.	0.9010	0.9701	0.5901	0.8621	0.9618	0.5877

Table 2.3: R^2 factors for cross-plots of type 3 and Examples B.1 and B.2, with and without regularization, for the test dataset.

To partially alleviate the above problem, we envision three possible solutions. First, we can increase the training dataset. This option is time-consuming and often impossible to achieve in practice. For example, herein, we already employ 1,000,000 samples. Second, we can include regularization. Results with regularization are of high quality (see Table 2.3). However, the regularization term may hide alternative physical solutions of the inverse problem. Thus, the regularization diminishes the ability to perform uncertainty quantification. Similarly, it may induce on the user excessive confidence in the results. A third option is to consider the two-step based loss function given by Equation 2.26. Following this approach, we first adjust the forward DNN approximation before training the DNN inverse approximation. Fixing the forward DNN often provides a proper forecast even in areas with a lower rate of training samples before producing a DNN approximation that approximates the inverse of the DNN forward approximation. Following this two-step approach without regularization, we obtain high R^2 factors for cross-plots of type 3: above 0.95 for the geosignal attenuation and phase, and above 0.99 for the remaining measurements.

Finally, the R^2 factors for the cross-plots of type 4 do not reflect on the accuracy of the DNN algorithm, but rather on the nature of the inverse problem at hand. Low R^2 factors indicate there exist multiple solutions. A regularization term (e.g., Equation 2.17) increases the R^2 indicator. Figure 2.14 clearly illustrates this fact. However, it is misleading to conclude that results without regularization are always worse. They may simply exhibit a different (but still valid) solution of the inverse problem.

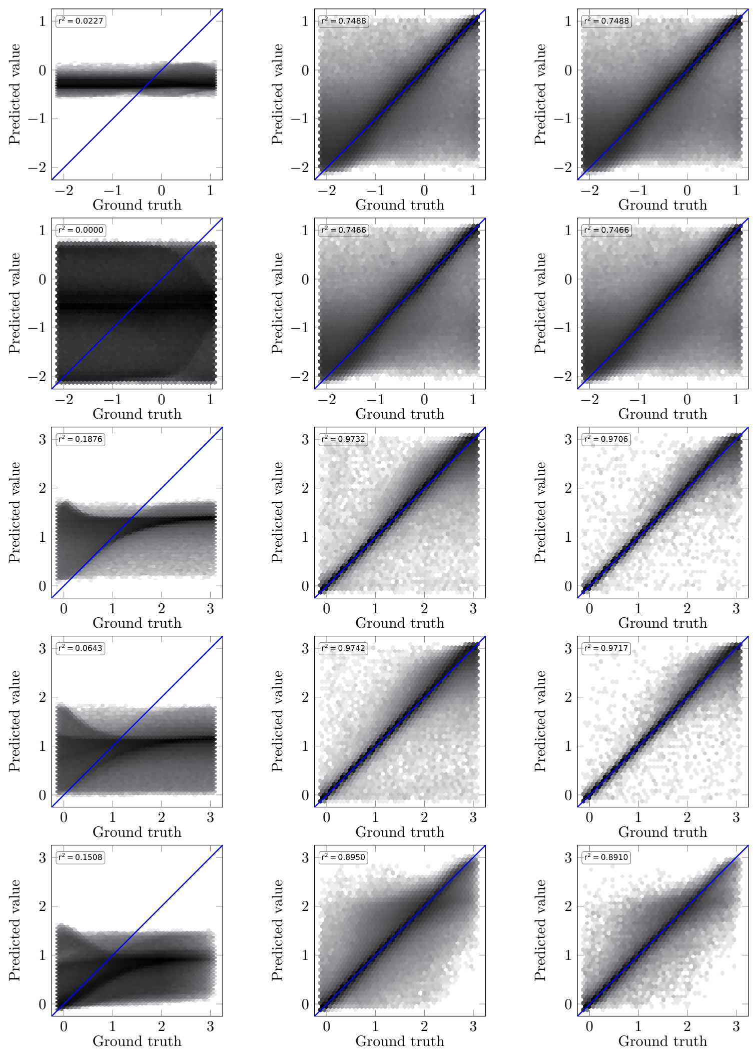


Figure 2.14: Cross-plots of type 4 for Example B.1 without regularization for the training dataset (first column), and with regularization for the training dataset (second column) and the test dataset (third column). First row: distance to the upper layer. Second row: distance to the lower layer. Third row: resistivity of upper layer. Fourth row: resistivity of lower layer. Fifth row: resistivity of central layer.

2.8.3 Inversion of Realistic Synthetic Models

We now consider three realistic synthetic examples to assess the performance of the inversion process. In terms of log accuracy, we observe qualitatively similar results for the attenuation and phase logs. Thus, in the following we only display the attenuation logs and omit the phase logs.

2.8.3.1 Model Problem I

Figure 2.15 describes a well trajectory in a synthetic model problem. The model has a resistive layer with a water-bearing layer underneath, and exhibits two geological faults.

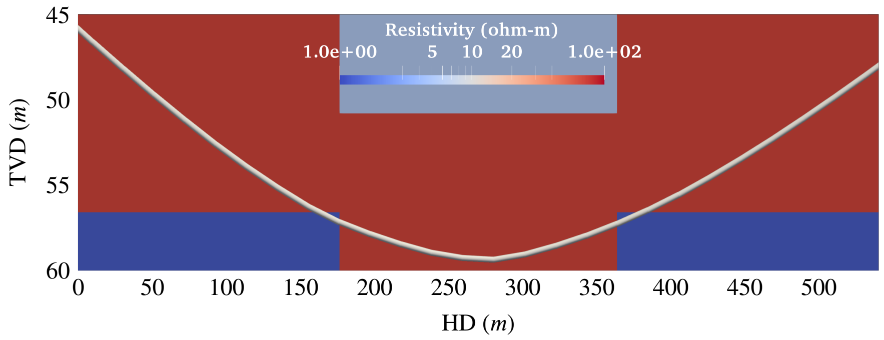
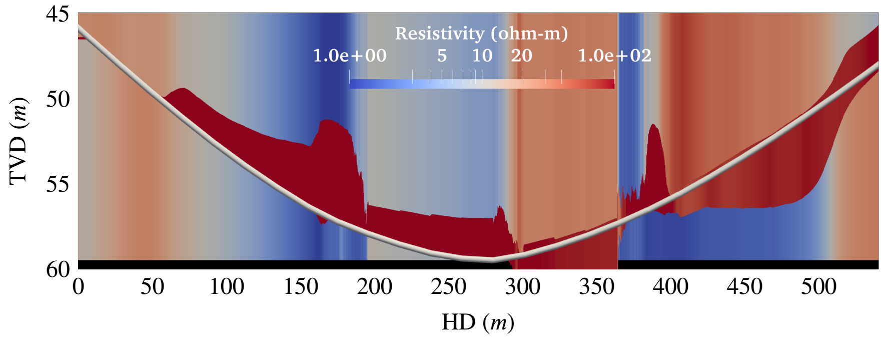


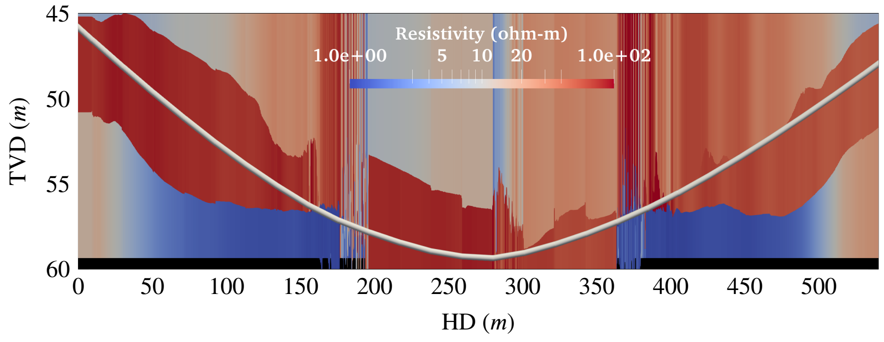
Figure 2.15: Formation of model problem I.

For the DNNs produced with the Example B.2 (with input measurements corresponding to 65 logging positions per sample), Figure 2.16 shows the corresponding inverted models using the Encoder-Decoder DNN with and without regularization. Results show inaccurate inversion results, specially for the case without regularization. Moreover, the predicted logs are far from the true logs, as Figure 2.17, and as expected from cross-plots 3 (see Table 2.3). The DNN inversion results are *piecewise* 1D models. However, the DNN approximation only trains with 1D models, not for piecewise 1D models, which explains the poor approximations they deliver (see Remark A on Section 2.6).

In the remainder of this section, we restrict to DNNs produced with Example B.1. That is, we parametrize all observations at one location using information from that location alone. Figure 2.18 shows the corresponding inverted models. For the case of the Encoder-Decoder loss function without regularization, we observe in Figure 2.18a an inverted model that is completely different from the original one. The corresponding logs (see Figure 2.19) are also inaccurate, as anticipated by the cross-plots results of type 3 shown in the previous subsection. When considering the two-step based loss function without regularization, the

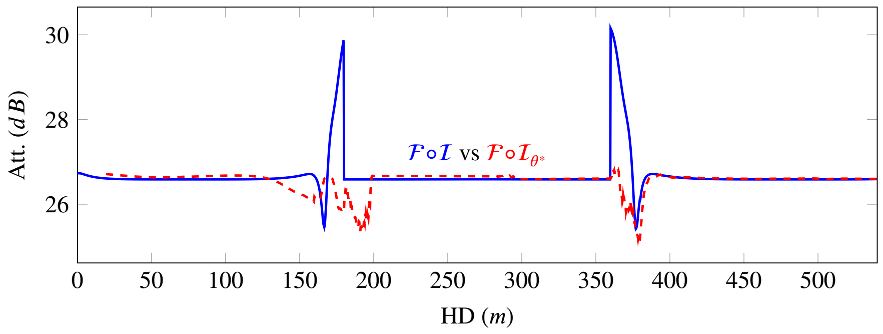


(a) Without regularization

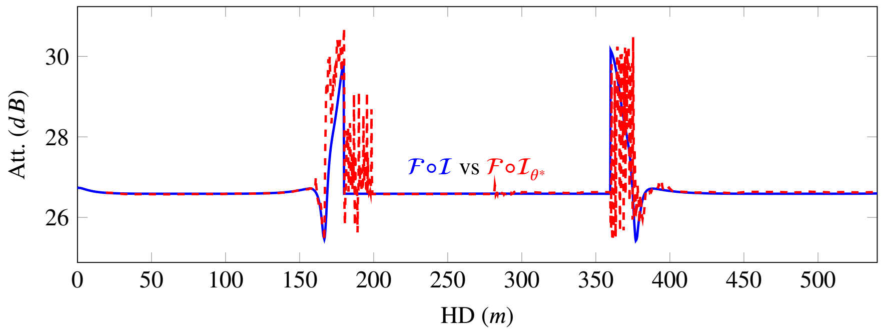


(b) With regularization

Figure 2.16: Inverted formation of model problem I using the inversion strategy of Example B.2, i.e., with input measurements corresponding to 65 logging positions per sample.



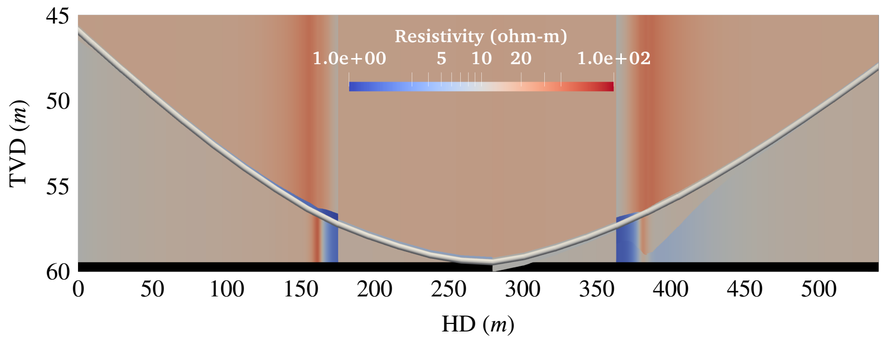
(a) LWD coaxial measurement. Without regularization



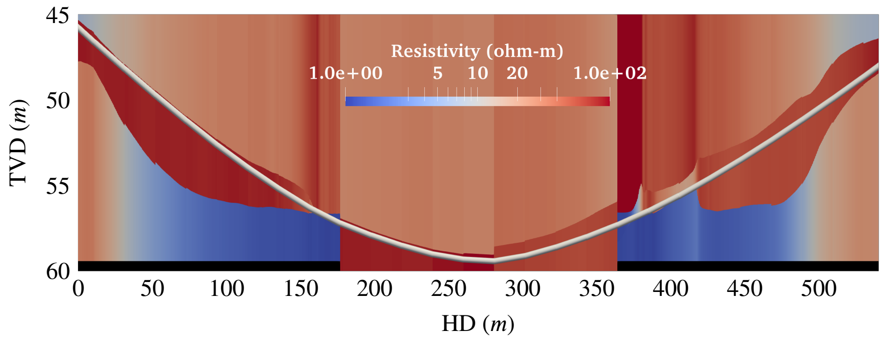
(b) LWD coaxial measurement. With regularization

Figure 2.17: Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta*}$ using the inversion strategy of Example B.2, i.e., with input measurements corresponding to 65 logging positions per sample.

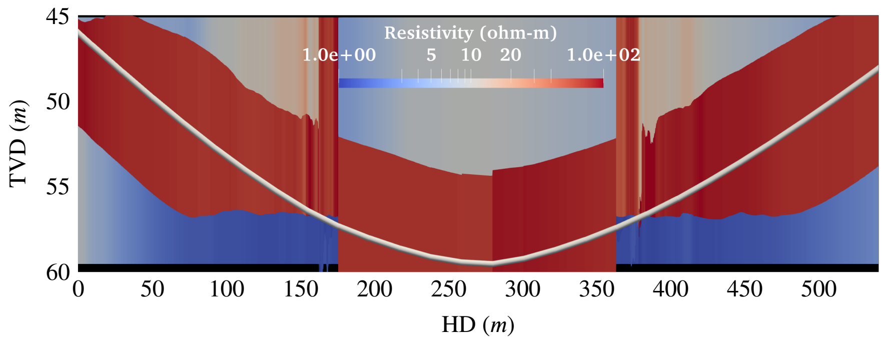
recovered model (see Figure 2.18b) is still quite different from the original one. Nonetheless, we observe a superb matching in the logs (see Figure 2.20), which indicates the presence of a different solution for the inverse problem. This confirms that the given measurements are insufficient to provide a unique solution for the inverse problem. For the case with regularization, inversion results (see Figure 2.18b) match the original model, and the corresponding logs properly approximate the synthetic ones, see Figure 2.21. Figures 2.22 and 2.23 confirm that our methodology delivers a proper training of the forward function approximation and the composition $\mathcal{F}_{\phi*} \circ \mathcal{I}_{\theta*}$, respectively.



(a) Predicted formation using the Encoder-Decoder loss function without regularization

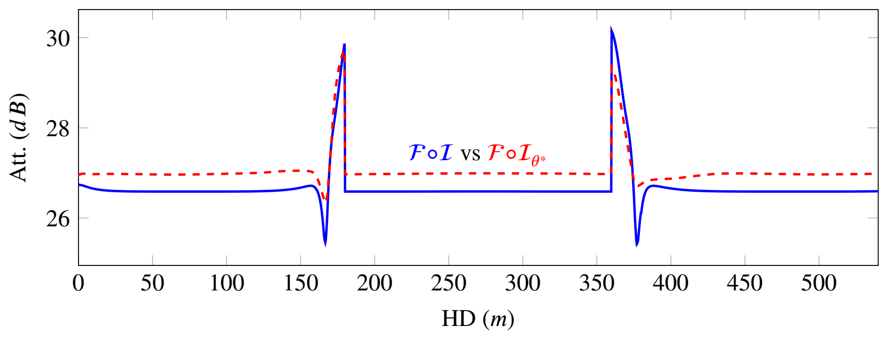


(b) Predicted formation using the two-step based loss function without regularization

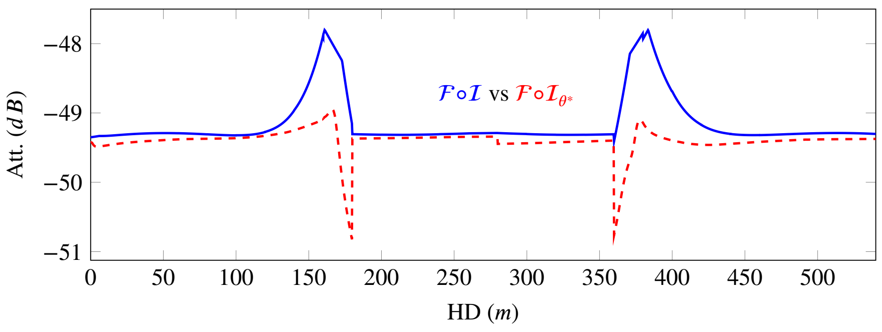


(c) Predicted formation using the Encoder-Decoder loss function with regularization

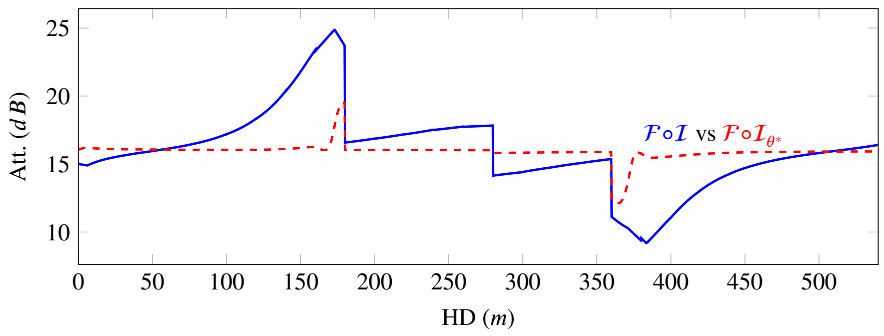
Figure 2.18: Inverted formation of model problem I using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) LWD coaxial measurement

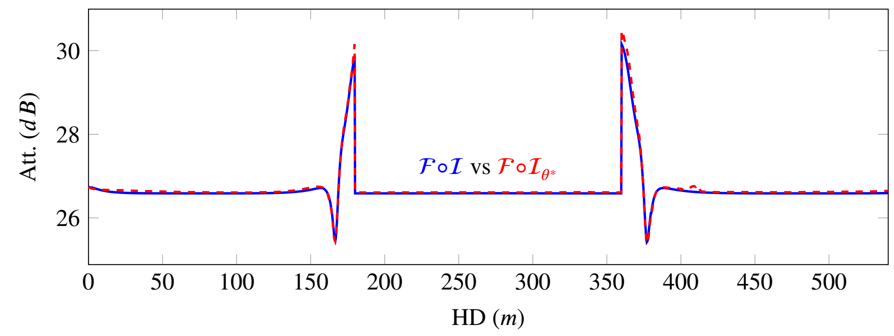


(b) Deep coaxial measurement

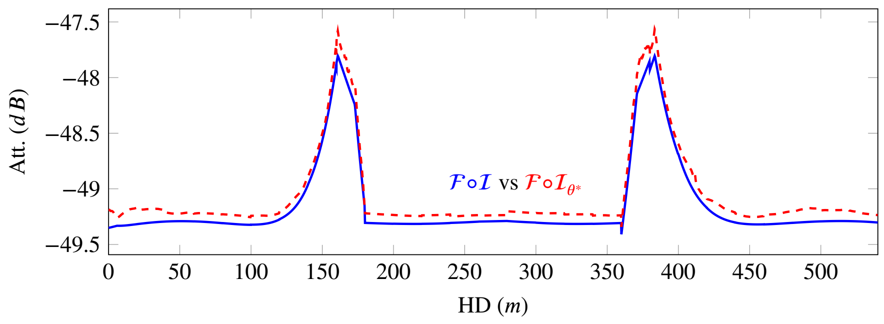


(c) Geosignal measurement

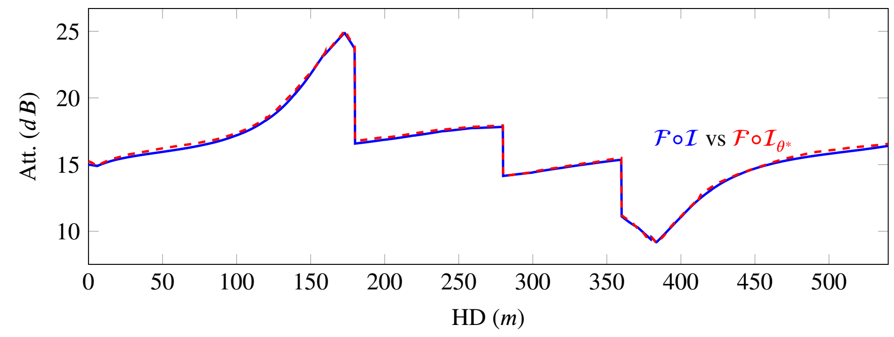
Figure 2.19: Model problem I. Comparison between $\mathcal{F} \circ \mathcal{I}$ and $\mathcal{F} \circ \mathcal{I}_{\theta *}$ without regularization using the Encoder-Decoder loss function and the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) LWD coaxial measurement

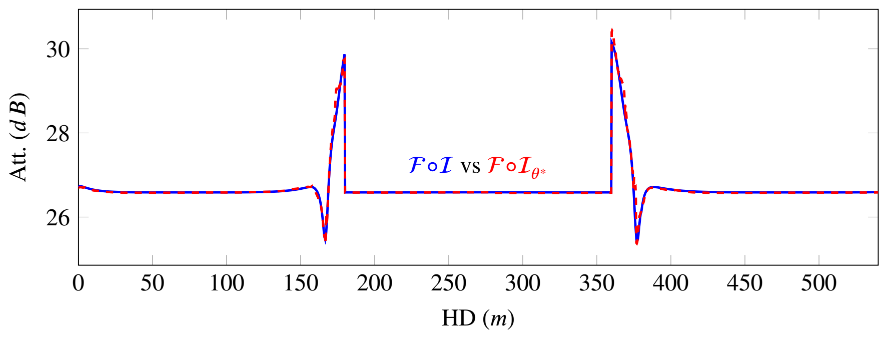


(b) Deep coaxial measurement

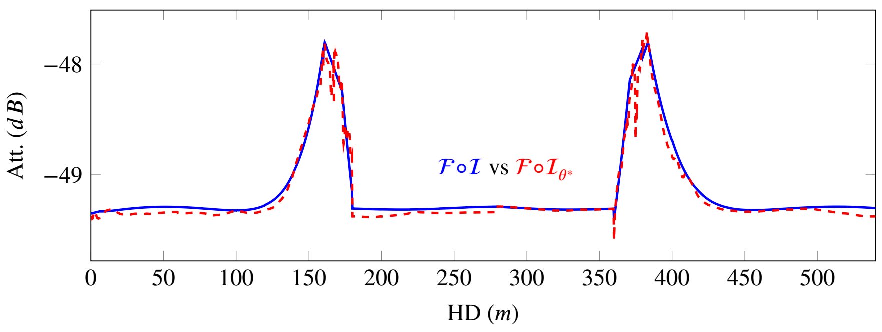


(c) Geosignal measurement

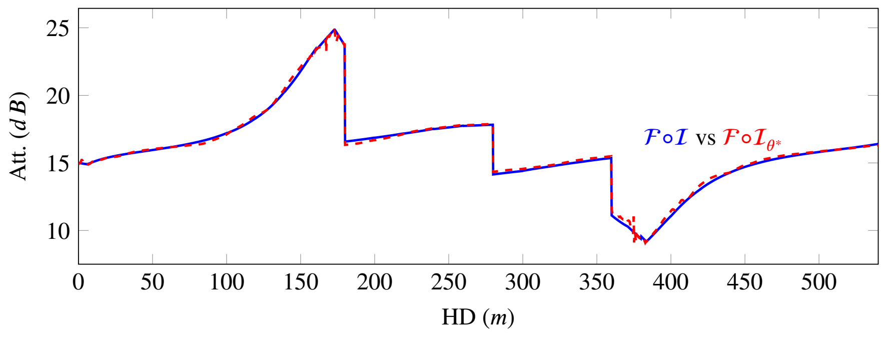
Figure 2.20: Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta*}$ using the two-step based loss function without regularization and the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) LWD coaxial measurement

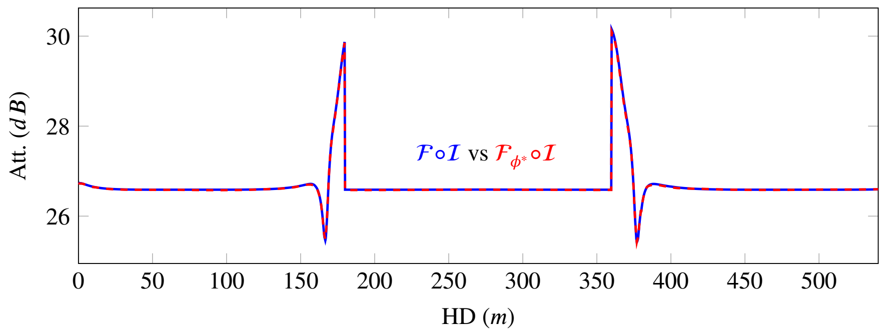


(b) Deep coaxial measurement

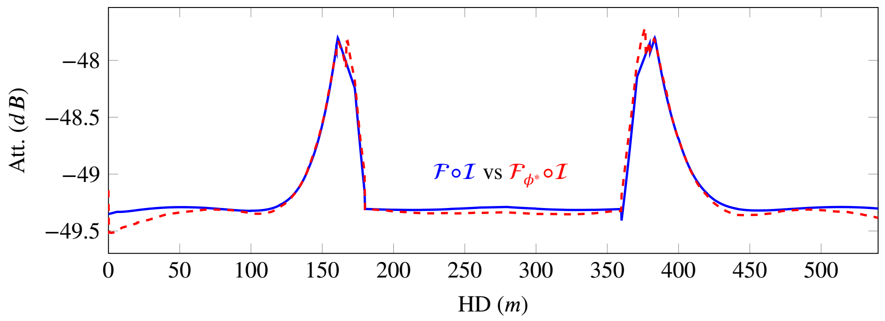


(c) Geosignal measurement

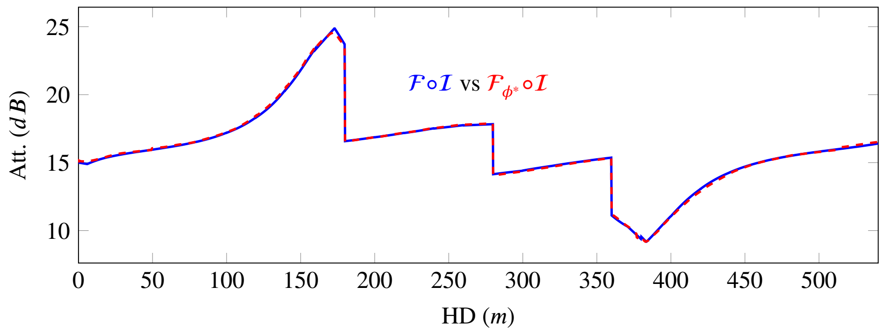
Figure 2.21: Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta*}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) LWD coaxial measurement

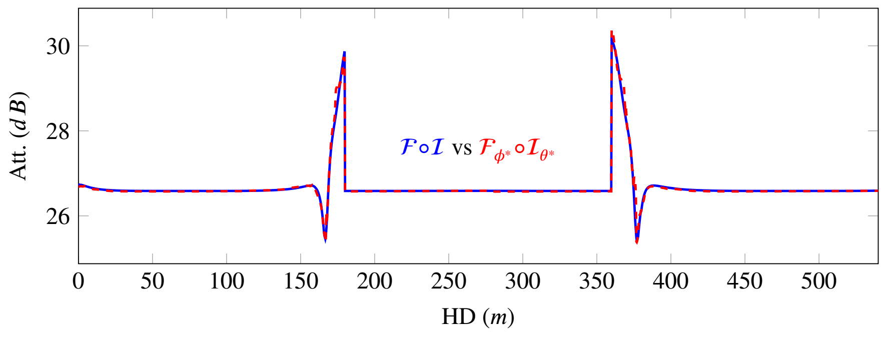


(b) Deep coaxial measurement

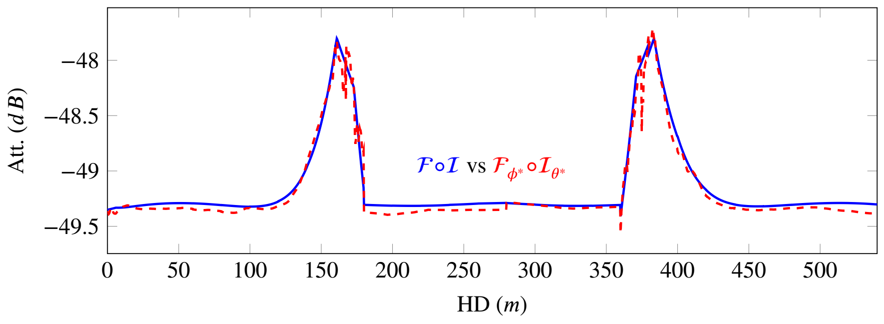


(c) Geosignal measurement

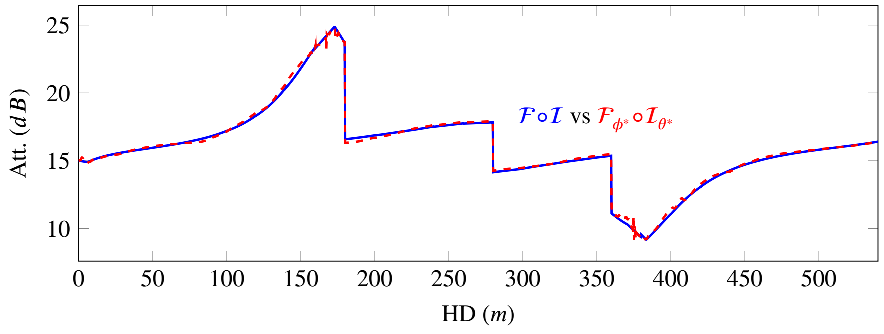
Figure 2.22: Model problem I. Comparison between $\mathcal{F} \circ \mathcal{I}$ and $\mathcal{F}_{\phi^*} \circ \mathcal{I}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) LWD coaxial measurement



(b) Deep coaxial measurement

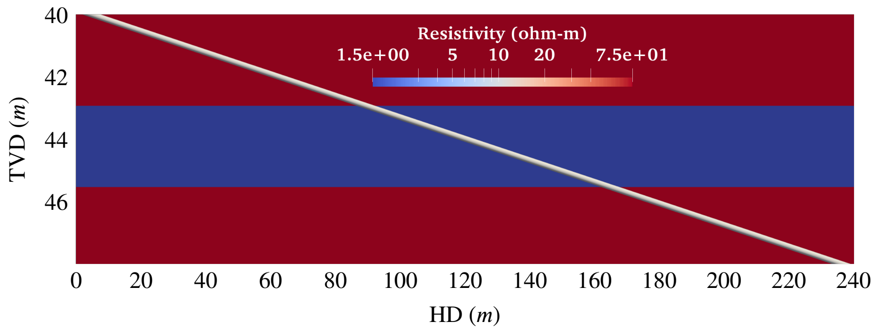


(c) Geosignal measurement

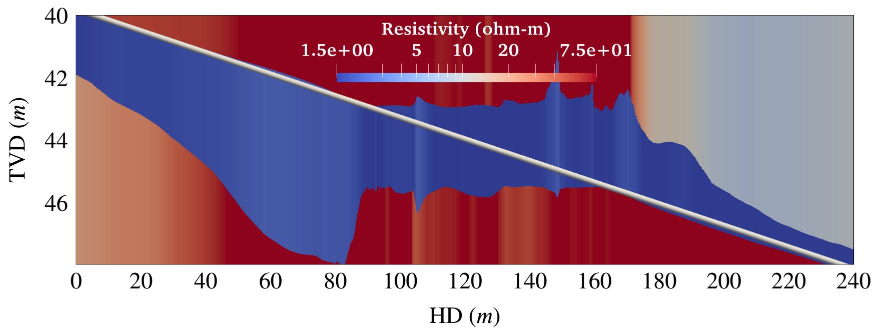
Figure 2.23: Model problem I. Comparison between $\mathcal{F} \circ \mathcal{I}$ and $\mathcal{F}_{\phi*} \circ \mathcal{I}_{\theta*}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.

2.8.3.2 Model Problem II

In this problem, we consider a 2.5m-thick conductive layer surrounded by two resistive layers. A well trajectory with a dip angle equal to 87° crosses the formation. Figure 2.24 displays the original and predicted models by DL. This example illustrates some of the limitations of DNNs. In this case, the Earth models associated with part of the trajectory are outside the model problems considered in Section 2.1, which restrict to only one layer above and below the logging trajectory. Thus, the DNN is untrained for such models, and results cannot be trusted in those zones. Numerical results confirm these observations. Nonetheless, inaccurate inversion results are simple to identify by inspection of the logs (Figures 2.25 and 2.26).

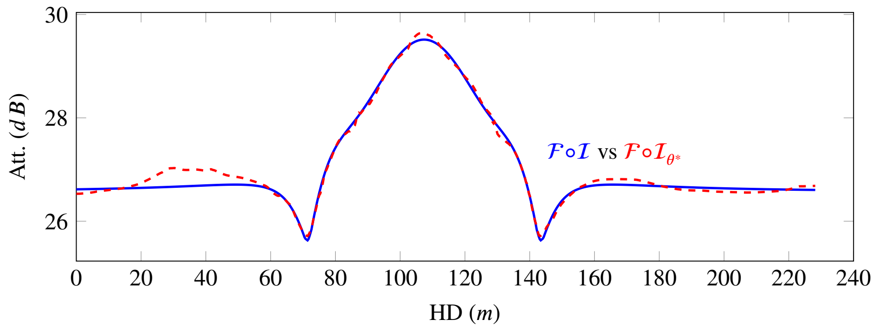


(a) Actual formation

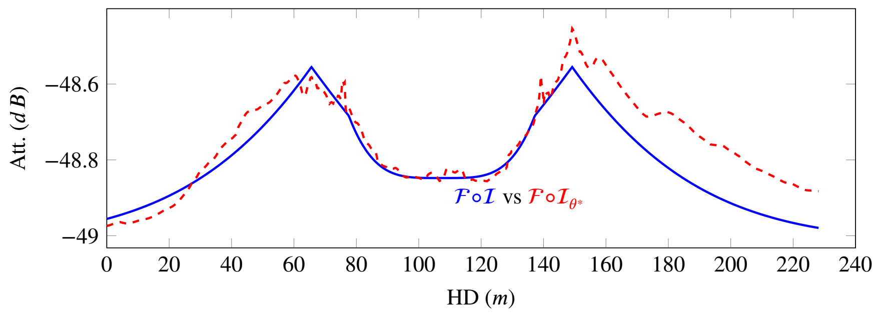


(b) Predicted formation using one logging position with regularization

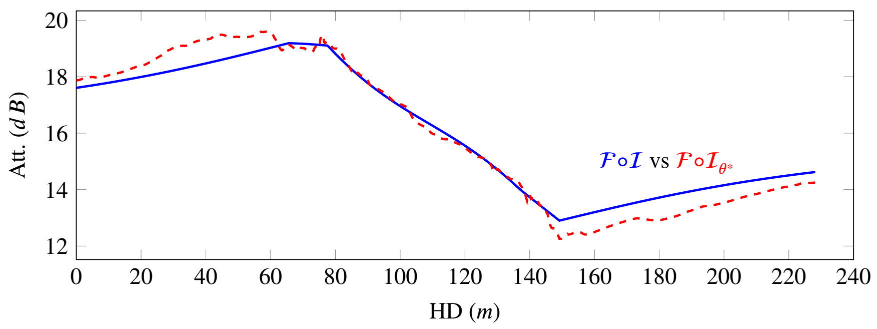
Figure 2.24: Model problem 2. Comparison between actual and predicted formations with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) LWD coaxial measurement

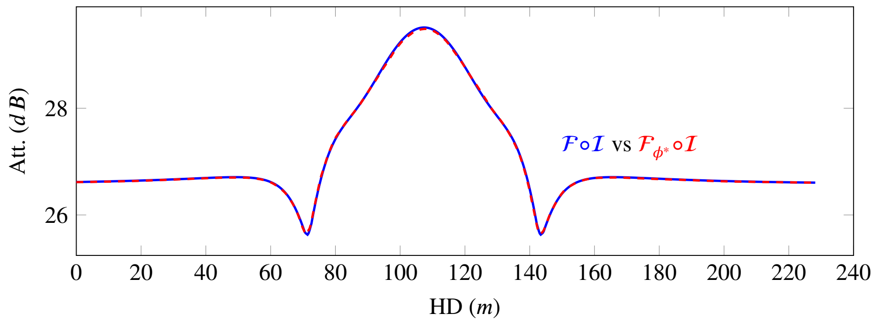


(b) Deep coaxial measurement

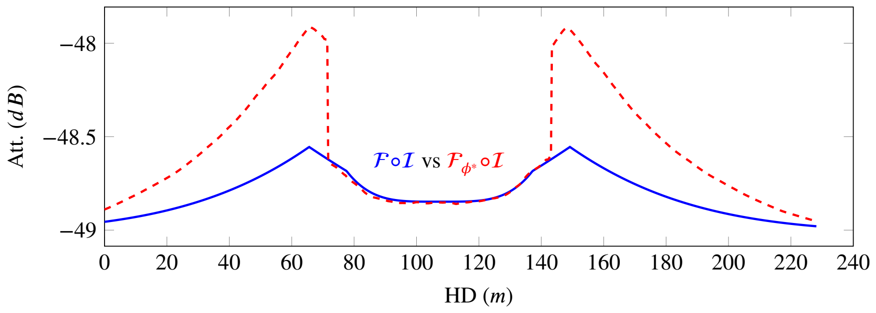


(c) Geosignal measurement

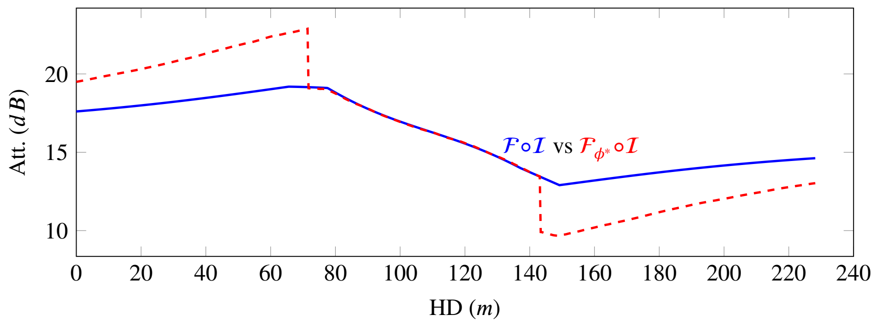
Figure 2.25: Model problem 2. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta*}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) LWD coaxial measurement



(b) Deep coaxial measurement

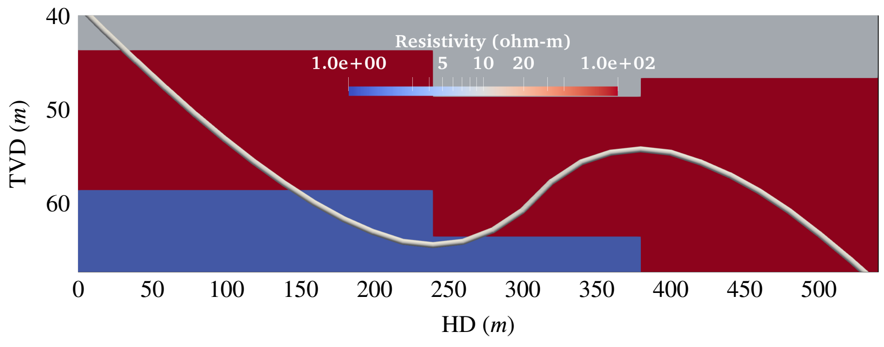


(c) Geosignal measurement

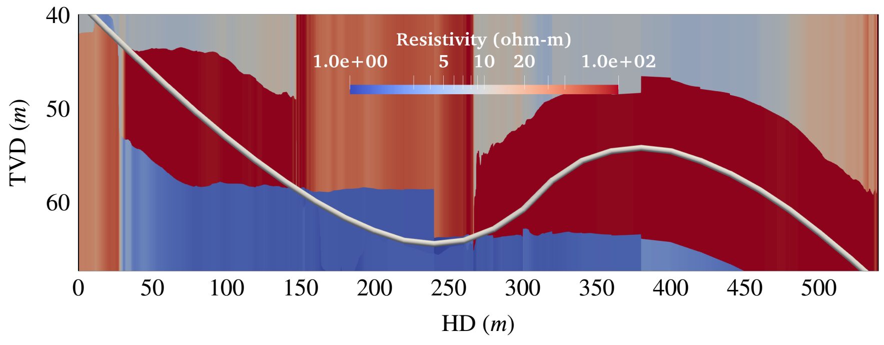
Figure 2.26: Model problem 2. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}_{\phi*}\circ\mathcal{I}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.

2.8.3.3 Model Problem III

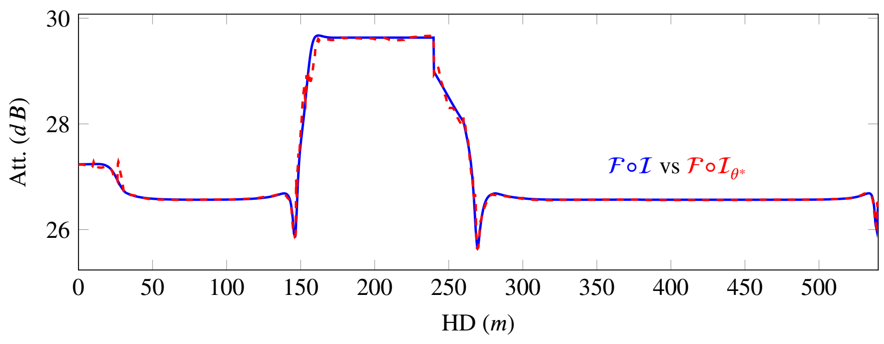
We now consider a model formation exhibiting geological faults and two different well trajectories. For well trajectory 1, Figure 2.27 shows the model problem, logging trajectory, inversion results, and coaxial attenuation logs. Inversion results are excellent. When considering the second well trajectory shown in Figure 2.28, we observe good inversion results except at the proximity of points with horizontal distance (HD) equals to 75m and 350m. These inaccurate inversion results are easily identified by examination of the corresponding logs.



(a) Actual formation

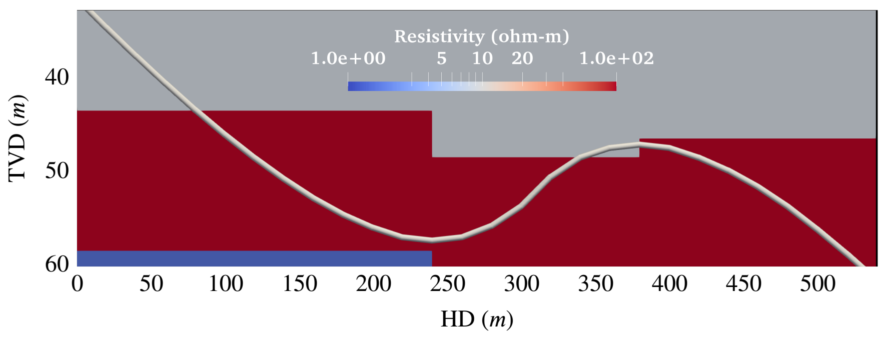


(b) Predicted formation

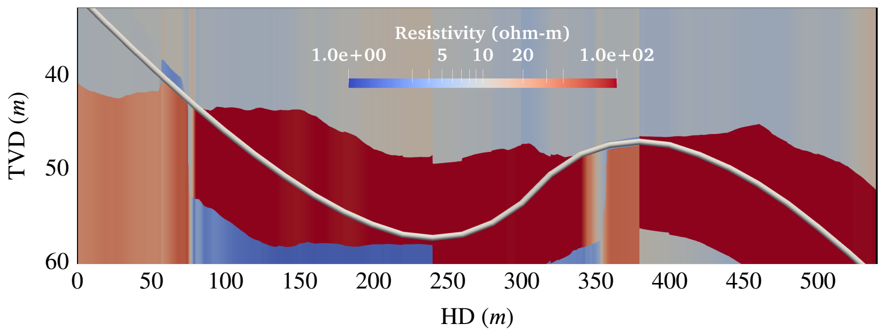


(c) LWD coaxial measurement

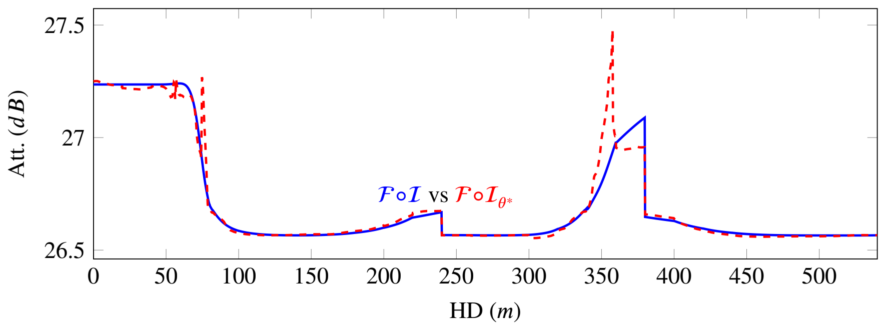
Figure 2.27: Model problem III, trajectory 1. Comparison between actual and predicted formations and the corresponding coaxial logs with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.



(a) Actual formation



(b) Predicted formation



(c) LWD coaxial measurement

Figure 2.28: Model problem III, Trajectory 2. Comparison between actual and predicted formations and the corresponding coaxial logs with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging position per sample.

3 Database Generation using IGA

Deep Learning (DL) methods are fast, but require a massive training dataset. To decrease the online computational time during field operations, we often produce such a large dataset a priori (offline) using tens of thousands of simulations of borehole resistivity measurements (see [75]). To generate the database for DL inversion, we employ simulation methods to solve Maxwell's equations with different conductivity distributions (Earth models). Since 3D simulations are expensive and possibly unaffordable when computing such large databases, it is common to reduce the Earth model dimensionality to two or one spatial dimensions using a Fourier or a Hankel transform. These transformations lead to the so-called 2.5D [3, 48, 92, 101, 135] and 1.5D [12, 100, 129] formulations, respectively. 1.5D simulations are inaccurate when dealing with geological faults.

In this work, we focus on the efficient generation of a massive database using 2.5D simulations – as a preliminary stage for DL inversions. We propose the use of refined Isogeometric Analysis (rIGA) discretizations to generate databases for DL inversion of 2.5D geosteering electromagnetic (EM) measurements.

3.1 2.5D Variational Formulation of Electromagnetic Measurements

3.1.1 3D Wave Propagation Problem

The two time-harmonic curl Maxwell's equations describing the 3D wave propagation in an isotropic medium are

$$\begin{aligned} \nabla \times \mathbf{E} + i\omega\mu\mathbf{H} &= -i\omega\mu\mathbf{M} \\ \nabla \times \mathbf{H} &= (\sigma + i\omega\varepsilon)\mathbf{E} \end{aligned} \tag{3.1}$$

where $\mathbf{E}$ is the electric field, $\mathbf{H}$ is the magnetic field, i is the imaginary unit, σ is the electric conductivity, ε is the electric permittivity, μ is the magnetic permeability, $\omega = 2\pi f$ is the angular frequency, with f being the transmitter frequency, and $\mathbf{M}$ is the time-harmonic magnetic source located at (x_0, y_0, z_0) and given by

$$\mathbf{M} = \delta(x - x_0)\delta(y - y_0)\delta(z - z_0)[M_x, M_y, M_z] \qquad \in \mathbb{R}^3, \tag{3.2}$$

with $\delta(\cdot)$ being the Dirac delta function defined as follows:

$$\delta(x - x_0) := \begin{cases} \infty, & x = x_0, \\ 0, & x \neq x_0. \end{cases} \tag{3.3}$$

To incorporate an integrable approximation of the Dirac delta function, we consider a bell-like representation for the delta function. For example, in the x direction, we approximate:

$$\delta(x - x_0) \approx \frac{1}{\alpha\sqrt{\pi}} \exp\left[-\left(\frac{x - x_0}{\alpha} \right)^2 \right], \tag{3.4}$$

where α is a positive value.

From Maxwell's equations, we obtain the following reduced wave formulation in terms of magnetic field $\mathbf{H}$:

$$\left\{ \begin{array}{lrll} \text{Find } \mathbf{H} = [H_x, H_y, H_z], \text{ with } \mathbf{H} : \boldsymbol{\Omega} \subset \mathbb{R}^3 \to \mathbb{C}^3, \quad \text{such that:} & & & \\ \nabla \times \left(\dfrac{1}{\sigma + i\omega\varepsilon} \nabla \times \mathbf{H} \right) + i\omega\mu\mathbf{H} = & -i\omega\mu\mathbf{M}, & & \text{in } \boldsymbol{\Omega}, \\ \mathbf{E} \times \mathbf{n} = & \mathbf{0}, & & \text{on } \partial\boldsymbol{\Omega}, \end{array} \right. \tag{3.5}$$

where $\boldsymbol{\Omega}$ is the domain of study and $\mathbf{n}$ is the unit normal (outward) vector on the boundary $\partial\boldsymbol{\Omega}$. We define $\boldsymbol{\Omega}$ as a tensor-product box:

$$\boldsymbol{\Omega} = \Omega_x \times \Omega_y \times \Omega_z = \big(-L_x/2, L_x/2 \big) \times \big(-L_y/2, L_y/2 \big) \times \big(-L_z/2, L_z/2 \big),$$

being L_x,L_y, and L_z positive real constants.

To introduce the weak formulation of this problem, we first define the $\boldsymbol{H}(\mathrm{curl}; \boldsymbol{\Omega})$-conforming functional spaces

$$\begin{aligned} \boldsymbol{H}(\mathrm{curl}; \boldsymbol{\Omega}) :=& \quad \{ \mathbf{W} = [W_x, W_y, W_z] \in (L^2(\boldsymbol{\Omega}))^3 : \nabla \times \mathbf{W} \in (L^2(\boldsymbol{\Omega}))^3 \}, \\ \boldsymbol{H}_0(\mathrm{curl}; \boldsymbol{\Omega}) :=& \quad \{ \mathbf{W} \in \boldsymbol{H}(\mathrm{curl}; \boldsymbol{\Omega}) : \mathbf{W} \times \mathbf{n} = \mathbf{0} \text{on } \partial\boldsymbol{\Omega} \}. \end{aligned} \tag{3.6}$$

The $\boldsymbol{H}(\mathrm{curl}; \boldsymbol{\Omega})$ space is endowed with the inner product

$$\begin{aligned} (\mathbf{W}, \mathbf{H})_{\boldsymbol{H}(\mathrm{curl};\boldsymbol{\Omega})} &:= (\nabla \times \mathbf{W}, \nabla \times \mathbf{H})_{(L^2(\boldsymbol{\Omega}))^3} + (\mathbf{W}, \mathbf{H})_{(L^2(\boldsymbol{\Omega}))^3} \\ &:= \int_\Omega (\nabla \times \mathbf{W})^* \cdot (\nabla \times \mathbf{H}) d\boldsymbol{\Omega} + \int_\Omega \mathbf{W}^* \cdot \mathbf{H} d\boldsymbol{\Omega}, \end{aligned} \tag{3.7}$$

where * is the conjugate transpose of complex vector space and $\cdot$ denotes the inner product.

We build the weak formulation by multiplying Eq. (3.5) with an arbitrary function $\mathbf{W} \in \boldsymbol{H}_0(\text{curl}; \boldsymbol{\Omega})$, using Green's formula, and integrating over the domain $\boldsymbol{\Omega}$. The weak formulation is then

$$\left\{ \begin{array}{l} \text{Find } \mathbf{H} \in \boldsymbol{H}_0(\text{curl}; \boldsymbol{\Omega}), \text{ such that for every } \mathbf{W} \in \boldsymbol{H}_0(\text{curl}; \boldsymbol{\Omega}), \\ \left(\nabla \times \mathbf{W}, \dfrac{1}{\sigma + i\omega\varepsilon} \nabla \times \mathbf{H} \right)_{(L^2(\boldsymbol{\Omega}))^3} + i\omega\mu(\mathbf{W}, \mathbf{H})_{(L^2(\boldsymbol{\Omega}))^3} = -i\omega\mu(\mathbf{W}, \mathbf{M})_{(L^2(\boldsymbol{\Omega}))^3} \end{array} \right. \tag{3.8}$$

3.1.2 2.5D Variational Formulation

Herein, we focus on the case when the material properties are homogeneous along one spatial direction, e.g., y-axis. We denote the domain for this case as $\boldsymbol{\Omega} := \Omega_y \times \boldsymbol{\Omega}_{x,z}$. We perform a Fourier transform along the y-axis to represent the 3D problem as a sequence of uncoupled 2D problems, one per Fourier mode. In this case, we define the magnetic field $\mathbf{H}$ as a series expansion using the complex exponentials:

$$\mathbf{H} := \sum_{\beta=-\infty}^{+\infty} \mathbf{H}_\beta \exp(i2\pi\beta y/L_y), \tag{3.9}$$

where β is the Fourier mode number and $\mathbf{H}_\beta = \left[H_x^\beta, H_y^\beta, H_z^\beta\right]$ with $\mathbf{H}_\beta : \boldsymbol{\Omega}_{x,z} \subset \mathbb{R}^2 \to \mathbb{C}^3$.

Fourier modes satisfy the following orthogonality relationships, where $\delta_{i,j}$ is the Kronecker delta:

$$\frac{1}{L_y} \int_{-L_y/2}^{L_y/2} \exp(i2\pi\beta_1 y/L_y)\exp(i2\pi\beta_2 y/L_y) dy = \delta_{\beta_1,\beta_2}. \tag{3.10}$$

By employing a test function of the form

$$\mathbf{W} := \frac{1}{L_y} \mathbf{W}_\beta \exp(i2\pi\beta y/L_y), \tag{3.11}$$

and defining the $\boldsymbol{H}(\text{curl}^\beta; \boldsymbol{\Omega}_{x,z})$-conforming functional spaces

$$\begin{aligned} \boldsymbol{H}(\text{curl}^\beta; \boldsymbol{\Omega}_{x,z}) := \; & \left\{ \mathbf{W}_\beta = [W_x^\beta, W_y^\beta, W_z^\beta] \in (L^2(\boldsymbol{\Omega}_{x,z}))^3 : \mathbf{W}_y^\beta \in H^1(\boldsymbol{\Omega}_{x,z}) \right. \\ & \left. \text{and } \nabla \times [W_x^\beta, W_z^\beta] \in (L^2(\boldsymbol{\Omega}_{x,z}))^2 \right\}, \\ \boldsymbol{H}_0(\text{curl}^\beta; \boldsymbol{\Omega}_{x,z}) := \; & \left\{ \mathbf{W}_\beta \in \boldsymbol{H}(\text{curl}^\beta; \boldsymbol{\Omega}_{x,z}) : \mathbf{W}_\beta \times \mathbf{n} = \mathbf{0} \text{ on } \partial\boldsymbol{\Omega} \right\}. \end{aligned} \tag{3.12}$$

we build the following variational formulation from Eq. (3.5) by integrating over $\mathbf{\Omega}_{x,z}$:

$$\begin{cases} \text{Find } \mathbf{H} = \sum_{\beta=-\infty}^{+\infty} \mathbf{H}_\beta \exp(i2\pi\beta y/L_y), \mathbf{H}_\beta \in \boldsymbol{H}_0(\text{curl}; \mathbf{\Omega}_{x,z}) \\ \text{such that for every } \beta \in \mathbb{Z} \text{ and } \mathbf{W}_\beta \in \boldsymbol{H}_0(\text{curl}; \mathbf{\Omega}_{x,z}), \\ \left(\nabla^\beta \times \mathbf{W}_\beta, \dfrac{1}{\sigma + i\omega\varepsilon} \nabla^\beta \times \mathbf{H}_\beta\right)_{(L^2(\mathbf{\Omega}_{x,z}))^3} + i\omega\mu(\mathbf{W}_\beta, \mathbf{H}_\beta)_{(L^2(\mathbf{\Omega}_{x,z}))^3} = -i\omega\mu(\mathbf{W}_\beta, \mathbf{M}_\beta)_{(L^2(\mathbf{\Omega}_{x,z}))^3}, \end{cases} \tag{3.13}$$

where

$$\nabla^\beta \times \mathbf{W}_\beta := \left[i\beta\frac{2\pi}{L_y} W_z^\beta - \frac{dW_y^\beta}{dz}, \frac{dW_x^\beta}{dz} - \frac{dW_z^\beta}{dx}, \frac{dW_y^\beta}{dx} - i\beta\frac{2\pi}{L_y} W_x^\beta \right], \tag{3.14}$$

and $\mathbf{M}_\beta$ is the time-harmonic magnetic source written in terms of the Fourier transform as

$$\mathbf{M}_\beta = \frac{1}{L_y}\delta(x - x_0)\delta(z - z_0)[M_x, M_y, M_z]\exp(i2\pi\beta y_0/L_y). \tag{3.15}$$

This formulation corresponds to the 2.5D variational formulation previously described by, e.g., [27] and [120].

Remark 3.1. To solve the variational problem of Eq. (3.13), we require an appropriate space in $\mathbf{\Omega}_{x,z}$ over which $\mathbf{W}_\beta$, $\nabla \times [\mathbf{W}_x^\beta, \mathbf{W}_z^\beta]$, and $\nabla \mathbf{W}_y^\beta$ are integrable, i.e., $[\mathbf{W}_x^\beta, \mathbf{W}_z^\beta] \in \boldsymbol{H}(\text{curl}; \mathbf{\Omega}_{x,z})$ and $\mathbf{W}_y^\beta \in H^1(\mathbf{\Omega}_{x,z})$. Thus, we use the $\boldsymbol{H}(\text{curl}^\beta; \mathbf{\Omega}_{x,z})$ solution space – equivalent to the $\boldsymbol{H}(\text{curl}; \mathbf{\Omega}_{x,z}) \times H^1(\mathbf{\Omega}_{x,z})$ mixed space – that fulfills the mentioned requirements.

3.2 Borehole Resistivity Measurement Acquisition System

We consider the logging-while-drilling (LWD) instrument equipped with transmitters (T_{x_i}) and receivers (R_{x_j}) of Figure 3.1. This tool is sensitive to resistivities within the range $0.2 \sim 500\ \Omega \cdot \text{m}$ (phase resistivity) and $0.2\ \sim 300\ \Omega \cdot \text{m}$ (amplitude resistivity) under an operating frequency between 0.1 and 2 MHz [79]. For the sake of simplicity, herein, we restrict to two transmitters and two receivers symmetrically located around the tool center (see Figure 3.1) at an operating frequency of 2 MHz.

Triaxial logging instruments generate measurements for all possible orientations of the transmitter–receiver pairs. We follow the notation presented by [34]

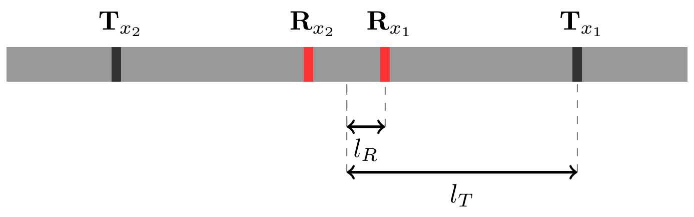


Figure 3.1: A schematic LWD instrument with two transmitters and two receivers located symmetrically around the tool center.

and [120] to denote the magnetic field. Thus, we write $H_{ZZ}^{T_{x_i}R_{x_j}} \in \mathbb{C}$ as the coaxial magnetic field in the borehole system of coordinates induced by transmitter T_{x_i} and measured at receiver R_{x_j} $(i, j = 1, 2)$. We use the magnetic fields measured at R_{x_1} and R_{x_2} to compute the *attenuation ratio* and *phase difference*. We symmetrize the signal originating from T_{x_1} and T_{x_2} to obtain the *quantity of interest* Q_{ZZ} at each logging position as

$$\mathrm{Q}_{ZZ} := \frac{1}{2}\left(\log\frac{H_{ZZ}^{T_{x_1}R_{x_1}}}{H_{ZZ}^{T_{x_1}R_{x_2}}} + \log\frac{H_{ZZ}^{T_{x_2}R_{x_2}}}{H_{ZZ}^{T_{x_2}R_{x_1}}}\right). \tag{3.16}$$

Then, we compute the attenuation ratio (A) and phase difference (P), respectively, as the real and imaginary parts of Q_{ZZ}:

$$A := \mathrm{Re}(\mathrm{Q}_{ZZ}), \tag{3.17}$$

$$P := \mathrm{Im}(\mathrm{Q}_{ZZ}), \tag{3.18}$$

We can then obtain the *apparent resistivities* based on the attenuation ratio and phase difference (ρ_A and ρ_P, respectively) using a *look-up table* algorithm. This algorithm obtains the apparent resistivities from the tool response in a homogeneous isotropic medium, which is analytically known (see [7]).

3.3 Refined Isogeometric Analysis

In this work, we consider a *multi-field* EM problem and discretize the 2.5D variational formulation of Eq. (3.13) using a B-spline generalization of a curl-conforming space, introduced by [20]. We first review some basic concepts of high-continuity Isogeometric Analysis (IGA) discretizations.

3.3.1 High-continuity IGA Discretization

Given the parametric domain $\{\xi, \zeta \in \hat{\mathbf{\Omega}}_{x,z} : (0,1)^2 \subset \mathbb{R}^2\}$, we introduce the spline space $\mathcal{S}^{p_x,p_z}_{k_x,k_z}$ as

$$\mathcal{S}^{p_x,p_z}_{k_x,k_z} := \operatorname{span}\left\{B^{p_x,p_z}_{i,j}\right\}^{n_x-1,n_z-1}_{i=0,j=0}, \tag{3.19}$$

where n, p, and k with their indices are the number of degrees of freedom, polynomial degree, and continuity of basis functions in x and z directions, respectively, resulting in $n_e := n - p$ number of elements in each direction. The bivariate basis functions are

$$B^{p_x,p_z}_{i,j} := B^{p_x}_i(\xi) \otimes B^{p_z}_j(\zeta), \qquad i = 0,1,...,n_x-1, \quad j = 0,1,...,n_z-1, \tag{3.20}$$

where the univariate bases are expressed by the Cox–De Boor recursion formula [107] as

$$B^0_i(\xi) = \begin{cases} 1, & \xi_i \leqslant \xi < \xi_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.21}$$

$$B^p_i(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B^{p-1}_i(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B^{p-1}_{i+1}(\xi), \tag{3.22}$$

and spanned over the respective knot sequences in x and z directions, given by

$$\Xi = [\underbrace{0,0,...,0}_{p_x+1}, \xi_{p_x+1}, \xi_{p_x+2}..., \xi_{n_x-1}, \underbrace{1,1,...,1}_{p_x+1}], \tag{3.23}$$

$$\mathrm{Z} = [\underbrace{0,0,...,0}_{p_z+1}, \zeta_{p_z+1}, \zeta_{p_z+2}..., \zeta_{n_z-1}, \underbrace{1,1,...,1}_{p_z+1}], \tag{3.24}$$

We assume single multiplicities for all knots, providing maximum continuity $k = p - 1$ for the IGA discretization.

Figure 3.2 illustrates the $\boldsymbol{H}(\text{curl}) \times H^1$ IGA discrete space in $\mathbf{\Omega}_{x,z}$ along with the univariate basis functions of the respective vector fields. For brevity, herein and in the following, we exclude the superscript β in referring to the components of the magnetic field.

We define the spaces in the parametric domain and introduce the appropriate transformations to obtain the discretization on the physical domain. We start with the set of discrete spaces in the parametric domain, given by

$$\hat{\mathcal{V}}^{\text{curl}}_h(\hat{\mathbf{\Omega}}_{x,z}) := \mathcal{S}^{p-1,p}_{k-1,k} \times \mathcal{S}^{p,p-1}_{k,k-1}, \tag{3.25}$$

$$\hat{\mathcal{Q}}^{\text{grad}}_h(\hat{\mathbf{\Omega}}_{x,z}) := \mathcal{S}^{p,p}_{k,k}. \tag{3.26}$$

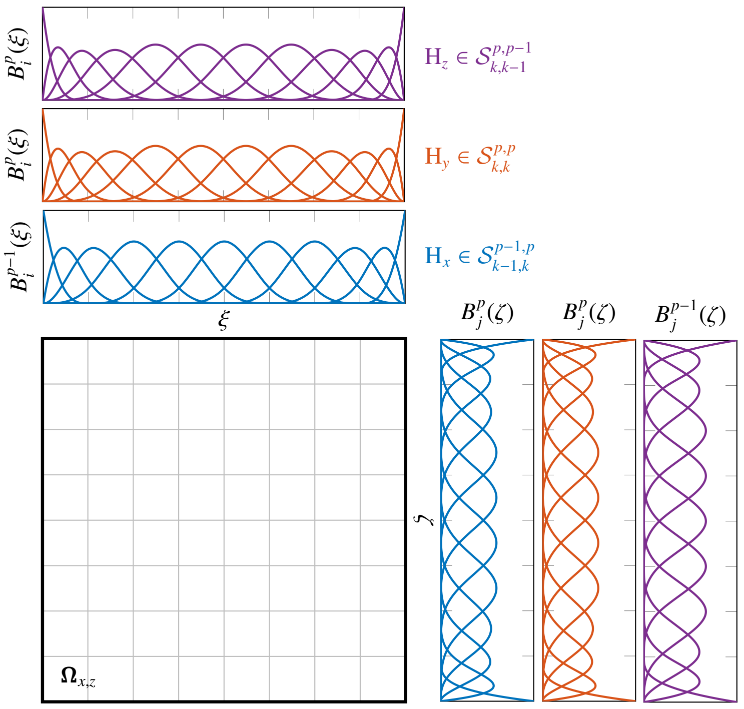

$B_i^p(\xi)$
$\mathrm{H}_z \in \mathcal{S}^{p,p-1}_{k,k-1}$
$B_i^p(\xi)$
$\mathrm{H}_y \in \mathcal{S}^{p,p}_{k,k}$
$B_i^{p-1}(\xi)$
$\mathrm{H}_x \in \mathcal{S}^{p-1,p}_{k-1,k}$
ξ
$B_j^p(\zeta)$
$B_j^p(\zeta)$
$B_j^{p-1}(\zeta)$
ζ
$\mathbf{\Omega}_{x,z}$

Figure 3.2: Example of the $\boldsymbol{H}(\mathrm{curl}) \times H^1$ space for a 2.5D formulation discretized by C^{p-1} IGA with uniform 8×8 elements in $\mathbf{\Omega}_{x,z}$, polynomial degree $p = 4$, and continuity $k = 3$. The univariate basis functions of H_x, H_y, and H_z are shown in blue, red, and purple, respectively. Thin gray lines in the mesh skeleton denote the high-continuity element interfaces.

By defining $\mathbf{F} : \hat{\mathbf{\Omega}}_{x,z} \to \mathbf{\Omega}_{x,z}$ as the geometric mapping from the parametric domain onto the physical domain, and $D\mathbf{F}$ as its Jacobian, we introduce the set of discrete spaces in the physical domain:

$$\mathcal{V}_h^{\mathrm{curl}}(\mathbf{\Omega}_{x,z}) := \left\{ \mathbf{H}_{x,z} = [H_x, H_z] \in \boldsymbol{H}(\mathrm{curl}; \mathbf{\Omega}_{x,z}) \cap \boldsymbol{H}^1(\mathbf{\Omega}_{x,z}) : \iota^{\mathrm{curl}}(\mathbf{H}_{x,z}) = \hat{\mathbf{H}}_{x,z} \in \hat{\mathcal{V}}_h^{\mathrm{curl}}(\hat{\mathbf{\Omega}}_{x,z}) \right\}, \tag{3.27}$$

$$\mathcal{Q}_h^{\mathrm{grad}}(\mathbf{\Omega}_{x,z}) := \left\{ H_y \in H^1(\mathbf{\Omega}_{x,z}) : \iota^{grad}(H_y) = \hat{H}_y \in \hat{\mathcal{Q}}_h^{\mathrm{grad}}(\hat{\mathbf{\Omega}}_{x,z}) \right\} \tag{3.28}$$

where we use the following curl- and grad-preserving pullback mappings [20, 46]:

$$\iota^{\mathrm{curl}}(\mathbf{H}_{x,z}) := (D\mathbf{F})^T (\mathbf{H}_{x,z} \circ \mathbf{F}), \tag{3.29}$$

$$\iota^{\mathrm{grad}}(H_y) := H_y \circ \mathbf{F}. \tag{3.30}$$

Thus, by defining the discrete space

$$\mathcal{H}_{h,0}(\mathbf{\Omega}_{x,z}) := \left\{ \mathbf{H}_{\beta,h} \in \mathcal{V}_h^{curl}(\mathbf{\Omega}_{x,z}) \times \mathcal{Q}_h^{grad}(\mathbf{\Omega}_{x,z}) : \mathbf{H}_{\beta,h} \times \mathbf{n} = \mathbf{0} \text{ on } \partial\mathbf{\Omega} \right\}, \quad (3.31)$$

we write the discrete form of Eq.(3.13) as follows (subscript h refers to discrete solution):

$$\left\{ \begin{array}{l} \text{Find } \mathbf{H}_h = \sum_{\beta=-\infty}^{+\infty} \mathbf{H}_{\beta,h} \exp(i2\pi\beta y/L_y), \mathbf{H}_{\beta,h} \in \mathcal{H}_{h,0}(\mathbf{\Omega}_{x,z}) \\ \text{such that for every } \beta \in \mathbb{Z} \text{ and } \mathbf{W}_{\beta,h} \in \mathcal{H}_{h,0}(\mathbf{\Omega}_{x,z}), \\ \left(\nabla^\beta \times \mathbf{W}_{\beta,h}, \dfrac{1}{\sigma + i\omega\varepsilon} \nabla^\beta \times \mathbf{H}_{\beta,h} \right)_{(L^2(\mathbf{\Omega}_{x,z}))^3} + i\omega\mu(\mathbf{W}_{\beta,h}, \mathbf{H}_{\beta,h})_{(L^2(\mathbf{\Omega}_{x,z}))^3} = -i\omega\mu(\mathbf{W}_{\beta,h}, \mathbf{M}_\beta)_{(L^2(} \end{array} \right. \quad (3.32)$$

3.3.2 rIGA Discretization

The rIGA is a discretization technique that optimizes the performance of direct solvers. In particular, rIGA preserves the optimal convergence order of the direct solvers with respect to a fixed number of elements in the domain. [47] first presented this strategy for H^1 spaces and then extended it to $\boldsymbol{H}$(curl), $\boldsymbol{H}$(div), and L^2 spaces (see [46]). Starting from the high-continuity C^{p-1} IGA discretization, rIGA reduces the continuity of certain basis functions by increasing the multiplicity of the respective existing knots. Hence, the computational domain is subdivided into high-continuity macroelements interconnected by low-continuity hyperplanes. These hyperplanes coincide with the locations of the *separators* at different partitioning levels of the multifrontal direct solvers. Thus, rIGA reduces the computational cost of matrix factorization when solving Partial Differential Equation (PDE) systems in comparison to IGA and Finite Element Method (FEM).

For multi-field problems discretized using $\boldsymbol{H}(\text{curl}; (\mathbf{\Omega}_{x,z})) \times H^1(\mathbf{\Omega}_{x,z})$ spaces, we preserve the commutativity of the *de Rham* diagram [37] by reducing the continuity in $k-1$ degrees. To achieve this, we use both C^0 and C^1 hyperplanes and reduce the continuity across the interface between the subdomains (i.e., macroelements). Figure 3.3 depicts the rIGA discretization of the $\boldsymbol{H}(\text{curl}; \mathbf{\Omega}_{x,z}) \times H^1(\mathbf{\Omega}_{x,z})$ space of Figure 3.2 after one level of symmetric partitioning, which results in macroelements containing 4×4 elements.

Previous works show rIGA discretizations provide significant improvements in the solution time and memory requirements. In particular, the rIGA solution is obtained up to $\mathcal{O}(p^2)$ faster in large domains – and $\mathcal{O}(p)$ faster in small domains – than the IGA solution. In comparison to traditional FEM with the same number

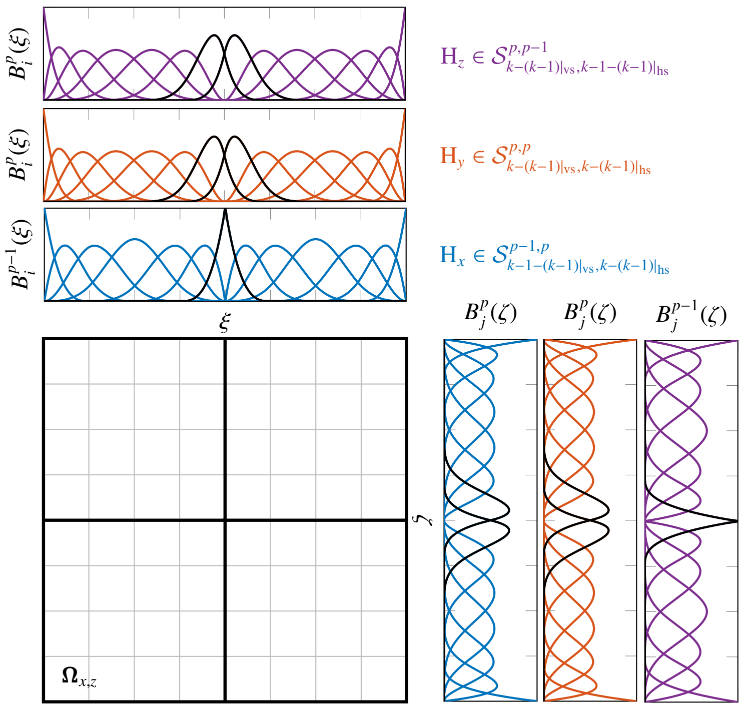

$B_i^p(\xi)$
$B_i^p(\xi)$
$B_i^{p-1}(\xi)$
$\mathrm{H}_z \in \mathcal{S}^{p,p-1}_{k-(k-1)|_{\mathrm{vs}},k-1-(k-1)|_{\mathrm{hs}}}$
$\mathrm{H}_y \in \mathcal{S}^{p,p}_{k-(k-1)|_{\mathrm{vs}},k-(k-1)|_{\mathrm{hs}}}$
$\mathrm{H}_x \in \mathcal{S}^{p-1,p}_{k-1-(k-1)|_{\mathrm{vs}},k-(k-1)|_{\mathrm{hs}}}$
ξ
$B_j^p(\zeta)$
$B_j^p(\zeta)$
$B_j^{p-1}(\zeta)$
ζ
$\boldsymbol{\Omega}_{x,z}$

Figure 3.3: $\boldsymbol{H}(\text{curl}) \times H^1$ rIGA space in $\boldsymbol{\Omega}_{x,z}$, associated with the 8×8 domain of Figure 3.2 with $p = 4$ and $k = 3$, after one level of symmetric partitioning by the rIGA discretization that results in 4×4 macroelements. rIGA reduces the continuity of basis functions by $k - 1$ degrees across the macroelement separators (the low-continuity bases are shown in black). Thin gray lines in the mesh skeleton denote the high-continuity element interfaces, while thick black lines illustrate the macroelement boundaries. We refer to the vertical and horizontal separators as "vs" and "hs", respectively.

of elements, rIGA provides even larger improvements. rIGA also reduces the memory requirements since the rIGA LU factors have fewer nonzero entries than the IGA LU factors. Finally, rIGA improves the approximation error with respect to IGA since the continuity reduction of basis functions enriches the Galerkin space (see [46, 47]).

3.4 Implementation Details

We implement discrete $\boldsymbol{H}(\text{curl}; (\boldsymbol{\Omega}_{x,z})) \times H^1(\boldsymbol{\Omega}_{x,z})$ spaces using PetIGA-MF [124], a multi-field extension of PetIGA [33], a high-performance isogeometric analysis implementation based on PETSc (portable extensible toolkit for scientific computation) [13]. PetIGA-MF allows the use of different spaces for each field of interest and employs data management libraries to condense the data of multiple fields in a single object, thus simplifying the discretization construction. This framework also allows us to investigate both IGA and rIGA discretizations in our 2.5D problem with different numbers of elements, different polynomial degrees of the B-spline spaces, and different partitioning levels of the mesh.

We use Intel MKL with PARDISO [105, 106] as our sparse direct solver package to construct LU factors for solving the linear systems of equations. PARDISO employs supernode techniques to perform the matrix factorization)see, e.g., [126, 127]. It provides parallel factorization using OpenMP directives [32] and uses the automatic matrix reordering provided by METIS [70]. We executed all tests on a workstation equipped with two Intel Xeon Gold 6230 CPUs at 2.10 GHz with 40 threads per CPU.

We employ a tensor-product mesh with variable element sizes (see Figure 3.4). At each logging position, the computational mesh has a fine subgrid in the central part of the domain with element size equal to $h \times h$. This subgrid is surrounded by another tensor-product grid whose element sizes grow slowly until reaching the boundary. Let n_e be the number of elements in each direction and $n_c < n_e$ is the number of elements in each direction located at the central part of the domain. We use the power function of Eq.(3.33) to model the geometrical progression of the mesh. We follow the algorithm presented by [120] to find suitable values for n_c and growth rate r. Starting from the center point of a symmetric domain, we obtain the size of the i-th element in each direction h_i ($i = 1, 2, ..., n_e/2$) as follows:

$$h_i = \begin{cases} h, & 1 \leqslant i \leqslant n_c/2, \\ hr^{(i-n_c/2)}, & n_c/2 < i \leqslant n_e/2. \end{cases} \tag{3.33}$$

Remark 3.2. For each logging position, we perform a single *symbolic* factorization common to all Fourier modes, followed by a numerical factorization per Fourier mode. Once we solve the system of equations for the first transmitter, we update the right-hand side of Eq. (3.32) to solve for the magnetic field induced by the second transmitter and use backward substitution. Hence, we perform only one LU factorization per Fourier mode per logging position for both transmitters.

Remark 3.3. The convergence of the Fourier series leads to a fast decay of the real and imaginary parts of H_{ZZ} for higher Fourier modes (see [119]). Thus, we truncate the series of Eq. (3.9) when the magnetic field at the receivers is sufficiently small, such that $\beta \in [-N_f, N_f]$, being $2N_f + 1$ the total number of

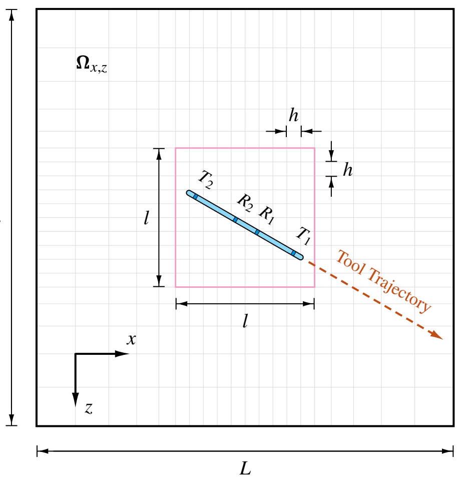


Figure 3.4: A drawing of the computational domain $\mathbf{\Omega}_{x,z}$ and the tool trajectory. The central subgrid bounded by a magenta box is composed of a set of fine elements located in the proximity of the logging instrument. The remaining elements grow smoothly in size until reaching the boundary.

Fourier modes. Due to the symmetry of the media along the y direction, we only consider $\beta \in [0, N_f]$.

3.5 Numerical Results

In this section, we first assess the accuracy of the rIGA approach in a homogeneous medium. We also investigate the computational efficiency of the rIGA framework in comparison with IGA and FEM approaches. Then, we consider two model problems consisting of high-angle wells crossing spatially heterogeneous media with multiple geological faults. Finally, we produce our synthetic training dataset as a preliminary stage for DL inversion. In our simulations, we consider one operational mode of a commercial logging tool [158] with $l_R = 10.16$ cm and $l_T = 56.8325$ cm (see Figure3.1). We select the free space electric permittivity and magnetic permeability as $\varepsilon = 8.85 \times 10^{-12}$ F $\cdot\, m^{-1}$ and $\mu = 4\pi \times 10^{-7}$ N $\cdot$ A^{-2}, respectively. We also consider a transmitter frequency $f = 2$ MHz.

3.5.1 Homogeneous Medium

We assume the logging instrument is placed in a homogeneous medium with resistivity $\rho = 1/\sigma = 100\ \Omega \cdot$ m. This high-resistivity case is numerically more challenging than low-resistivity cases since it requires a larger number of Fourier modes and numerical precision. We consider a cube domain of length $L = 18$ m for our first case study.

3.5.1.1 Accuracy Assessment

To assess the accuracy and select certain discretization parameters, we compare the numerical attenuation ratio, A, and phase difference, P, given by Eq.(3.17) and Eq.(3.18), with the expected (i.e., exact) values, A_e and P_e, obtained from $\rho_e = 1/\sigma$. Figure 3.5 shows the numerical errors, i.e., $|1 - A/A_e|$ and $|1 - P/P_e|$, as a function of the number of Fourier modes when computing attenuation ratio and phase difference in a homogeneous medium. Herein, we select a domain with 64×64 elements to ensure a fast numerical solution for our measurements. We compare the results of the high-continuity C^{p-1} IGA with FEM and also with a rIGA discretization that employs 8×8 macroelements. This macroelement size provides the fastest results for moderate size domains (see [47]). We consider three different mesh sizes – $h = 0.025$ m, 0.033 m, and 0.050 m– and different polynomial degrees –$p = 3,\ 4,$ and 5. The best results correspond to $h = 0.025$ m (blue lines in the figure). We also observe that rIGA and FEM discretizations deliver lower errors compared to their IGA counterparts –when the same number of elements and polynomial degree are considered– taking into account that rIGA provides solutions with higher computational efficiency (see Section 3.5.1.2).

To investigate the decay of the solution for each Fourier mode, we compare numerical results with the analytical 2.5D solution in the homogeneous medium presented by [120]. In particular, given M_z as the only nonzero component of the magnetic source, it is possible to analytically determine the coaxial magnetic field for each Fourier mode as follows:

$$\mathrm{H}_{ZZ}(\beta) = -i\omega\mu\sigma\tau_{\beta_z} + \frac{\partial^2 \tau_{\beta_z}}{\partial z^2}, \tag{3.34}$$

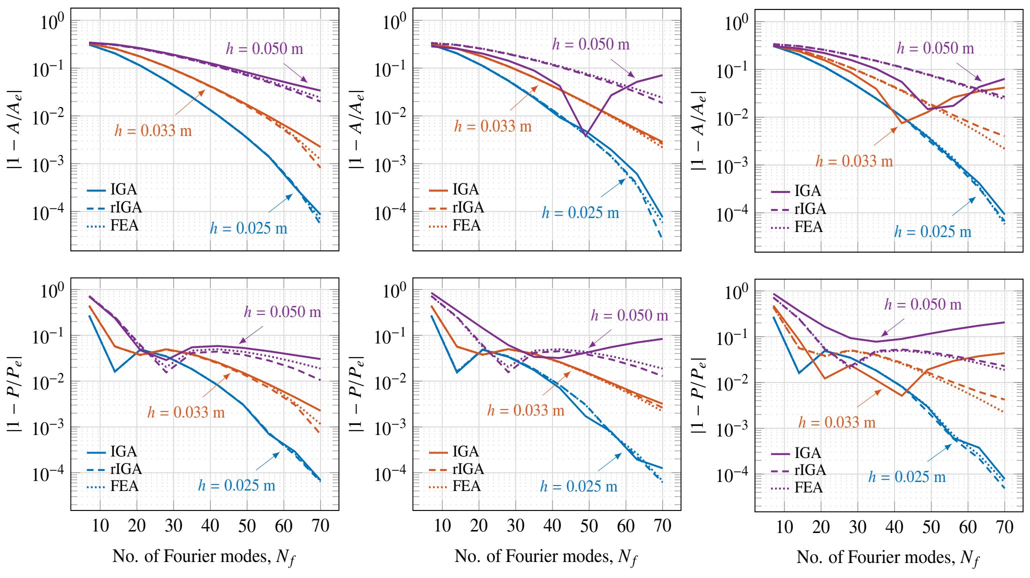
with

$$\tau_{\beta_z} = \frac{M_z}{2\pi}\frac{1}{L_y}K_0(CR)\exp(i2\pi\beta y_0/L_y), \tag{3.35}$$

where $K_0(\cdot)$ is the modified Bessel function of the second kind of order zero, and

$$C = (2\pi\beta/L_y)^2 + i\omega\mu\sigma, \tag{3.36}$$

$$R = \sqrt{(x - x_0)^2 + (z - z_0)^2}. \tag{3.37}$$



(a) $p = 3$ (b) $p = 4$ (c) $p = 5$

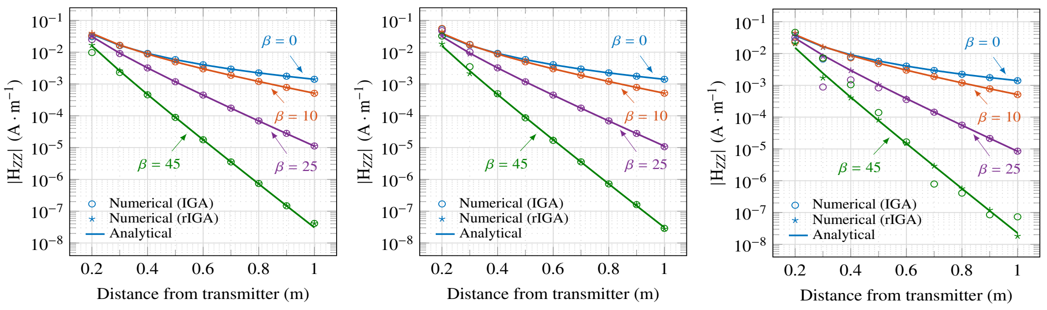
Figure 3.5: Numerical errors when computing attenuation ratio, A, and phase difference, P, in a homogeneous medium using IGA and rIGA discretizations, obtained by a 64×64 element mesh with different element sizes h and polynomial degrees p.

Figure 3.6 compares the decay of the numerical coaxial magnetic field $H_{ZZ}(\beta)$ with its analytical counterpart for some Fourier modes. Using a domain with 64×64 elements and $h = 0.025$ m, we monitor the decay of the propagated waves at distances within the interval $[0.2, 1.0]$ m from the transmitters to ensure that the solutions at both receivers properly approximate the analytical ones. Results show that rIGA discretizations deliver increased accuracy for all tested polynomial degrees.

3.5.1.2 Computational Efficiency

[46, 47] provide theoretical cost estimates of solving H^1 and $\boldsymbol{H}(\text{curl})$ discrete spaces, respectively. Herein, we add these estimates to predict the cost of discretizing the $\boldsymbol{H}(\text{curl}; \boldsymbol{\Omega}_{x,z}) \times H^1(\boldsymbol{\Omega}_{x,z})$ space appearing in our 2.5D EM problem. We conclude that the cost of LU factorization of the rIGA matrix for this combined space is between $\mathcal{O}(p)$ and $\mathcal{O}(p^2)$ times smaller than that for IGA. Details are omitted for the sake of simplicity.

To numerically assess the computational efficiency confirming the aforementioned theoretical results, we consider two different grids in $\boldsymbol{\Omega}_{x,z}$ with 64×64 and 128×128 elements, respectively. Using continuity reduction, we split the

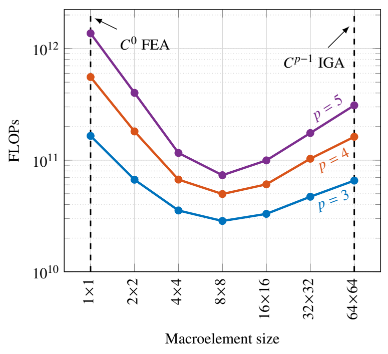


(a) $p = 3$ (b) $p = 4$ (c) $p = 5$

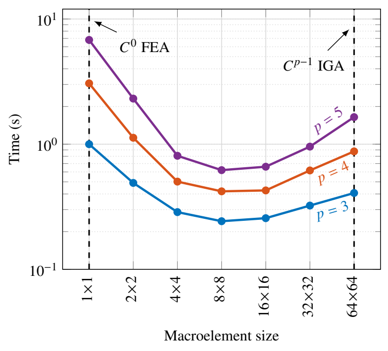
Figure 3.6: Comparison of the decay of the numerical and analytical coaxial magnetic fields for some Fourier modes, obtained in a grid of 64×64 elements with $h = 0.025$ m and different polynomial degrees.

mesh symmetrically into macroelements whose sizes are powers of two. In this context, the maximum-continuity C^{p-1} IGA discretization is composed of one macroelement containing the entire grid, while C^0 FEM with minimum continuity across all element interfaces is composed of macroelements that contain only one element. Figure 3.7 shows the number of FLOPs and time required to solve the borehole resistivity problem for each Fourier mode per logging position. We compare the computational costs for different polynomial degrees and different continuity reduction levels of basis functions. The cost of rIGA reaches the minimum with 8×8 macroelements almost in all cases, confirming the theoretical estimates obtained from the results of [47].

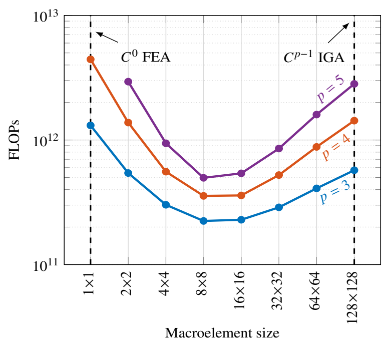
Our numerical tests show that for a moderate size 2.5D problem, the reduction in the number of FLOPs is $\mathcal{O}(p)$ with respect to IGA. When compared to FEM, rIGA delivers larger improvement factors. These improvement factors in terms of FLOPs also hold in terms of time when performing a sequential factorization. In our parallel PARDISO solver, we observe a small degradation of the rIGA improvement factors in terms of times in comparison to those obtained in terms of FLOPs (see Table 3.1).



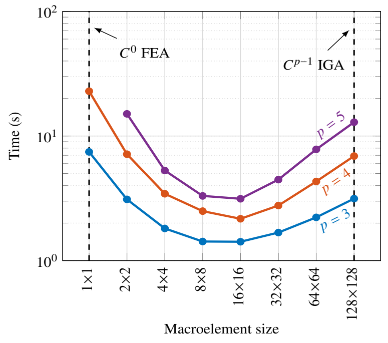
(a) FLOPs (64 × 64 elements)



(b) Time (64 × 64 elements)



(c) FLOPs (128 × 128 elements)



(d) Time (128 × 128 elements)

Figure 3.7: Computational cost in terms of FLOPs and time for solving a 2.5D borehole resistivity problem per logging position per Fourier mode. We test rIGA discretizations with two different grids of 64 × 64 and 128 × 128 elements. The computational times correspond to the use of parallel solver PARDISO using two threads.

Table 3.1: Computational cost for the 2.5D borehole resistivity measurements per logging position per Fourier mode. We report the solution time and FLOPs when using C^{p-1} IGA, rIGA with 8×8 macroelements, and C^0 FEM with the same number of elements and polynomial degree. The computational times correspond to the use of parallel solver PARDISO using two threads.

Domain size	Polynomial degree	Discretization method	Number of FLOPs	Improvement factor (FLOPs)		Time (s)	Improvement factor (time)	
		IGA	6.56e+10			0.407		
	3	rIGA	2.85e+10	IGA/rIGA	2.30	0.242	IGA/rIGA	1.68
		FEA	1.65e+11	FEA/rIGA	5.79	0.999	FEA/rIGA	4.12
		IGA	1.62e+11			0.875		
64×64	4	rIGA	4.97e+10	IGA/rIGA	3.26	0.419	IGA/rIGA	2.09
		FEA	5.56e+11	FEA/rIGA	11.19	3.061	FEA/rIGA	7.29
		IGA	3.10e+11			1.645		
	5	rGA	7.33e+10	IGA/rIGA	4.23	0.620	IGA/rIGA	2.65
		FEA	1.37e+12	FEA/rIGA	18.69	6.806	FEA/rIGA	10.98
		IGA	5.72e+11			3.144		
	3	rIGA	2.24e+11	IGA/rIGA	2.55	1.423	IGA/rIGA	2.21
		FEA	1.31e+12	FEA/rIGA	5.85	7.456	FEA/rIGA	5.24
		IGA	1.43e+12			6.903		
128×128	4	rIGA	3.56e+11	IGA/rIGA	4.02	2.495	IGA/rIGA	2.77
		FEA	4.44e+12	FEA/rIGA	12.47	22.885	FEA/rIGA	9.17
		IGA	2.82e+12			12.911		
	5	rGA	4.97e+11	IGA/rIGA	5.67	3.305	IGA/rIGA	3.91
		FEA	–	FEA/rIGA	–	–	FEA/rIGA	–

3.5.2 Heterogeneous Media

We further examine the accuracy of our rIGA approximation over two synthetic heterogeneous model problems.

3.5.2.1 One Geological Fault

We consider the model problem of Figure 3.8 with a constant dip angle of 80°. We consider the Logging While Drilling (LWD) instrument described in Section 3.2 and simulate measurements recorded over 200 equally-spaced logging positions throughout the well trajectory.

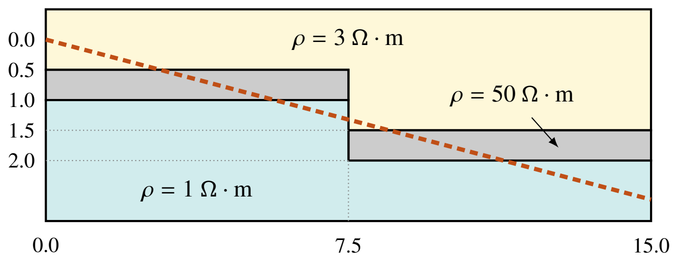

$\rho = 3\,\Omega \cdot \mathrm{m}$
$\rho = 50\,\Omega \cdot \mathrm{m}$
$\rho = 1\,\Omega \cdot \mathrm{m}$
0.0
0.5
1.0
1.5
2.0
0.0
7.5
15.0

Figure 3.8: Model problem with a constant dip angle of 80° passing through a geological fault and three different materials (well trajectory is highlighted by a red dashed line). Dimensions are in meters.

Figure 3.9 shows the apparent resistivities based on the attenuation ratio and phase difference (ρ_A and ρ_P, respectively). We obtain the results using $N_f = 70$ and a rIGA discretization with 64×64 elements, $p = 4$, and 8×8 macroelements. Results are in good agreement with those presented by [120].

3.5.2.2 Two Geological Faults and Inclined Layers

Figure 3.10 shows the second model problem containing two geological faults and inclined layers. The logging trajectory starts from a sandstone layer with a resistivity of $\rho = 3\ \Omega \cdot \mathrm{m}$, and passes through an oil-saturated layer with $\rho = 100\ \Omega \cdot \mathrm{m}$. The tool trajectory also passes through a water-saturated layer with $\rho = 0.5\ \Omega \cdot \mathrm{m}$.

In particular, inclined layers produce the so-called *staircase* approximations [25]. This phenomenon occurs because the physical interfaces of the conductivity model are not aligned with the element edges. Thus, the conductivity parameter takes different values inside some elements of the mesh. To tackle this issue, discretization techniques using nonfitting grids [26, 27] are available, but they have not been considered here for simplicity.

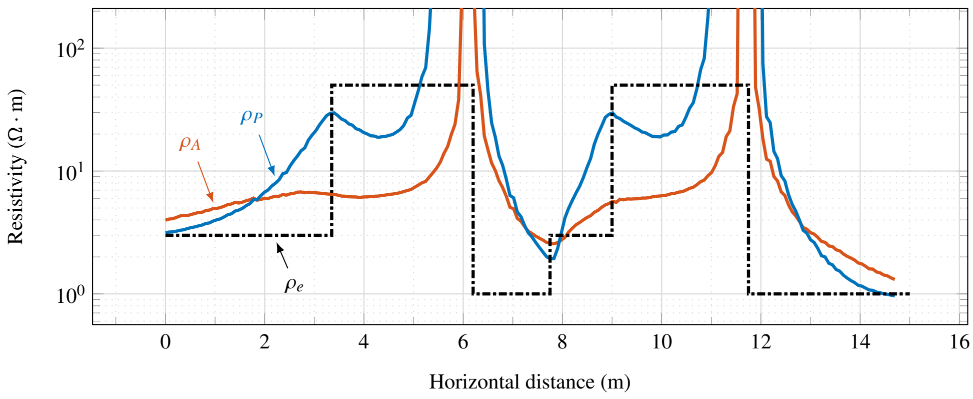


Figure 3.9: Apparent resistivities based on the attenuation ratio, ρ_A, and phase difference, ρ_P, for the first model problem, compared with the real (exact) resistivity, ρ_e. We obtain the results using a rIGA discretization with 64 × 64 elements, $p = 4$, and 8 × 8 macroelements.

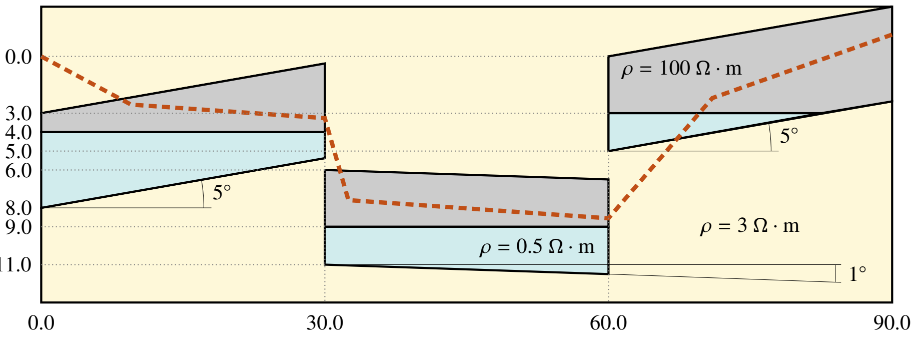


Figure 3.10: Second model problem with two geological faults and inclined layers. The tool trajectory (red dashed line) has different dip angles and passes through sandstone (yellow), oil-saturated (gray), and water-saturated (green) layers. Dimensions are in meters.

Figure 3.11 shows the apparent resistivities based on the attenuation ration and phase difference throughout the logging trajectory and compares their value with the exact resistivity. We simulate the resistivities at 1,080 logging positions with $N_f = 70$. We use a rIGA discretization with 64 × 64 elements, $p = 4$, and 8 × 8 macroelements.

3.5.3 Database Generation for DL Inversion

To produce our synthetic training dataset for DL inversion, we consider heterogeneous medium containing three different layers and six varying parameters at

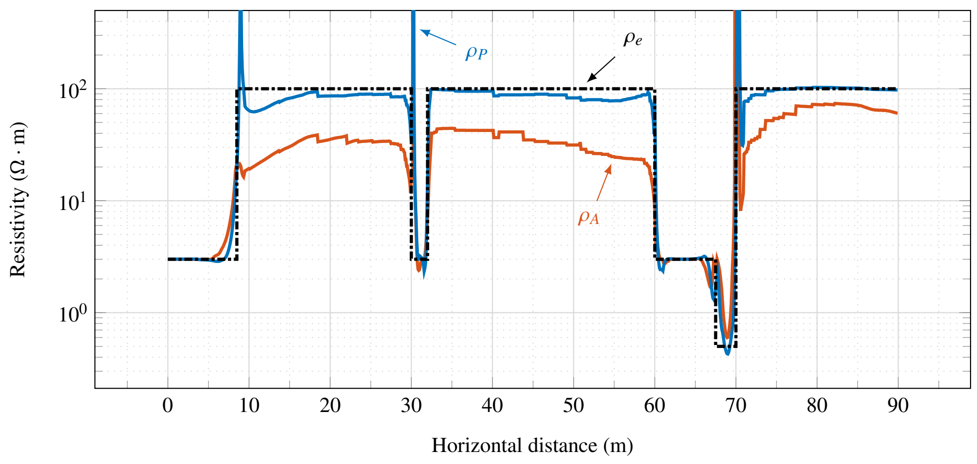


Figure 3.11: Apparent resistivities based on the attenuation ratio, ρ_A, and phase difference, ρ_P, for the second model problem, compared with the real (exact) resistivity, ρ_e. We employ a rIGA discretization with 64×64 elements, $p = 4$, and 8×8 macroelements.

each logging position, as described in Figure 3.12 and Table 3.2. We select three different electrical conductivities: σ_c for the central layer, and σ_u and σ_l for the upper and lower layers, respectively. We assume the tool center is always within the middle layer and has vertical distances of d_u and d_l from the upper and lower layers, respectively. The sixth varying parameter is the dip angle, φ, measured from the vertical direction.

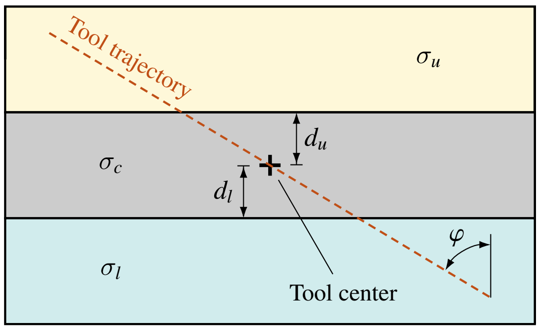


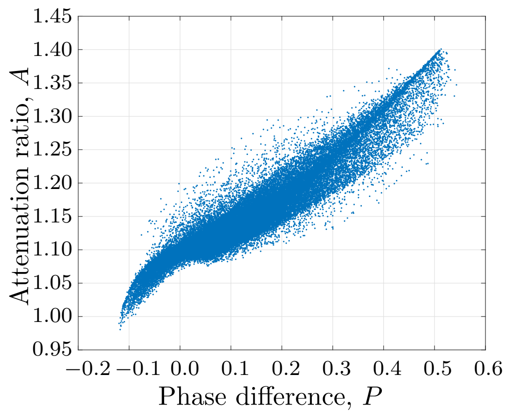
Figure 3.12: Varying parameters at each logging position when producing the training dataset for DL inversion.

In here, we create a dataset of 100,000 samples and compute the apparent resistivities obtained from random combinations within a given range of resistivities $\rho = 1/\sigma \in [1, 100]\ \Omega \cdot \text{m}$ (see Table 3.2). For generating the dataset, we use two different types of parallelization. One parallelization is related to the parallel

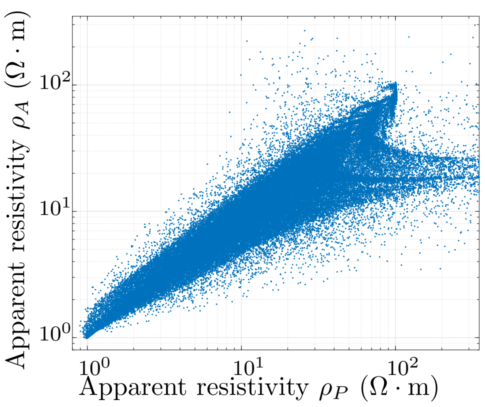
Table 3.2: Varying parameters employed to generate the training dataset for DL inversion.

Varying parameters		Interval
Electrical conductivity of the central layer	$\log_{10}(\sigma_c)$	$[-2, 0]$
Electrical conductivity of the upper layer	$\log_{10}(\sigma_u)$	$[-2, 0]$
Electrical conductivity of the lower layer	$\log_{10}(\sigma_l)$	$[-2, 0]$
Distance of the tool center from the upper layer	$\log_{10}(d_u)$	$[-2, 1]$
Distance of the tool center from the lower layer	$\log_{10}(d_l)$	$[-2, 1]$
Dip angle between the tool and the layered media	φ	$[80^\circ, 100^\circ]$

factorization of the direct solver, and the other is the trivial parallelization based on scheduling the solutions of independent Earth models onto different processors. Using 40 threads, we solve for 20 different Earth models, each executing over two threads. Table 3.1 shows that the required time for matrix factorization of the 2.5D EM problem using optimal rIGA discretization with 64×64 grid, $p = 4$, and 8×8 macroelements is about 0.42 seconds per Fourier mode. Considering $N_f = 70$, and the additional time required for pre/postprocessing and inter-thread communications, each set of independent runs (consists of 20 different Earth models) takes about 40 seconds. Thus, we perform 5,000 sequential runs to construct our 100,000 samples in about 56 hours. To create a larger database, we could execute over a cluster of hundreds of CPUs/threads, expecting a perfect parallel scalability. Figure 3.13a depicts the graphs of attenuation ratio, A, versus phase difference, P, obtained from the 100,000 Earth models when using rIGA discretization for generating the database. Since there is a strong correlation between A and P, the data distribution on the plot follows an almost straight line. We also display in Figure 3.13b the correlation between apparent resistivities based on attenuation ratio and phase difference.



(a)



(b)

Figure 3.13: (a) Attenuation ratio vs. phase difference, and (b) apparent resistivity based on attenuation vs. apparent resistivity based on phase, obtained for the 100,000 Earth models. We use rIGA discretization with 64×64 elements, $p = 4$, and 8×8 macroelements for generating the database.

3.6 Discussions

Herein, we discuss some of the topics observed during the course of this research about using IGA and rIGA for database generation for DL:

FEM vs. IGA and rIGA When using FEM, the solution space is characterized by basis functions that have support over up to two elements in each spatial direction. Considering n_e as the number of elements in each direction, p as the polynomial degree of basis functions, and d as the space dimension, we deal with a linear system of equations that has $\mathcal{O}\left((n_e p)^d\right)$ unknowns. Whereas in IGA, each basis function is spanned over $p + 1$ elements. As a result, not only we have smoother solution space using IGA (because of higher continuity of basis functions), but also the system of equations has lower number of unknowns, namely, $\mathcal{O}\left((n_e + p)^d\right)$, resulting in cheaper computations compared to FEM. In practice, if we want to reduce the FEM computational cost, we may decrease either the degree p or the number of elements n_e. Either of these deteriorates the solution accuracy. Herein, we propose to use the “refined IGA” (rIGA). Thus, we conserve the desirable properties of IGA while reducing the solution cost of direct solvers by decreasing the interconnection of elements in system matrices.

Different types of parallelization One may consider different types of parallelization, e.g., along the Fourier modes, parallel factorization using parallel direct solvers, and the trivial parallelization based on independent Earth models. In this work, we consider the latter two: parallelizations related to the parallel factorization, and scheduling the solutions of independent Earth models onto different processors. Since each new Earth model entails a new symbolic factorization, the advantage of using these parallelizations lies in the fact that we can use one symbolic factorization for all Fourier modes associated with the same processor.

2D Earth models for database generation Using more detailed 2D Earth models do not affect the computational costs, but may affect the solution accuracy if we deal with geological faults that are not aligned with the computational grid. Thus, we need to align them or use some kind of numerical strategy, e.g., by using nonfitting grids [26, 27], in order to improve the accuracy.

4 Quadrature Rules when Solving PDEs using Deep Learning

In traditional methods for solving inversion problems (such as gradient based or statistics based methods), we need to solve the forward problem multiple times. Moreover, to solve an inverse problem using Deep Learning (DL) techniques we need to create a database, and to do so we also need to solve the forward problem – governed by Partial Differential Equations (PDEs) – thousand of times.

Computational cost is one of the main limitations that traditional methods to solve PDEs have (such as Finite Element Method (FEM), Finite Difference Method (FDM) or Isogeometric Analysis (IGA)). The high computational cost of using these methods thousands of times makes them inadequate for inversion problems. As an alternative, we propose the use of DL techniques to solve parametric PDEs.

In the last decades, the use of DL techniques for solving PDEs has grown exponentially. One of the most popular methods used to carry out this task is based on Physics-Informed Neural Networks (PINNs) [22, 50, 65, 86, 87, 112, 122, 136, 153]. Here, we approximate the solution of a PDE via a Neural Network (NN) that contains the physical information of the problem (more precisely, the loss function contains information about the PDE that governs the problem and the boundary conditions). There exist multiple PINN-based approaches, including: Variational Physics-Informed Neural Networkss (VPINNs) [71], VarNets [73], hp-VPINNs [72], or Deep Galerkin Method (DGM) [139].

While solving PDEs using NNs, we have realized that there exists a problem when computing the integral that appears in the loss function. We must select the quadrature rule to approximate the integral carefully. Otherwise, overfitting may occur, which will result in a disastrous approximation of the PDE solution.

In this Chapter, we solve a simple 1D problem governed by Poisson's equation using the Deep Ritz Method (DRM) [149]. We show the problems that appear due to the inadequate selection of the quadrature rule and propose several alternatives to overcome this limitation.

4.1 Model Problems

Let $\Omega \subset \mathbb{R}$ be a computational domain, and Γ_D and Γ_N two disjoint sections of its boundary, where $\Gamma_D \cup \Gamma_N = \partial\Omega$ and the subscripts D and N denote the Dirichlet and Neumann bounds, respectively. We consider the following boundary value problem:

$$\begin{cases} -u'' = f & x \in \Omega, & (4.1a) \\ u = 0 & x \in \Gamma_D, & (4.1b) \\ u' \cdot \mathbf{n} = g & x \in \Gamma_N. & (4.1c) \end{cases}$$

In the above, $\mathbf{n}$ is the unit normal outward (to the domain) vector, and we assume the usual regularity assumptions, namely, $f \in L^2(\Omega)$, $g \in H^{-1/2}(\Gamma_N)$, and $u \in V = H_0^1(\Omega) = \{v \in H^1(\Omega) \text{ and } v|_{\Gamma_D} = 0\}$, where $H^1(\Omega) = \{v \in L^2(\Omega), v' \in L^2(\Omega)\}$.

In this work we solve two different model problems to illustrate our numerical results.

4.1.1 Model Problem 1

The solution of model problem 1 is $u(x) = x^{0.7}$ and it satisfies Eq. (4.2).

$$\begin{cases} -u''(x) &=& 0.21x^{-1.3} & x \in (0, 10), \\ u(0) &=& 0, \\ u'(10) &=& \frac{0.7}{10^{0.3}}. \end{cases} \tag{4.2}$$

We select this problem because its solution exhibits a singularity (the derivatives of $u(x)$ equal to infinity) at $x = 0$.

4.1.2 Model Problem 2

The solution of model problem 2 is $u(x) = x^2$ and it satisfies Eq. (4.3).

$$\begin{cases} u''(x) &=& 2 & x \in (0, 10), \\ u(0) &=& 0, \\ u'(10) &=& 20. \end{cases} \tag{4.3}$$

This problem has a smooth ($\mathcal{C}^\infty$) solution.

4.2 Loss Functions

We introduce the following standard $L^2(\Omega)$ inner products:

$$(u, v) = \int_\Omega u\, v \qquad \text{and} \qquad (g, v)_{\Gamma_N} = \int_{\Gamma_N} g\, v. \tag{4.4}$$

In the following, we consider two methods: the Ritz Method [116], and the Least Squares Method [91].

4.2.1 Ritz Method

Multiplying the PDE from Eq. (4.1a) by a test function $v \in V$ (where $V = H_0^1(\Omega)$), integrating by parts and incorporating the boundary conditions, we arrive at the variational formulation:

$$\text{Find } u \in V \text{ such that } (u', v') = (f, v) + (g, v)_{\Gamma_N} \qquad \forall v \in V. \tag{4.5}$$

To introduce the Ritz method, we define the energy function $\mathcal{F}_R : V \longrightarrow \mathbb{R}$ given by

$$\mathcal{F}_R(v) = \frac{1}{2}(v', v') - (f, v) - (g, v)_{\Gamma_N}. \tag{4.6}$$

And we define to the following energy minimization problem:

$$u = \arg\min_{v \in V} \mathcal{F}_R(v). \tag{4.7}$$

Theorem 4.1. Problems (4.1) and (4.7) are equivalent [69].

Proof. First, we show (4.1) $\Rightarrow$ (4.5). Then, we prove (4.5) $\Leftrightarrow$ (4.7). To close the equivalences, we see (4.5) $\Rightarrow$ (4.1). Finally, we show that the solution is unique.

(4.1) $\Rightarrow$ (4.5)

We multiply the PDE from (4.1) by an arbitrary test function $v \in V$ and we integrate over Ω

$$-(\Delta u, v) = (f, v). \tag{4.8}$$

We integrate the left hand side by parts to obtain

$$(\nabla u, \nabla v) = (f, v) + (g, v)_{\Gamma_N} \qquad \forall v \in V. \tag{4.9}$$

(4.5) $\Rightarrow$ (4.7)

Let u solve (4.5) and $v \in V$. We set $w = v - u \in V$. We have

$$\begin{aligned} F(v) = F(u+w) &= \frac{1}{2}(\nabla u + \nabla w, \nabla u + \nabla w) - (f, u+w) + (g, u+w)_{\Gamma_N} = \\ &= \frac{1}{2}(\nabla u, \nabla u) - (f,u) + (\nabla u, \nabla w) - (f,w) + \frac{1}{2}(\nabla w, \nabla w) \\ &+ (g,u)_{\Gamma_N} + (g,w)_{\Gamma_N}. \end{aligned} \tag{4.10}$$

Since u solves (4.5), we obtain that

$$\frac{1}{2}(\nabla u, \nabla u) - (f,u) + \frac{1}{2}(\nabla w, \nabla w) + (g,u)_{\Gamma_N}. \tag{4.11}$$

Taking in care that $(\nabla w, \nabla w) \geqslant 0$, we conclude that

$$F(v) \geqslant \frac{1}{2}(\nabla u, \nabla u) - (f,u) + (g,u)_{\Gamma_N} = F(u). \tag{4.12}$$

<u>(4.7) ⇒ (4.5)</u>

Let u solve (4.7). Then, for any $v \in V$ and $\epsilon \in \mathbb{R}$ we have

$$F(u) \leqslant F(u + \epsilon v) =: G(\epsilon). \tag{4.13}$$

Thus, the differentiable function $G(\epsilon)$ has a minimum at $\epsilon = 0$. Hence $\nabla G(0) = 0$, i.e:

$$\nabla G(0) = (\nabla u, \nabla v) - (f,v) + (g,v)_{\Gamma_N} = 0 \qquad \forall v \in V. \tag{4.14}$$

So, u is a solution of (4.5).

<u>(4.5) ⇒ (4.1)</u>

We assume that $u \in V$ satisfies (4.5)

$$(\nabla u, \nabla v) = (f,v) + (g,v)_{\Gamma_N} \Rightarrow \int_\Omega \nabla u \nabla v dx - \int_\Omega f v dx - \int_{\Gamma_N} v(g \cdot \boldsymbol{n}) dx = 0, \forall v \in V. \tag{4.15}$$

We can integrate the first term by parts, obtaining

$$\int_{\Gamma_D} v(\nabla u \cdot \boldsymbol{n}) dx + \int_{\Gamma_N} v(g \cdot \boldsymbol{n}) dx - \int_\Omega v \Delta u dx - \int_\Omega f v dx - \int_{\Gamma_N} v(g \cdot \boldsymbol{n}) dx = 0, \forall v \in V. \tag{4.16}$$

Simplifying terms we reach to

$$-\int_\Omega v(\Delta u + f)dx = 0, \forall v \in V. \tag{4.17}$$

With the assumption that $(\Delta u + f)$ is continuous that equality can only be satisfied if

$$(\Delta u + f)(x) = 0 \qquad x \in \Omega, \tag{4.18}$$

and this means that u is solution of (4.1).

To finish the proof, we show that the solution of (4.5) is unique. Suppose that $u_1 \in V$ and $u_2 \in V$ are solutions of (4.5),

$$\begin{aligned}(\nabla u_1, \nabla v) &= (f, v) + (g, v)_{\Gamma_N} \qquad \forall v \in V,\\ (\nabla u_2, \nabla v) &= (f, v) + (g, v)_{\Gamma_N} \qquad \forall v \in V.\end{aligned} \tag{4.19}$$

Subtracting the equations from (4.19) and selecting $v = u_1 - u_2$ we obtain

$$\int_\Omega (\nabla u_1 - \nabla u_2)^2 dx = 0, \tag{4.20}$$

which implies that

$$\nabla u_1(x) - \nabla u_2(x) = \nabla(u_1 - u_2)(x) = 0 \qquad \forall x \in \Omega. \tag{4.21}$$

This implies that $(u_1 - u_2)(x)$ is constant on Ω, and with the boundary condition $u_1(x) = 0$ for $x \in \Gamma_D$ we reach to $u_1(x) = u_2(x)$, $\forall x \in \Omega$.

□

4.2.2 Least Squares Method

Reordering the terms of Eq. (4.1a) and (4.1c), we define:

$$\begin{cases} \mathcal{G}u := u'' + f & x \in \Omega,\\ \mathcal{B}u := u' \cdot \mathbf{n} - g & x \in \Gamma_N. \end{cases} \tag{4.22}$$

To introduce the Least Squares method, we define the function $\mathcal{F}_{LS} : V \longrightarrow \mathbb{R}$, where the function $v \in V$ satisfies the Dirichlet conditions:

$$\mathcal{F}_{LS}(v) = |(\mathcal{G}v, \mathcal{G}v)| + |(\mathcal{B}v, \mathcal{B}v)_{\Gamma_N}|. \tag{4.23}$$

We want to minimize the function $\mathcal{F}_{LS}(v)$ subject to the essential (Dirichlet) Boundary Conditions (BCs). We often find the minimum by taking the derivative equal to zero and ending up with a linear system of equations. In the context of

DL, we can simply introduce the above loss function $\mathcal{F}_{LS}(v)$ directly in our NN. Therefore, we want to find

$$u = \underset{v \in V}{\arg\min}\, \mathcal{F}_{LS}(v). \tag{4.24}$$

4.3 Neural Network Implementation

We train a NN, named $u_{NN}(x;\theta)$, with the following architecture. We define the trainable part of our NN with learnable parameters θ. We call it u_θ. It is composed by:

1. An input layer. This layer receives the data in the form of a $n \times d$ matrix, where n is the number of samples and d is the dimension of the data.
2. One hidden dense layer with m neurons and a *sigmoid* activation function.
3. An output layer that delivers u_θ.

Then, we add non-trainable layers to our arquitecture in order to impose Equations (4.6) or (4.23). For that, we introduce:

4. A non-trainable layer to impose the Dirichlet boundary conditions. For that, we select a function $\phi(x)$ that satisfies the Dirichlet conditions of the problem and its value is nonzero everywhere else [76]. In this work, we select the following $\phi(x)$ functions for 1D problems in the interval $\Omega = [a, b]$:

   $$\phi(x) = \prod_{x_D \in \Gamma_D} (x - x_D). \tag{4.25}$$

   Then, we generate a new output of the NN: $u_{NN}(x;\theta) = \phi(x)u_\theta(x)$ that strongly imposes the homogeneous Dirichlet boundary conditions.
5. A non-trainable layer to compute the loss function $\mathcal{F}_R$ or $\mathcal{F}_{LS}$ following Eqs. (4.6) or (4.23). Within this layer, we evaluate the integrals and the derivatives. We consider different quadrature rules, being the quadrature points part of the input data of our NN, along with the physical points of the domain. For computing the derivatives, we use automatic differentiation, except in some specific cases, where we employ FDM. These cases are explicitly indicated throughout the text.

Figure 4.1 shows a schematic graph of the described NN architecture. Our software is developed in Python and we use the library *Tensorflow 2.0*.

To train the NN, we replace in Equations (4.7) or (4.24) the search space V by the manifold generated by our learnable parameters θ included in our NN. The result of the minimization is a function $u_{NN}(x, \tilde{\theta})$, where $\tilde{\theta}$ are the optimal learnable parameters encountered as a result of the training. For simplicity, in the following we abuse notation and use the symbol u_{NN} to denote also the solution $u_{NN}(x, \tilde{\theta})$ of our minimization problem.

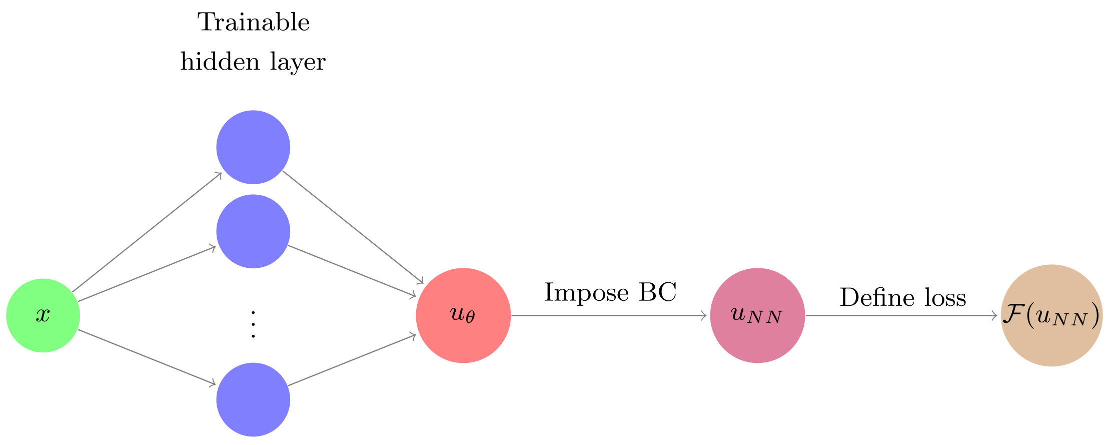


Figure 4.1: Sketch of the arquitecture of u_{NN}.

4.4 Quadrature Rules

We approximate our integrals from Eqs. (4.6) and (4.23) using a quadrature rule of the form

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} \omega_i f(x_i), \tag{4.26}$$

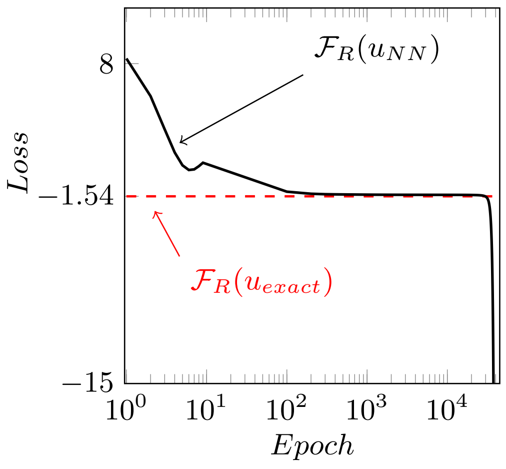
where ω_i are the weights and x_i are the quadrature points. Examples of quadrature rules that follow the above formula include trapezoidal rule and Gaussian quadrature rules [90]. We classify these quadrature rules into two groups: (1) those that only employ points from the interior of the interval; and (2) those that evaluate the solution at one extreme point or more (a or b). Integration rules within the later group (e.g., the trapezoidal rule) are inadequate for our minimization problems because the integrand can be infinite at the boundary points in the case of singular solutions (e.g. model problem 1). Thus, we focus on quadrature rules that only evaluate the solution at interior points of the domain, with a special focus on Gaussian quadrature rules.

4.4.1 Illustration of Quadrature Problems in Neural Networks

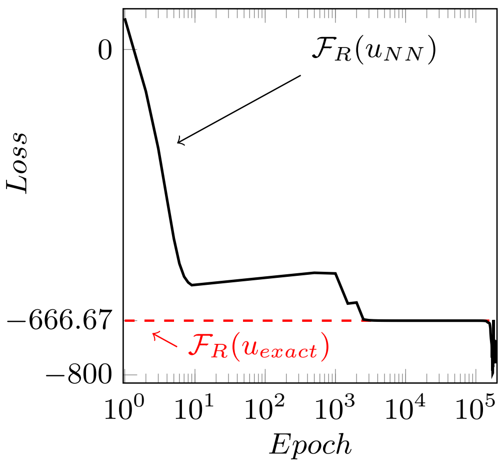
4.4.1.1 Ritz Method

We consider the two model problems from Section 4.1. We approximate $u(x)$ using the Ritz method. Thus, we search for a NN that minimizes the loss functional given by Eq. (4.6). Our NN has one hidden layer with 10 neurons (31 trainable weights). We use automatic differentiation to compute the derivatives and a three-point Gaussian quadrature rule to approximate the integrals within each element. We select the Stochastic Gradient Descent (SGD) optimizer. For model problem 1, we discretize our domain with four equal-size elements and execute $40,000$ iterations during the optimization process. For model problem 2, we discretize our domain with ten equal-size elements and execute $200,000$ iterations during the optimization process.

Figures 4.2a and 4.2b describe the loss evolution of the training process. We obtain a lower loss than the optimum loss computed analytically using the exact solution (i.e., $\mathcal{F}_R(u_{exact})$). This has to be due to some numerical error, in this case, quadrature errors.



(a) Model problem 1.



(b) Model problem 2.

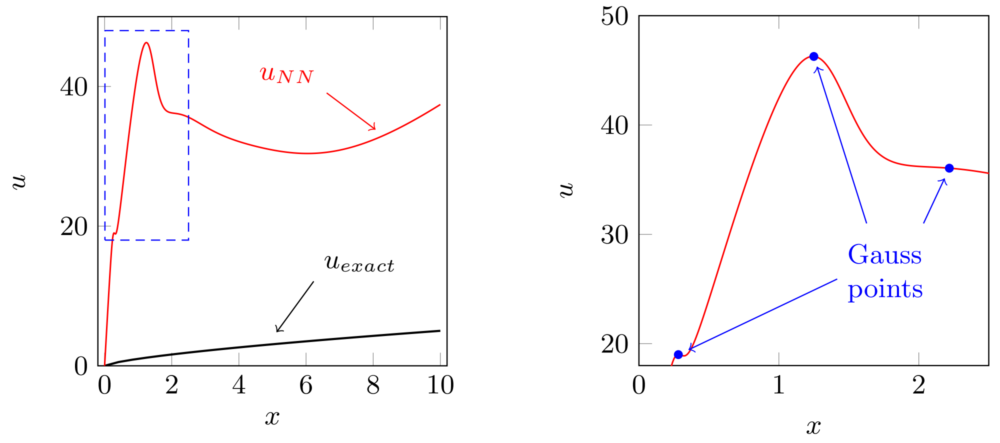
Figure 4.2: Loss evolution of the training process for our two model problems.

Figures 4.3 and 4.4 compare the approximate and exact solutions. We observe a disastrous NN approximation due to quadrature errors. Figures 4.3b and 4.4b show that the gradient is (almost) zero at the training (quadrature) points of the first interval. Therefore, this value minimizes the numerical approximation of

(u', u'). This behavior allows the approximated solution to reach larger values in the first interval, and consequently maximizing the term (f, u), and minimizing the total loss:

$$\mathcal{F}_R(u_{NN}) = \frac{1}{2} \underbrace{(u'_{NN}, u'_{NN})}_{\sum_{q_i} \omega_i (u'_{NN})^2 \approx 0} - \underbrace{(f, u_{NN})}_{\approx \infty} - (g, u_{NN})_{\Gamma_N} \simeq -\infty$$

The described quadrature errors can be interpreted as overfitting over the derivative of the solution.



(a) Exact and approximate solutions.

(b) Approximate solution in the interval $[0, 2.5]$. The Gauss quadrature points corresponding to the first element are indicated in blue.

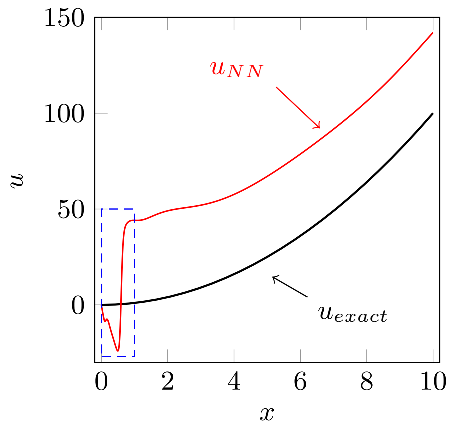
Figure 4.3: Exact vs approximate Ritz method solutions of model problem 1 using four elements for evaluating $\mathcal{F}_R(v)$ and a NN with 31 weights.

4.4.1.2 Least Squares Method

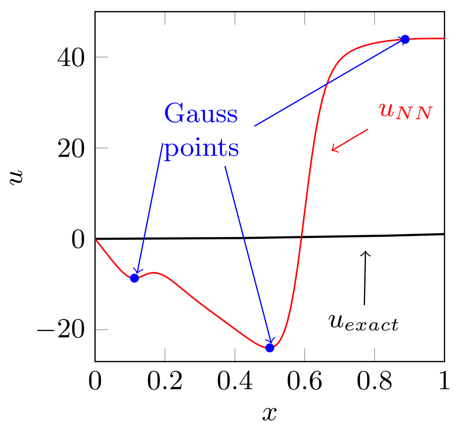
We now consider the following one-dimensional problem:

$$\begin{cases} -u''(x) = 0 & x \in (0, 1), \\ u(0) = u'(1) = 0, & \end{cases} \tag{4.27}$$

where the exact solution is $u(x) = 0$. We can easily construct an approximating function u_{NN} that satisfies Eq. (4.27) at the three considered Gaussian points and minimizes Eq. (4.23), while still being a poor approximation of the exact solution due to quadrature errors. Figure 4.5 shows an example.



(a) Exact and approximate solutions.



(b) Approximate solution in the interval $[0, 1]$. The Gauss quadrature points corresponding to the first element are indicated in blue.

Figure 4.4: Exact vs approximate Ritz method solutions of model problem 2 using ten elements for evaluation of $\mathcal{F}_R(v)$ and a NN with 31 weights.

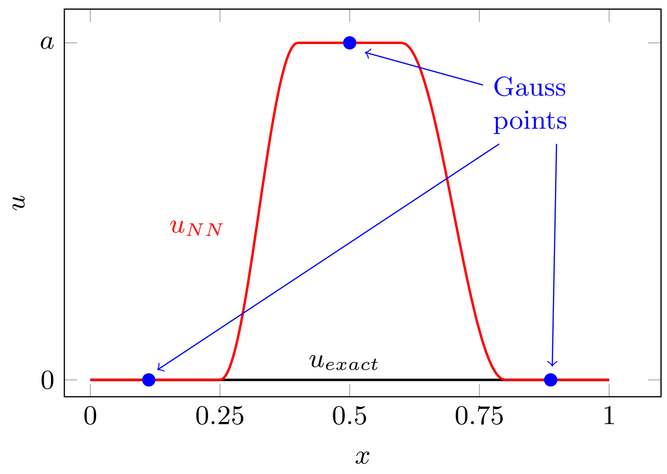


Figure 4.5: Exact ($u_{exact} = 0$) and approximated solution of a problem given by Eq. (4.27) and solved with the Least Square (LS) method.

4.5 Integral Approximation

We now describe four different methods to improve the integral approximations.

4.5.1 Monte Carlo Integration

We consider the following Monte Carlo integral approximation over a set of points $x_i \in (a, b)$,

$$\int_a^b f(x)dx \approx \frac{(b-a)}{n}\sum_{i=1}^{n} f(x_i), \quad x_i \in (a,b) \ \forall i = \{1, \cdots, n\} \tag{4.28}$$

In the above, points x_i are randomly selected [1]. While this method is useful for high-dimensional integrals, for low dimensions (1D, 2D, 3D) the computational cost is high since the value of the integral approximation converges as $1/\sqrt{n}$ [150].

4.5.2 Piecewise-polynomial Approximation

We replace the original NN u_{NN} by a piecewise-polynomial approximation $u^*_{NN,\cdot}$, where $\cdot$ represents the number of pieces of our piecewise-linear interpolator. This approximation can be exactly differentiated (e.q., via FDM) and integrated (via a Gaussian quadrature rule). Figure 4.6 shows an example when we train a NN and we build a piecewise-linear approximation of the NN with four elements.

This method controls quadrature errors. However, it is inadequate for high-dimensional problems as we need a mesh that is difficult to implement and integration becomes time consuming.

4.5.3 Adaptive Integration

We first consider a training dataset over the interval (a, b) by taking an equidistant partition of n elements. Then, we define the validation set as a global h-refinement of the training dataset. Figure 4.7 shows an example of a training and the corresponding validation datasets. Then, for each element of the training mesh (e.g., E_1 in Figure 4.7), we compare the numerical integral over that element vs the sum of the integrals over the two corresponding elements on the validation dataset (in our case, $E_1^1 + E_1^2$). If the integral values differ by more than a stipulated tolerance, we h-refine the training element, and we upgrade the validation dataset so it is built as a global h-refinement of the training dataset. This process is described in Algorithm 1. Figure 4.7 also shows a training and the corresponding validation dataset after refining the first and third elements.

We are able to control the quadrature errors by adding new quadrature points to the training dataset. However, the simplest way to implement such method is

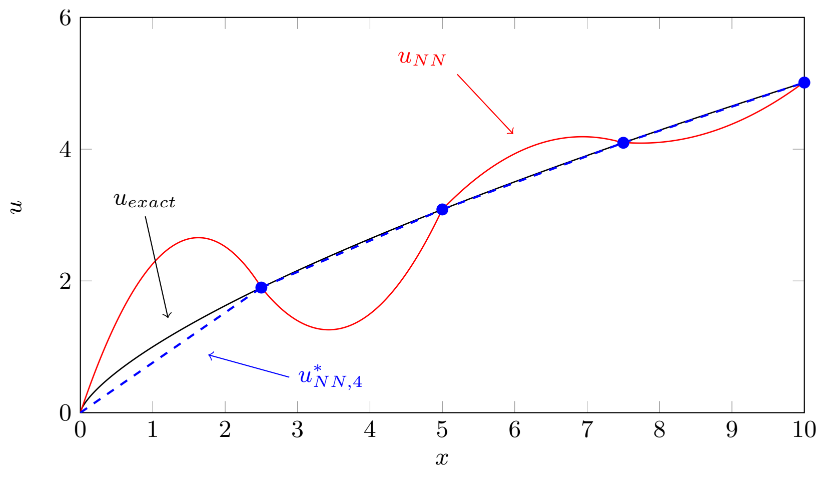
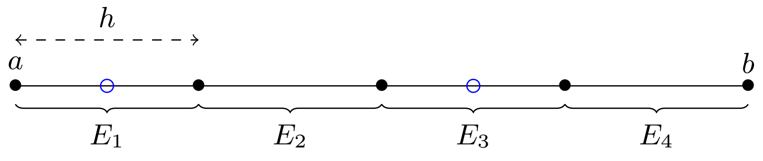


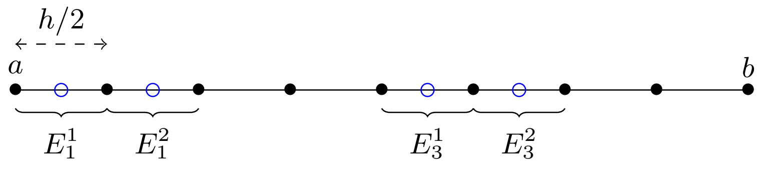
Figure 4.6: Neural Network approximation u_{NN} and its piecewise-linear element approximation $u^*_{NN,4}$.

Algorithm 1: Adaptive integration method

```
Generate a training dataset;
Generate the corresponding validation dataset;
Set tolerance $\epsilon$ and maximum iteration number $i_{max}$;
while $i < i_{max}$ do
    for $j = 1, \cdots, n$ do
        Compute integral values $I_j$ over the training dataset of elements
         $E_j$;
        Compute integral values $I_j^1$,$I_j^2$ over the validation dataset of
         elements $E_j^1$,$E_j^2$ ;
        if $|(I_j^1 + I_j^2) - I_j| > \epsilon$ then
            $h$-refine the $E_j$-th element of the training set;
            $h$-refine the $E_j^1$-th and $E_j^2$-th elements of the validation set;
        else
            continue;
        end
    end
    $i = i + 1$;
end
```



(a) Training set



(b) Validation set

Figure 4.7: Black points (dots) correspond to the original (a) training/(b) validation partitions and blue points (circles) are the points added by the refinement performed in the first and third elements.

by using meshes, which posses a limitation on high-dimensional integrals. As an alternative to generating a mesh, one can randomly add points to the training set. This entails difficulties when designing an adaptive algorithm.

In the same way that we propose an h-adaptive method, we can also work with p-adaptivity [14] or a combination of them (e.g., hp-adaptivity [38]).

4.5.4 Regularization Methods

We now introduce a problem-specific regularizer designed to control the quadrature error.

In a one-dimensional setting, we consider the integral functional $\mathcal{F}_R$ as given in (4.6), and its approximation via a midpoint rule, $\hat{\mathcal{F}}_R$, given by

$$\hat{\mathcal{F}}_R(u) = \frac{b-a}{N} \sum_{j=1}^{N} \left(\frac{1}{2} |u'(x_j)|^2 - f(x_j) u(x_j) \right) + g(b)u(b) + g(a)u(a). \tag{4.29}$$

We note that $g = 0$, except where the Neumann condition is imposed. While we focus on the Ritz method, a similar heuristic can be applied to the LS method.

We introduce a function $\mathcal{R}$ that depends on the learnable parameters θ of a given Neural Network u_{NN}, such that for any Neural Network with a given architecture,

$$|\mathcal{F}_R(u_{NN}) - \hat{\mathcal{F}}_R(u_{NN})| < \mathcal{R}(\theta). \tag{4.30}$$

If we then consider a loss function $\mathcal{L}$ given by

$$\mathcal{L}(\theta) = \hat{\mathcal{F}}_R(u_{NN}) + \mathcal{R}(\theta), \tag{4.31}$$

then we may be able to improve the approximation of the quadrature rule, as the loss contains a term that by design controls the quadrature error.

For simplicity, we consider only the case of a single-layer network with a one-dimensional input, and the midpoint rule for calculating the integral over a uniform partition of (a, b). We consider a mid-point rule as in (4.29), and define the interval length $\delta = \frac{b-a}{N}$ and intervals $I_j = \left(\frac{\delta}{2} + x_j, x_j + \frac{\delta}{2}\right)$. We estimate the error of the midpoint rule to integrate F as

$$\begin{aligned}\left|\int_a^b F(x)\,dx - \sum_{j=1}^N F(x_j)\delta\right| &= \left|\sum_{i=1}^N \int_{x_j-\frac{\delta}{2}}^{x_j+\frac{\delta}{2}} (F(x) - F(x_j))\,dx\right| \\ &\leqslant \sum_{i=1}^N \int_{x_j-\frac{\delta}{2}}^{x_j+\frac{\delta}{2}} |F(x) - F(x_j)|\;dx \\ &\leqslant \sum_{i=1}^N \max_{t\in I_j} |F'(t)| \int_{x_j-\frac{\delta}{2}}^{x_j+\frac{\delta}{2}} |x - x_j|\,dx \\ &= \frac{\delta^2}{4}\sum_{i=1}^N \max_{t\in I_j} |F'(t)|.\end{aligned} \tag{4.32}$$

This estimate scales as $\mathcal{O}(\frac{1}{N})$ for fixed F, and thus, for a large number of integration points, we expect the estimate to be sufficiently accurate and to avoid "overdamping" of the loss.

With (4.32) in mind, we estimate the local Lipschitz constants of the integrand as in (4.6). The numerical estimation of the Lipschitz constants of NNs has attracted attention, as they form a way of estimating the generalizability of a Neural Network, and have been used in the training process as a way to encourage accurate generalization [44, 51, 125]. As we are dealing with loss functions that involve $d$erivatives of the Neural Network, we however need estimates of higher order derivatives of u_{NN}. The approach that we employ is similar in spirit to the work of [88] for obtaining a posteriori error estimates in PINNs.

Despite the arithmetic complications involved in calculating $\mathcal{R}$, conceptually the idea reduces to an application of Taylor's theorem. On a single interval of integration I_j, we have that for every x, there exists some ξ_x so that

$$|F'(x)| = |F'(x_i) + (x - x_i)F''(\xi_x)| \leqslant |F'(x_i)| + \frac{\delta}{2}||F''||_\infty. \tag{4.33}$$

We then find $\mathcal{R}$ using a combination of local and global estimates for the derivatives of the integrand corresponding to simple pointwise evaluations at the integration points and global estimates involving the Neural Network weights. The necessary steps are:

1. Using the chain rule, we find global upper bounds for the derivatives of a simple Neural Network in terms of the weights.
2. Via Taylor's theorem with remainder and using the global estimates for simple networks, we find local estimates of the derivatives of NNs with a cutoff function to ensure a homogeneous Dirichlet condition.
3. Using local estimates for the derivatives of a NN, we find local Lipschitz estimates for integrands corresponding to the Ritz method.

We tackle each of these estimations in the following subsections.

4.5.4.1 Global Estimates for Derivatives of a Single Layer Network

Let $\hat{u}_{NN}$ be a single layer Neural Network. We write it in the form

$$\hat{u}_{NN}(x) = b^1 + \sum_{i=1}^{M} A_i^1 \sigma(A_i^0 x + b_i^0). \tag{4.34}$$

for weights $\theta = (b^1, A^1, b^0, A^0)$ and an activation function σ. We assume that σ has globally bounded derivatives, so that $||\sigma^{(n)}||_\infty$ is finite for every $n = 0, 1, 2....$

A simple application of the triangle quality and that σ is bounded gives that

$$\begin{aligned} |\hat{u}_{NN}(x)| \leqslant & |b^1| + \sum_{i=1}^{M} |A_i^1 \sigma(A_i^0 x + b_i^0)| \\ \leqslant & |b^1| + \sum_{i=1}^{M} |A_i^1| ||\sigma||_\infty \end{aligned} \tag{4.35}$$

The derivatives of $\hat{u}_{NN}$ are

$$\hat{u}_{NN}^{(n)}(x) = \sum_{i=1}^{M} A_i^1 (A_i^0)^n \sigma^{(n)}(A_i^0 x + b_i^0), \tag{4.36}$$

for $n \geqslant 1$. Similarly, this gives the immediate estimate that

$$\left|\hat{u}_{NN}^{(n)}(x)\right| \leqslant \sum_{i=1}^{M} |A_i^1| |A_i^0|^n ||\sigma^{(n)}||_\infty. \tag{4.37}$$

Thus, we define the global upper bounding function $\mathcal{R}^1(\theta; n)$ by

$$\mathcal{R}^1(\theta; n) = \begin{cases} |b^1| + ||\sigma||_\infty \sum_{i=1}^{M} |A_i^1| & n = 0, \\ ||\sigma^{(n)}||_\infty \sum_{i=1}^{M} |A_i^1| |A_i^0|^n & n \geqslant 1, \end{cases} \tag{4.38}$$

which gives the global upper bound

$$\left|\hat{u}_{NN}^{(n)}(x)\right| \leqslant \mathcal{R}^1(\theta; n) \tag{4.39}$$

for every x.

4.5.4.2 Local Derivative Estimation of a Single Layer Neural Network With Cutoff Function

Next, we consider the commonly considered case where our Neural Network admits a cutoff function to ensure a Dirichlet boundary condition, and turn to the case of *l*ocal estimation. Explicitly, we take $u_{NN}(x) = \hat{u}_{NN}(x)\phi(x)$, where ϕ is zero at the homogeneous Dirichlet condition, and $\hat{u}_{NN}$ is as in (4.34). We presume that ϕ has bounded derivatives; i.e. $||\phi^{(k)}||_\infty$ is finite for $k = 0, 1, ...$ Let x be in the interval $I_j = \left(x_j - \frac{\delta}{2}, x_j + \frac{\delta}{2}\right)$. Then, we have

$$\begin{aligned} u_{NN}^{(n)}(x) =& u_{NN}^{(n)}(x_j) + (x - x_j) u_{NN}^{(n+1)}(\xi_x) \\ =& u_{NN}^{(n)}(x_j) + (x - x_j) \sum_{k=0}^{n+1} \binom{n+1}{k} \hat{u}_{NN}^{(k)}(\xi_x) \phi^{(n+1-k)}(\xi_x) \end{aligned} \tag{4.40}$$

via Taylor's theorem with remainder and the product rule for higher order derivatives. Employing the global upper bounds $\mathcal{R}^1(\theta; n)$ from (4.38), we have that for $x \in I_j$,

$$\begin{aligned} \left|u_{NN}^{(n)}(x)\right| \leqslant& \left|u_{NN}^{(n)}(x_j)\right| + \frac{\delta}{2} \sum_{k=0}^{n+1} \binom{n+1}{k} |\hat{u}_{NN}^{(k)}(\xi_x)||\phi^{(n+1-k)}(\xi_x)| \\ \leqslant& \left|u_{NN}^{(n)}(x_j)\right| + \frac{\delta}{2} \sum_{k=0}^{n+1} \binom{n+1}{k} \mathcal{R}^1(\theta; k) ||\phi^{(n+1-k)}||_\infty. \end{aligned} \tag{4.41}$$

Thus, we define the second intermediate regularizer as

$$\mathcal{R}^2(\theta; I_j, n) = \left|u_{NN}^{(n)}(x_j)\right| + \frac{\delta}{2} \sum_{k=0}^{n+1} \binom{n+1}{k} \mathcal{R}^1(\theta; k) ||\phi^{(n+1-k)}||_\infty, \tag{4.42}$$

giving the estimate that for all $x \in I_j$,

$$\left|u_{NN}^{(n)}(x)\right| \leqslant \mathcal{R}^2(\theta; I_j, n). \tag{4.43}$$

We note that via automatic differentiation, $u^{(n)}$ may be evaluated at the training data x_j.

4.5.4.3 Application to Integral Errors

Our aim is to estimate the error in the integral functional

$$\mathcal{F}(u_{NN}) = \int_a^b \frac{1}{2}|u'_{NN}(x)|^2 - f(x)u_{NN}(x)\,dx - g(a)u_{NN}(a) - g(b)u_{NN}(b), \tag{4.44}$$

when approximated by a simple quadrature rule

$$\sum_{j=1}^{N}\left(\frac{1}{2}|u'_{NN}(x_j)|^2 - f(x_j)u_{NN}(x_j)\right)\delta - g(b)u_{NN}(b) - g(a)u_{NN}(a). \tag{4.45}$$

The boundary terms can be calculated in one dimension without quadrature error and thus we ignore their contribution. We estimate the error for the quadrature rule by obtaining Lipschitz bounds of the integrand via

$$\begin{aligned}&\left|\frac{d}{dx}\left(\frac{1}{2}|u'_{NN}(x)|^2 - f(x)u_{NN}(x)\right)\right|\\ =&\,|u'_{NN}(x)u''_{NN}(x) - f'(x)u_{NN}(x) - f(x)u'_{NN}(x)|\\ \leqslant&\,|u'_{NN}(x)||u''_{NN}(x)| + |f'(x)||u_{NN}(x)| + |f(x)||u'_{NN}(x)|.\end{aligned} \tag{4.46}$$

Estimating the (local) Lipschitz constant of the integrand reduces to estimating (locally) various derivatives of u_{NN}. For $x \in I_j$, we estimate the Lipschitz constant of the integrand via

$$\left|\frac{d}{dx}\left(\frac{1}{2}|u'_{NN}(x)|^2 - f(x)u_{NN}(x)\right)\right| \leqslant \mathcal{R}^3(\theta; I_j), \tag{4.47}$$

where the regularizer $\mathcal{R}^3(\theta, I_j)$ is given by

$$\mathcal{R}^3(\theta; I_j) = \Big(\mathcal{R}^2(\theta; I_j, 1)\mathcal{R}^2(\theta; I_j, 2) + ||f||_\infty\mathcal{R}^2(\theta; I_j, 1) + ||f'||_\infty\mathcal{R}^2(\theta; I_j, 0)\Big). \tag{4.48}$$

We define the final regularizer $\mathcal{R}$ by

$$\mathcal{R}(\theta) = \frac{\delta^2}{4}\sum_{j=1}^{N}\mathcal{R}^3(\theta; I_j), \tag{4.49}$$

which following (4.32) gives the estimate

$$\begin{aligned}&|\mathcal{F}(u_{NN}) - \hat{\mathcal{F}}(u_{NN})|\\ =&\left|\int_a^b \frac{1}{2}|u'_{NN}(x)|^2 - f(x)u_{NN}(x)\,dx - \sum_{j=1}^{N}\left(\frac{1}{2}|u'_{NN}(x_j)|^2 - f(x_j)u_{NN}(x_j)\right)\delta\right|\\ \leqslant&\,\mathcal{R}(\theta).\end{aligned} \tag{4.50}$$

4.6 Numerical Results

In this section, we study numerically some of the alternatives proposed in Section 4.5 to overcome the quadrature problems. Specifically, we solve the two model problems from Section 4.1 using: (a) a piecewise-linear approximation of the NN, and (b) an adaptive integration method. We also solve model problem 2 using regularization methods.

For the cases of piecewise-linear approximation and adaptive integration, we use a NN architecture composed of one hidden layer with ten neurons and a *sigmoid* activation function. We select SGD as the optimizer. In the case of regularization, we use a *hyperbolic tangent* activation function with the Adam optimizer [155].

4.6.1 Piecewise-linear Approximation

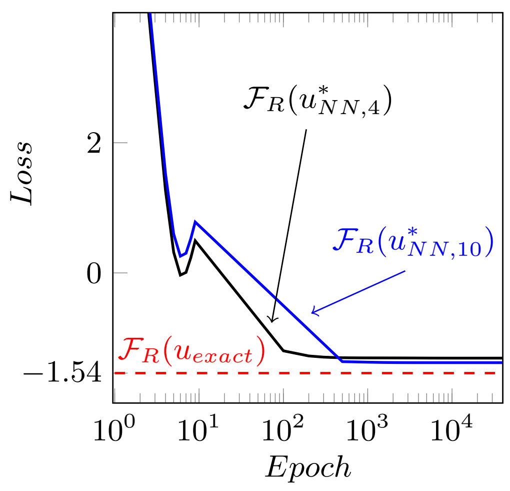
We select a piecewise-linear approximation of the NN as our approximate solution. We compute the gradients using FDM and the integrals using a one-point Gaussian quadrature rule (i.e., the midpoint rule). For model problem 1, we use two different uniform partitions composed of four and ten elements, and execute $40,000$ iterations. For model problem 2, we use a uniform partition composed of ten elements and execute $200,000$ iterations.

Figures 4.8a and 4.8b show that the loss converges to the loss of the exact solution. We also observe better results as we increase the number of elements, as physically expected. Figures 4.9a and 4.9b show the corresponding solutions, which are consistent with the loss evolutions displayed in Figures 4.8a and 4.8b.

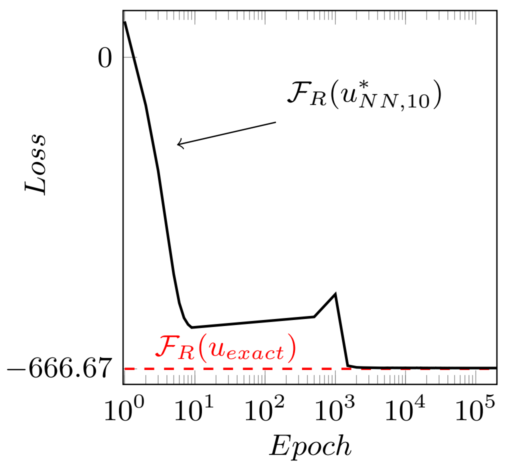
Table 4.1 shows the loss value of the exact solution, of the optimum piecewise-/linear solution, and of the obtained Deep Neural Network (DNN) piecewise-linear solution. Since we are solving a 1D Laplace problem, the best piecewise-linear approximation of the solution is its interpolator at the vertex nodes. For model problem 1, we observe that we do not obtain the best piecewise-linear solution. For model problem 2, we reach the optimum piecewise-solution.

	$\mathcal{F}_R(u_{exact})$	$\mathcal{F}_R(\tilde{u}^*_{NN,4})$	$\mathcal{F}_R(u^*_{NN,4})$	$\mathcal{F}_R(\tilde{u}^*_{NN,10})$	$\mathcal{F}_R(u^*_{NN,10})$
Model problem 1	-1.54	-1.36	-1.31	-1.41	-1.38
Model problem 2	-666.67	-	-	-665	-664.99

Table 4.1: Loss values of the exact solution $\mathcal{F}_R(u_{exact})$, optimum piecewise-linear solution $\mathcal{F}_R(\tilde{u}^*_{NN,\cdot})$ (for a four and a ten equidistant element partition), and piecewise-linear solution $\mathcal{F}_R(u^*_{NN,\cdot})$ using a DNN.

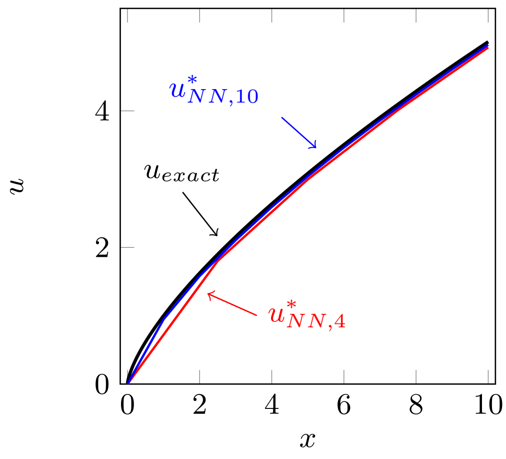


(a) Model problem 1.

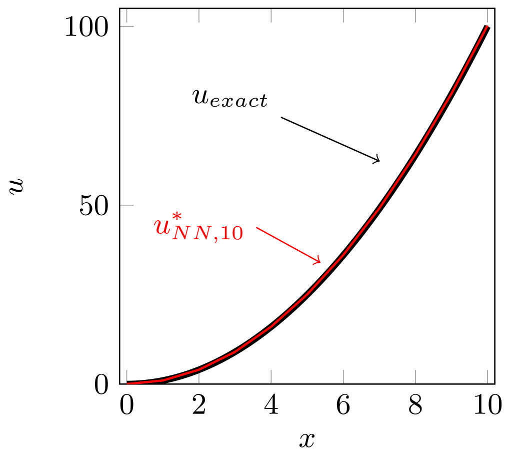


(b) Model problem 2.

Figure 4.8: Loss evolution of the training process for our two model problems when we use a piecewise-linear approximation of the NN.



(a) Model problem 1.



(b) Model problem 2.

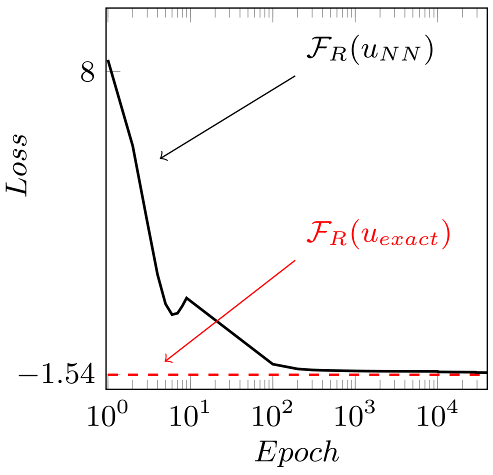
Figure 4.9: Ritz method solution when we use a piecewise-linear approximation of the NN to solve the problem.

While the use of a piecewise-linear approximation overcomes the quadrature problems, the convergence is limited to $\mathcal{O}(h)$, where h is the element size [17]. To increase this speed, it is possible to consider different piecewise-polynomial approximations, including the use of r-adaptive algorithms [19].

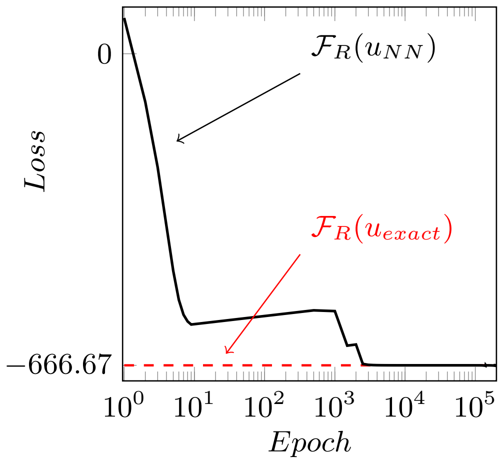
4.6.2 Adaptive Integration

We compute the gradients using automatic differentiation and the integrals using a three-point Gaussian quadrature rule. As in our previous examples, we start the training with a uniform partition of four elements for model problem 1 and of ten elements for model problem 2. We select a half-size partition of the training dataset for validation. For model problem 1, we compare the integral values of the training and validation sets (i.e., we execute Algorithm 1) every 1000 iterations, with an error tolerance of $0,01$; for model problem 2, we compare every $10,000$ iterations, with an error tolerance of 10. The tolerance is selected as a small percentage of the value of the loss. In both cases, we do not execute the adaptive algorithm in the first $10,000$ iterations.

Figures 4.10a and 4.10b show that the loss converges to the optimum value. As explained in Section 4.5, the adaptive integration algorithm refines the training dataset. For model problem 1, the algorithm performs two refinements in the first interval. For model problem 2, one refinement occurs in the first interval. The adaptive integration algorithm automatically selects the first interval for refinement, where overfitting was taking place in Figures 4.3 and 4.4. Figures 4.11a and 4.11b show the corresponding solutions. We observe that the approximate solutions properly approximate the exact ones.



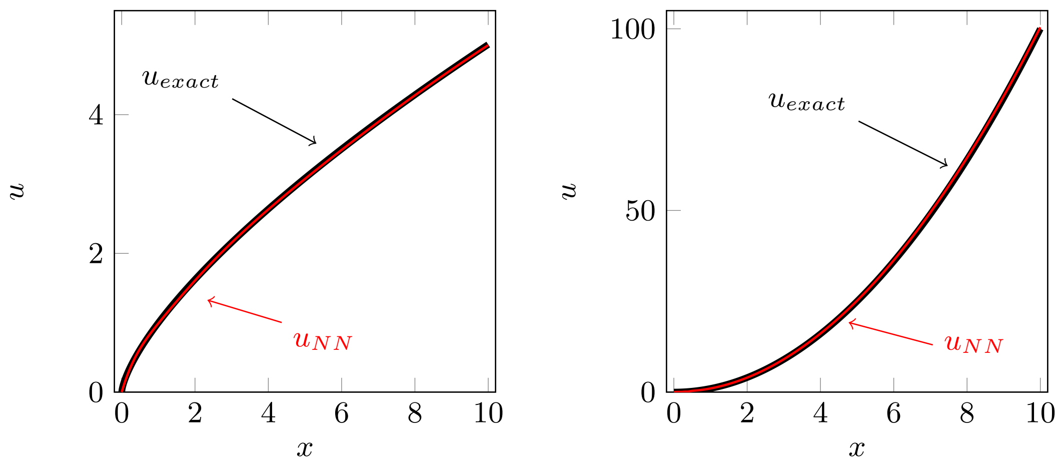
(a) Model problem 1.



(b) Model problem 2.

Figure 4.10: Loss evolution of the training process for our two model problems when using adaptive integration.

The extrapolation of this method to higher dimensions (2D or 3D) requires higher-dimensional discretizations and quadrature rules.



(a) Model problem 1. (b) Model problem 2.

Figure 4.11: Ritz method solution when using adaptive integration.

4.6.3 Regularization Methods

We do not apply the regularization method to model problem 1 as the method requires sufficient regularity in order to provide the necessary estimates in the calculation of $\mathcal{R}$. Since the solution is singular at $x = 0$, the necessary Lipschitz bounds on the integral functional cannot be obtained within this framework. Instead, we aim to demonstrate that for problems that are sufficiently regular, our technique can avoid overfitting, and leave open the question as to how one may adapt the technique to singular problems for future work. We thus consider model problem 2. We propose the loss defined via

$$\mathcal{L}(\theta) = \hat{\mathcal{F}}_R(u_{NN}) + \mathcal{R}(\theta). \tag{4.51}$$

Explicitly,

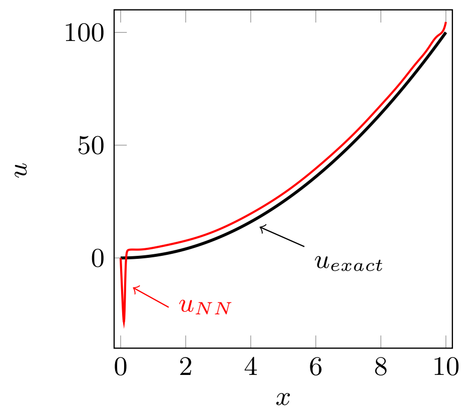
$$\hat{\mathcal{F}}_R(u_{NN}) = \frac{10}{N}\sum_{j=1}^{N} \frac{1}{2}|u'_{NN}(x_j)|^2 - 2u_{NN}(x_j) - 20u_{NN}(20), \tag{4.52}$$

where $x_j = \frac{10}{N}\left(i - \frac{1}{2}\right)$.

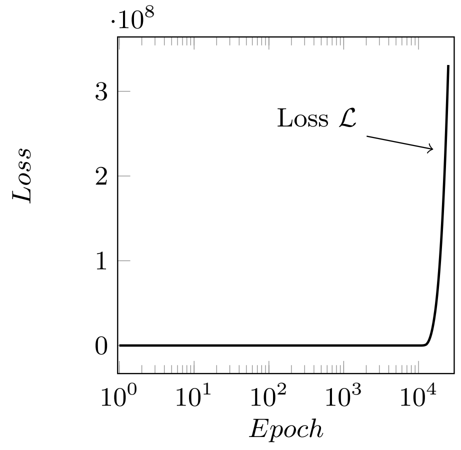
4.6.3.1 Experiment 1

We consider $N = 50$ points, and a single layer network with $M = 10$ neurons. We use the Adam optimizer with learning rate 10^{-2}. We solve model problem 2 with two losses: with and without regularization. In both cases, we measure the metrics $\mathcal{L}$, $\mathcal{R}$, and $\hat{\mathcal{F}}_R$. For validation, we use an equidistant partition of $(0, 10)$

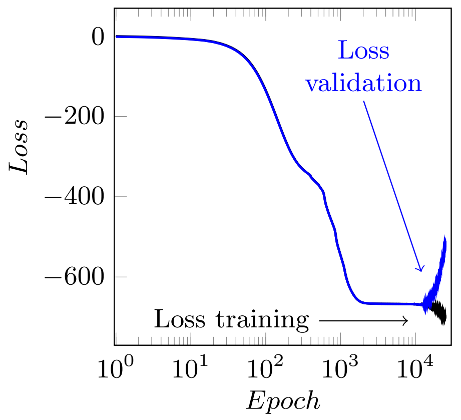
with 49 points, so that we still use a midpoint rule but with different integration points.



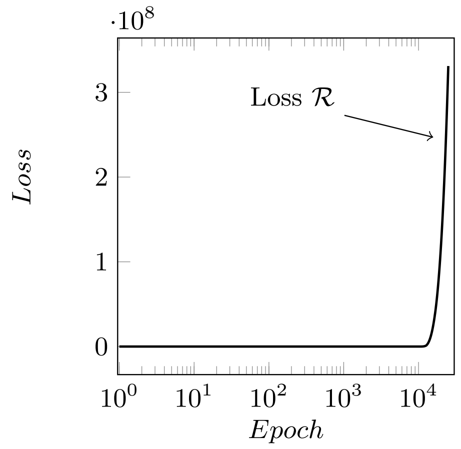
(a) Exact and approximated solutions.



(b) Evolution of $\mathcal{L}$ during training.



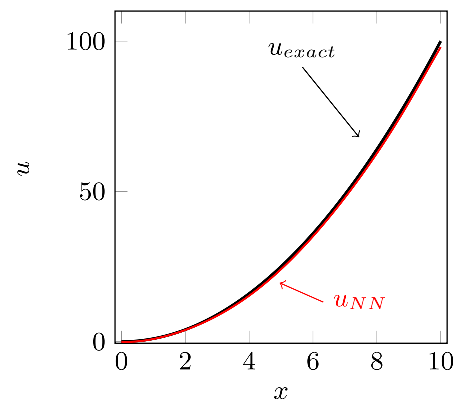
(c) Evolution of $\hat{\mathcal{F}}_R$ during training.



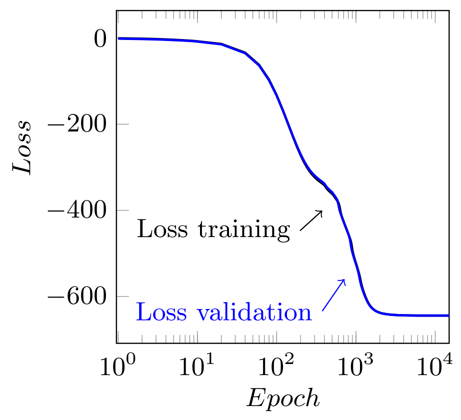
(d) Evolution of $\mathcal{R}$ during training.

Figure 4.12: The solution and training information for Experiment 1 without regularization.

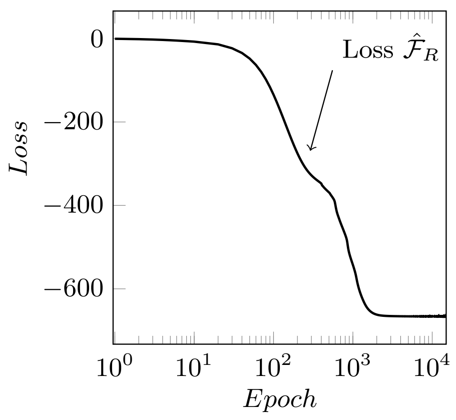
Figure 4.12 shows the results without regularization. As expected, we see in Figure 4.12a that the approximation is poor due to overfitting, which is most notable around $x = 0$ and attained within 5000 epochs. Via the provided plots we can observe the beginning of overfitting in two distinct manners. First, we observe in Figure 4.12c that the value of $\hat{\mathcal{F}}_R$ evaluated over the validation data begins to diverge from the value on the training data, becoming apparent at around 1000 epochs. We also see this behaviour reflected in the evolution of $\mathcal{R}$ in Figure 4.12d, with its most dramatic increase beginning around the same



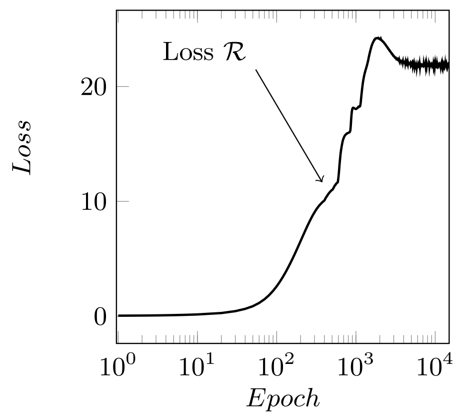
(a) Exact and approximated solutions.



(b) Evolution of $\mathcal{L}$ during training.



(c) Evolution of $\hat{\mathcal{F}}_R$ during training.



(d) Evolution of $\mathcal{R}$ during training.

Figure 4.13: The solution and training information for Experiment 1 with regularization.

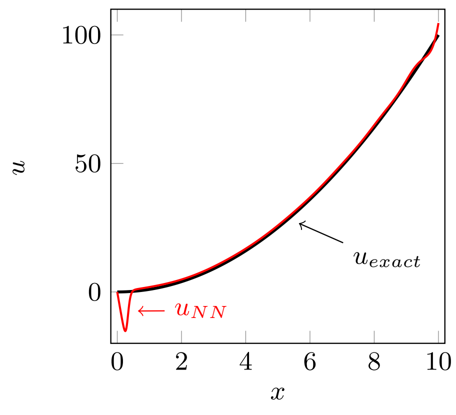
iteration. This rapid increase also provokes an increase in $\mathcal{L}$, as seen in Figure 4.12b. This also indicates that even if $\mathcal{R}$ is not used as part of the training process, its increase could be used as a metric to identify overfitting.

Figure 4.13 describes the results with regularization and we observe a different behaviour. The approximation is generally good, and we do not see any signs of overfitting within 10^5 epochs, as shown in Figure 4.13a. In particular, the values of $\hat{\mathcal{F}}_R$ at the training and validation data remain consistent in Figure 4.13c. Throughout Figure 4.13 we see that within 10^5 epochs all metrics appear to have converged to a limiting value. We obtain final values $\mathcal{L} \approx -644.22$, $\hat{\mathcal{F}}_R \approx -666.07$, $\mathcal{R} \approx 24.8$. We recall that the true energy of the exact solution is $\mathcal{F}_R(u_{exact}) \approx -666.667$, which suggests the quadrature rule is accurate. Notice that in the case without regularization, before overfitting became apparent, $\mathcal{R}$

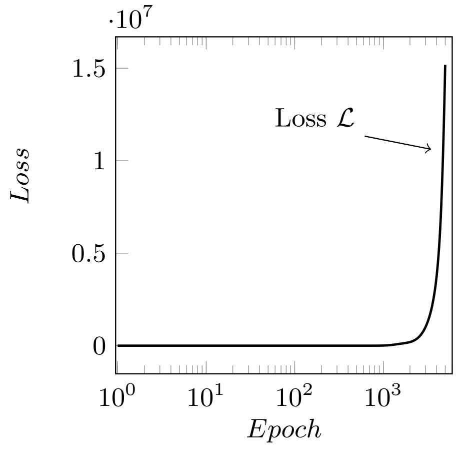
had already attained values of around 1000, which is far larger than the value of $\mathcal{R}$ at the obtained solution when regularization was used.

4.6.3.2 Experiment 2

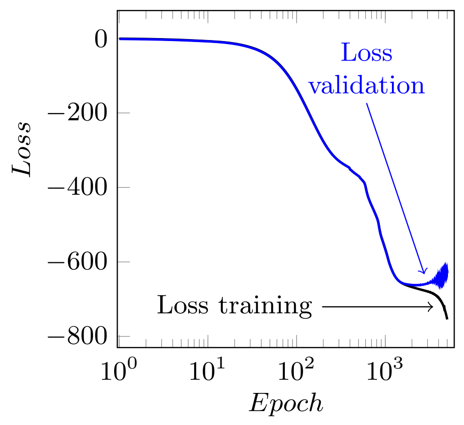
We now consider a smaller N. As we expect $\mathcal{R}$ to scale as $\frac{1}{N}$, we anticipate a more adverse effect when N is small. To view this, we consider the same problem of Experiment 1, where we now select $N = 20$ integration points. We consider $M = 10$ neurons and minimize our problem using the Adam optimizer with a learning rate of 10^{-2}. As before, we consider the cases with and without regularization.



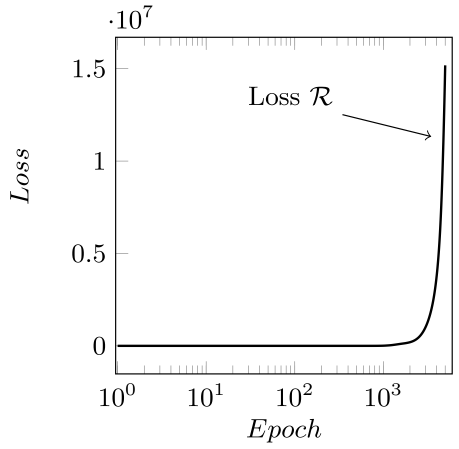
(a) Exact and approximated solutions.



(b) Evolution of $\mathcal{L}$ during training.

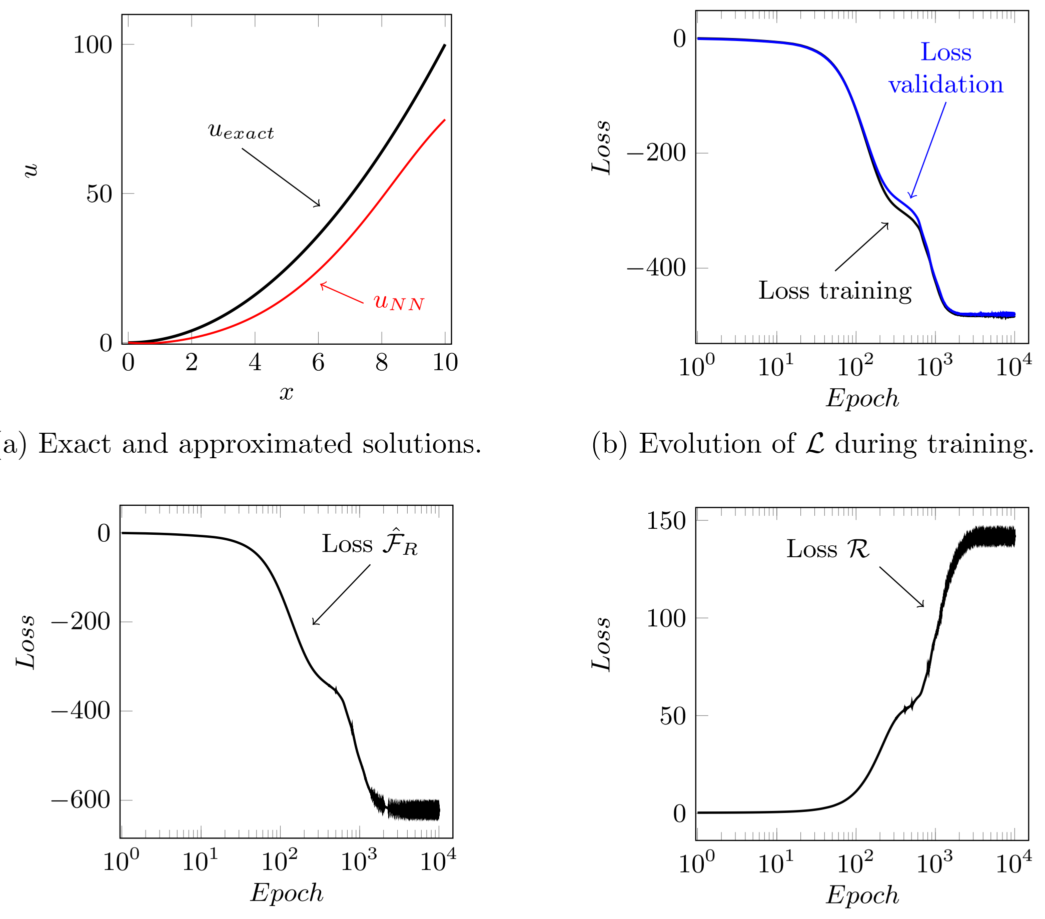


(c) Evolution of $\hat{\mathcal{F}}_R$ during training.



(d) Evolution of $\mathcal{R}$ during training.

Figure 4.14: The solution and training information for Experiment 2 without regularization.



(a) Exact and approximated solutions.

(b) Evolution of $\mathcal{L}$ during training.

(c) Evolution of $\hat{\mathcal{F}}_R$ during training.

(d) Evolution of $\mathcal{R}$ during training.

Figure 4.15: The solution and training information for Experiment 2 with regularization.

Figure 4.14 presents the loss evolution without regularization. We observe overfitting, which is accompanied by divergence of the loss on the validation dataset, as well as a rapid increase in $\mathcal{R}$, with these features visible within 5000 epochs.

Figure 4.15 presents the results with regularization. We observe no signs of overfitting, with the validation and training loss remaining close in Figure 4.15b. All metrics appear to have converged to a limiting value within 10^4 epochs. However, the large value of $\mathcal{R}$ at the found solution (approximately 140) has substantially changed the optimization problem so that the obtained minimizer is far from the desired solution. The final value of $\hat{\mathcal{F}}_R$ is around -622, which is far from the desired value of -666.67. This experiment highlights the fact that the regularizer becomes more effective when a large number of integration points are used.

5 Conclusions and Future Work

5.1 Conclusions

In this dissertation, we first focus on the use of Deep Neural Networks (DNNs) for the inversion of borehole resistivity measurements for geosteering applications. We analyze the strong impact that different loss functions have on the prediction results. For this, we illustrate via a simple benchmark example that a traditional data misfit loss function delivers poor results. As a remedy, we propose the use of an Encoder-Decoder based or a two-step based loss function. These approaches generate two DNN approximations: one for the forward function and another one for the inverse operator. Then, we apply these two loss functions in a field example with synthetic data, and we obtain adequate results.

To guarantee that the inverse DNN approximation provides meaningful results, we need to ensure that the training dataset contains sufficient samples. Otherwise, both forward and inverse DNN operators may provide incorrect solutions while still ensuring the composition of both operators is close to the identity. Thus, the approach is highly dependent on the existence of a sufficiently rich training dataset, which facilitates the learning process of the DNNs.

To ensure that the inverse DNN approximation delivers significant results, we find it highly beneficial to add a regularization term to the loss function based on the existing training dataset. This reduces the richness we need to guarantee within the training datasets. Nevertheless, such regularization terms may hide alternative feasible solutions for the inverse operator, which may provide overconfidence in the results. Another possibility is to consider a two-step based loss function. Using this approach, we have shown that the **inverse problem considered in this work admits different solutions that are physically feasible**, a fact that was obscured when using the regularization term.

Other critical limitations of DNNs we encounter in this work are: (a) the limited approximation capabilities of DNNs to reproduce discontinuous functions, (b) the need for a new dataset and trained DNN for each subsurface parametrization, and (c) the poor results they exhibit when they are evaluated over a sample that is outside the training dataset space. More importantly, it is often difficult to identify the source of poor results, which may include inadequate selections of: (i) loss function, (ii) DNN architecture, (iii) regularization term, (iv) training dataset, (v) optimization algorithm, (vi) rescaling operator and norms, (vii) model parameterization, (viii) approximation capabilities of DNNs, or simply (ix) the nature of the problem due to a lack of adequate measurements.

To deal with the afforementioned sources of errors, **we propose a careful step-by-step error control** based on: (a) selecting adequate norms, (b) proper rescaling of the variables, (c) selecting a well suited loss function possibly with a regularization term, (d) analyzing the evolution of the different terms of the loss function, (e) studying multiple cross-plots of different nature, and (f) performing an in-depth assessment of the results over multiple realistic test examples.

We also show it is possible to obtain a good-quality inversion of geosteering measurements with limited *online* computational cost, thus, suitable for real-time inversion. Moreover, the quality of the inversion results can be rapidly evaluated to detect its possible inaccuracies in the field and select alternative inversion methods when needed.

As mentioned before, the DNN approximation of the inverse operator is highly dependent on the existence of a sufficiently rich training dataset. In the case of 1D layered formations, it is often feasible to produce the required dataset. However, for more complicated cases, for example, in 2D and 3D geometries, a direct extension may be limited due to the larger number of inversion variables and the extremely time-consuming process of producing an exhaustive dataset. Such a large database is essential for layer-by-layer estimation of the inverted Earth models, which may be used for real-time adjustments of the well trajectory during geosteering operations.

In the second part of this dissertation, **we propose the use of refined Isogeometric Analysis (rIGA) discretizations for generating a massive synthetic database for Deep Learning (DL) inversion of 2.5D borehole electromagnetic (EM) measurements**. rIGA delivers computational savings of up to $\mathcal{O}(p)$ compared to the high-continuity Isogeometric Analysis (IGA). When compared to a traditional Finite Element Method (FEM) with the same mesh size and polynomial degree, rIGA provides higher improvement factors. At the same time, rIGA provides sufficiently accurate solutions for geosteering purposes.

To create a dataset for DL inversion, we first selected certain discretization parameters based on the results of several homogeneous solutions. Then, we checked the accuracy over homogeneous and heterogeneous media. Finally, we generated a synthetic database composed of 100,000 Earth models with the corresponding measurements in about 56 hours using a workstation equipped with two CPUs.

In the last part of this work, we focus on the use of Neural Networks (NNs) for solving a Partial Differential Equation (PDE). We first illustrate via two simple examples how **quadrature errors can destroy the quality of the**

approximated solution when solving PDEs using DL methods. For this, we solve two simple 1D problems based on Poisson's equation using the Deep Ritz Method (DRM) and a three-point Gaussian quadrature rule. Then, **we propose four different alternatives to overcome the quadrature problems**, discuss their advantages and limitations, and illustrate their performance.

In high dimensions, Monte Carlo integration methods are the best choice. Regularizer methods are another option, but they are problem dependent and they need to be derived for each different architecture. Moreover, they require further analysis for highly nonlinear integrands. Furthermore, they are limited only to sufficiently smooth integral functionals. In addition, more complex NN architectures (which should be needed in higher dimensions) will hinder the derivation of $\mathcal{R}$.

In low dimensions (three or below), Monte Carlo integration is not competitive because of its low convergence speed. In these cases, adaptive integration exhibits faster convergence. In the cases of piecewise-linear approximation and regularizers, we are also able to overcome the quadrature problems, but the convergence speed is often slower and the accuracy is lower than with adaptive integration.

5.2 Future Work

There are several possible future research lines regarding this work. The first one is to consider more complex Earth models, possibly containing geological faults or other relevant subsurface features, and analyze the performance of the Encoder-Decoder and two-step based loss functions.

Another line of research consists of reducing the dataset size required for solving inverse borehole problems. Thus, decreasing the computational cost of creating the dataset. For this, we use using Active Learning techniques. Another option is to use Transfer Learning techniques for higher spatial dimensions, which can also alleviate data requirements to train the corresponding DNN.

Concerning Chapter 4, one possible future work is to implement adaptive integration for 2D and 3D problems. In the same way, the piecewise-polynomial approximation could be improved by implementing r-adaptivity to optimize the grid.

Ultimately, we aim at solving parametric PDEs using NNs. By this, we will be able to solve the forward problem corresponding to different Earth models using NNs. Thus, we will be able to properly and efficiently create the dataset needed to train our DNN that approximates the solution of inverse borehole problems.

6 Main Achievements

6.1 Scientific Achievements

In the first part of this dissertation, we investigate appropriate loss functions to train a Deep Neural Network (DNN) when dealing with an inverse problem.

In the second part of this work, we propose the use of refined Isogeometric Analysis (rIGA) discretizations to generate databases for DL inversion of 2.5D geosteering electromagnetic (EM) measurements.

In the third part of this work, we analyze the problems associated with quadrature rules in Deep Learning (DL) methods when solving Partial Differential Equations (PDEs), and we propose several alternatives to overcome quadrature problems.

6.2 Peer-reviewed Publications

6.2.1 Journals

2022 J. A. Rivera, J. M. Taylor, Á. J. Omella and D. Pardo. ***On quadrature rules for solving Partial Differential Equations using Neural Networks.*** Computer Methods in Applied Mechanics and Engineering, 2022, vol. 393, p. 114710.
`https://doi.org/10.1016/j.cma.2022.114710`

2021 A. Hashemian, D. Garcia, J. A. Rivera and D. Pardo. ***Massive database generation for 2.5 D borehole electromagnetic measurements using refined isogeometric analysis.*** Computers & Geosciences, 2021, vol. 155, p. 104808.
`https://doi.org/10.1016/j.cageo.2021.104808`

2021 M. Shahriari, D. Pardo, J. A. Rivera, C. Torres-Verdín, A. Picon, J. Del Ser, S. Ossandón and V. M. Calo. ***Error control and loss functions for the deep learning inversion of borehole resistivity measurements.*** International Journal for Numerical Methods in Engineering, 2021, vol. 122(6), p. 1629-1657.
`https://doi.org/10.1002/nme.6593`

2020 J. A. Rivera, D. Pardo and E. Alberdi. ***Design of loss functions for solving inverse problems using deep learning.*** International Conference on Computational Science, 2020, p. 158-171.
`https://doi.org/10.1007/978-3-030-50420-5_12`

6.2.2 Extended Abstracts

2021 J.A. Rivera, M. Shahriari, D. Pardo, J. Omella and C. Torres-Verdín. ***Uncertainty Quantification on the Inversion of Geosteering Measurements using Deep Learning.*** Conference Proceedings, 3rd EAGE/SPE Geosteering Workshop, 2021, vol. 2021, p. 1-5.
`https://doi.org/10.3997/2214-4609.2021624005.`

6.3 International Conferences

2022 J. A. Rivera, A. J. Omella, J. M. Taylor and D. Pardo. ***On quadrature rules for solving Partial Differential Equations with Neural Networks.***
ECCOMAS 2022, Oslo, Norway.

2022 M. Shahriari, D. Pardo and J. A. Rivera. ***On quadrature rules for solving Partial Differential Equations with Neural Networks.***
ECCOMAS 2022, Oslo, Norway.

2021 J. A. Rivera, M. Shahriari, D. Pardo, J. Omella and C. Torres-Verdín. ***Uncertainty Quantification on the Inversion of Geosteering Measuremenets using Deep Learning.***
3RD EAGE/SPE GEOSTEERING WORKSHOP, Online.

2021 M. Shahriari, D. Pardo, A. Hazra, and J. A. Rivera. ***Application of auto-ML for deep learning-based inversion of borehole resistivity measurement.***
MMLDT-CSET 2021, California, USA.

2021 A. Hashemian, D. Garcia, J. A. Rivera and D. Pardo. ***Refined Isogeometric Analysis: an Efficient Numerical Method for Massive Database Generation for 2.5D Borehole Electromagnetic Measurements.***
MMLDT-CSET 2021, California, USA.

2021 J. A. Rivera, Á. J. Omella and D. Pardo. ***Deep Learning for solving partial differential equations using Ritz method.***
ICCS2021, Krakow, Poland.

2020 D. Pardo, M. Shahriari, C. Torres-Verdín, A. Hazra and J. A. Rivera. ***Deep learning inversion of borehole resistivity measurements: algorithms, uncertainty, and tool design.***
Formation Evaluation Research Consortium 2020, Texas, USA.

2020 J. A. Rivera, D. Pardo and E. Alberdi. ***Design of Loss Functions for Solving Inverse Problems using Deep Learning.***
ICCS2020, Amsterdam, Netherlands.

2019 M. Shahriari, D. Pardo, C. Torres-Verdín, J. A. Rivera, A. Picon and J. Del Ser. ***Design considerations for the deep-learning inversion of borehole resistivity measurements.***
Formation Evaluation Research Consortium 2019, Texas, USA.

2019 J. A. Rivera, E. Alberdi and D. Pardo. ***Zulaketa bidezko erresistibitate neurketen simulazioa problema errealen ebazpenean.***
IkerGazte: Nazioarteko ikerketa euskaraz Kongresua, Bayonne, France.

6.4 Seminars & Workshops

2022 J. A. Rivera, D. Pardo and E. Alberdi. ***Kuadratura erregelen erabilera ekuazio diferentzialak ebazteko ikasketa sakona erabiliz.***
Matematikari Euskaldunen V. Topaketa, Eibar, Spain.

2021 J. A. Rivera. ***Solving Inverse Problems using Deep Learning.*** Seminarium, Krakow, Poland. (Online)

2020 J. A. Rivera, D. Pardo and E. Alberdi. ***Galera-funtzioen diseinua alderantzizko problemak ikasketa sakonaren bidez ebazteko.*** Matematikari Euskaldunen IV. Topaketa, Eibar, Spain.

6.5 Research Stays

2021 Software Competence Center Hagenberg (SCCH), Linz (Austria)
Supervisor: Mostafa Shahriari
Date: 15 September 2021 - 21 December 2021 (98 days)

Bibliography

[1] 14 - Monte Carlo integration I: Basic concepts. In M. Pharr and G. Humphreys, editors, *Physically Based Rendering*, pages 631–660. Morgan Kaufmann, Burlington, 2004. (cited in page(s) 81)

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. (cited in page(s) 17, 24)

[3] A. Abubakar, T. Habashy, V. Druskin, D. Alumbaugh, A. Zerelli, and L. Knizhnerman. Two-and-half-dimensional forward and inverse modeling for marine CSEM problems. In *SEG Technical Program Expanded Abstracts*. Society of Exploration Geophysicists, Jan. 2006. (cited in page(s) 4, 49)

[4] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman. Deep Audio-visual Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 12 2018. (cited in page(s) 3)

[5] M. Alam, M. Samad, L. Vidyaratne, A. Glandon, and K. Iftekharuddin. Survey on Deep Neural Networks in Speech and Vision Systems. *Neurocomputing*, 417:302–321, 2020. (cited in page(s) 3)

[6] J. Alvarez-Aramberri and D. Pardo. Dimensionally adaptive hp-finite element simulation and inversion of 2D magnetotelluric measurements. *Journal of Computational Science*, 18:95–105, 2017. (cited in page(s) 4)

[7] B. I. Anderson. *Modeling and Inversion Methods for the Interpretation of Resistivity Logging Tool Response.* DUP Science, Delft, Netherlands, 2001. (cited in page(s) 53)

[8] W. L. Anderson. A hybrid fast hankel transform algorithm for electromagnetic modeling. *Geophysics*, 54(2):263–266, 1989. (cited in page(s) 24)

[9] H. Antil, R. Khatri, R. Löhner, and D. Verma. Fractional deep neural network via constrained optimization. *Machine Learning: Science and Technology*, 2(1), dec 2020. (cited in page(s) 5)

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. (cited in page(s) 3, 21)

[11] S. A. Bakr, D. Pardo, and T. Mannseth. Domain decomposition Fourier FE method for the simulation of 3D marine CSEM measurements. *J. Comput. Phys.*, 255:456–470, 2013. (cited in page(s) 4)

[12] S. A. Bakr, D. Pardo, and C. Torres-Verdín. Fast inversion of logging-while-drilling resistivity measurements acquired in multiple wells. *Geophysics*, 82(3):E111–E120, May 2017. (cited in page(s) 4, 49)

[13] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997. (cited in page(s) 58)

[14] F. B. Barros, S. P. B. Proença, and C. S. de Barcellos. On error estimator and p-adaptivity in the generalized finite element method. *International Journal for Numerical Methods in Engineering*, 60(14):2373–2398, 2004. (cited in page(s) 83)

[15] J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018. (cited in page(s) 5)

[16] B. Bhanu and A. Kumar. *Deep Learning for Biometrics.* Springer, Switzerland, 2017. (cited in page(s) 3)

[17] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*, volume 15 of *Texts in Applied Mathematics.* Springer, 2008. (cited in page(s) 89)

[18] I. Brevis, I. Muga, and K. Zee. A machine-learning minimal-residual (ml-mres) framework for goal-oriented finite element discretizations. *Computers & Mathematics with Applications*, 95, 2020. (cited in page(s) 5)

[19] C. J. Budd, W. Huang, and R. D. Russell. Adaptivity with moving grids. *Acta Numerica*, 18:111–241, 2009. (cited in page(s) 89)

[20] A. Buffa, G. Sangalli, and R. Vázquez. Isogeometric analysis in electromagnetics: B-splines approximation. *Computer Methods in Applied Mechanics and Engineering*, 199(17-20):1143–1152, Mar. 2010. (cited in page(s) 4, 53, 55)

[21] A. Buffa, G. Sangalli, and R. Vázquez. Isogeometric methods for computational electromagnetics: B-spline and T-spline discretizations. *Journal of Computational Physics*, 257:1291–1320, Jan. 2014. (cited in page(s) 4)

[22] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6), 2021. (cited in page(s) 71)

[23] Z. Cai, J. Chen, M. Liu, and X. Liu. Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. *Journal of Computational Physics*, 420:109707, 2020. (cited in page(s) 6)

[24] V. M. Calo, D. Pardo, and M. R. Paszyński. Goal-oriented self-adaptive *hp* finite element simulation of 3D DC borehole resistivity simulations. *Procedia Computer Science*, 4:1485–1495, 2011. (cited in page(s) 4)

[25] A. C. Cangellaris and D. B. Wright. Analysis of the numerical error caused by the stair-stepped approximation of a conducting boundary in FDTD simulations of electromagnetic phenomena. *IEEE Transactions on Antennas and Propagation*, 39(10):1518–1525, 1991. (cited in page(s) 65)

[26] T. Chaumont-Frelet, S. Nicaise, and D. Pardo. Finite element approximation of electromagnetic fields using nonfitting meshes for geophysics. *SIAM Journal on Numerical Analysis*, 56(4):2288–2321, Jan. 2018. (cited in page(s) 65, 70)

[27] T. Chaumont-Frelet, D. Pardo, and Á. Rodríguez-Rozas. Finite element simulations of logging-while-drilling and extra-deep azimuthal resistivity measurements using non-fitting grids. *Computational Geosciences*, 22(5):1161–1174, 2018. (cited in page(s) 4, 52, 65, 70)

[28] R. Chemali, M. Bittar, F. Hveding, M. Wu, and M. Dautel. Improved geosteering by integrating in real time images from multiple depths of investigation and inversion of azimuthal resistivity signals. *Society of Petrophysicists and Well-Log Analysts*, pages 1–7, 2010. (cited in page(s) 2)

[29] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015. (cited in page(s) 3, 24)

[30] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, and V. M. Calo. The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers. *Computer Methods in Applied Mechanics and Engineering*, 213-216:353–361, 2012. (cited in page(s) 4)

[31] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, Ltd, 2009. (cited in page(s) 4, 5)

[32] L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55, 1998. (cited in page(s) 58)

[33] L. Dalcin, N. Collier, P. Vignal, A. M. A. Côrtes, and V. M. Calo. PetIGA: A framework for high-performance isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 308:151–181, 2016. (cited in page(s) 57)

[34] S. Davydycheva. Two triaxial induction tools: sensitivity to radial invasion profile. *Geophysical Prospecting*, 59(2):323–340, 2011. (cited in page(s) 52)

[35] S. Davydycheva, D. Homan, and G. Minerbo. Triaxial induction tool with electrode sleeve: FD modeling in 3D geometries. *Journal of Applied Geophysics*, 67:98–108, 2004. (cited in page(s) 4)

[36] S. Davydycheva and T. Wang. A fast modelling method to solve Maxwell's equations in 1D layered biaxial anisotropic medium. *Geophysics*, 76 (5):F293–F302, 2011. (cited in page(s) 4)

[37] L. Demkowicz, P. Monk, L. Vardapetyan, and W. Rachowicz. De Rham diagram for *hp* finite element spaces. *Computers & Mathematics with Applications*, 39(7-8):29–38, 2000. (cited in page(s) 56)

[38] L. Demkowicz, W. Rachowicz, and P. Devloo. A fully automatic hp-adaptivity. *Journal of Scientific Computing*, 17, 2002. (cited in page(s) 83)

[39] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero. Recent advances in deep learning for speech research at microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8604–8608, 2013. (cited in page(s) 3)

[40] R. Desbrandes and R. Clayton. Chapter 9 measurement while drilling. *Developments in Petroleum Science*, 38:251 – 279, 1994. (cited in page(s) 2)

[41] C. Dupuis and J. M. Denichou. Automatic inversion of deep-directional-resistivity measurements for well placement and reservoir description. *The Leading Edge*, 34(5):504–512, 2015. (cited in page(s) 2)

[42] W. Ee, J. Han, and A. Jentzen. Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations. *To appear in Communications in Mathematics and Statistics*, 5, 06 2017. (cited in page(s) 5)

[43] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean. A guide to deep learning in healthcare. *Nature Medicine*, 25:24–29, 2019. (cited in page(s) 3)

[44] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas. Efficient and accurate estimation of Lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32:11427–11438, 2019. (cited in page(s) 84)

[45] D. Fleisch. *A student's guide to Maxwell's equations.* Cambridge University Press, 2008. (cited in page(s) 4)

[46] D. Garcia, D. Pardo, and V. M. Calo. Refined isogeometric analysis for fluid mechanics and electromagnetics. *Computer Methods in Applied Mechanics and Engineering*, 356:598–628, 2019. (cited in page(s) 5, 55, 56, 57, 61)

[47] D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V. M. Calo. The value of continuity: Refined isogeometric analysis and fast direct solvers. *Computer Methods in Applied Mechanics and Engineering*, 316:586–605, 2017. (cited in page(s) 4, 56, 57, 60, 61, 62)

[48] S. Gernez, A. Bouchedda, E. Gloaguen, and D. Paradis. Aim4res, an open-source 2.5D finite differences MATLAB library for anisotropic electrical resistivity modeling. *Computers & Geosciences*, 135:104401, Feb. 2020. (cited in page(s) 4, 49)

[49] M. Ghasemi, Y. Yang, E. Gildin, Y. Efendiev, and V. M. Calo. Fast multiscale reservoir simulations using pod-deim model reduction. *Society of Petroleum Engineers*, pages 1–18, 2015. (cited in page(s) 2)

[50] S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 106:102447, 2020. (cited in page(s) 71)

[51] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021. (cited in page(s) 84)

[52] A. Güneş Baydin, B. A. Pearlmutter, A. Andreyevich Radul, and J. Mark Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18:1–43, 2018. (cited in page(s) 6)

[53] J. Gunning and M. E. Glinsky. Detection of reservoir quality using bayesian seismic inversion. *Geophysics*, 72(3):R37–R49, 2007. (cited in page(s) 3)

[54] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021. (cited in page(s) 3)

[55] J. Hadamard. *Lectures on Cauchy's problem in linear partial differential equations*. Yale University Press, 1923. (cited in page(s) 7)

[56] T. Hageman, K. M. P. Fathima, and R. de Borst. Isogeometric analysis of fracture propagation in saturated porous media due to a pressurised non-Newtonian fluid. *Computers and Geotechnics*, 112:272–283, Aug. 2019. (cited in page(s) 4)

[57] J. Han, A. Jentzen, and W. Ee. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115, 07 2017. (cited in page(s) 5)

[58] J. R. Hauser. *Partial Differential Equations: The Finite Element Method*. Numerical Methods for Nonlinear Engineering Models. Springer Netherlands, 2009. (cited in page(s) 4, 5)

[59] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015. (cited in page(s) 3, 16)

[60] C. F. Higham and D. J. Higham. Deep learning: An introduction for applied mathematicians. *Computing Research Repository*, abs/1801.05894, 2018. (cited in page(s) 4, 16, 17)

[61] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005. (cited in page(s) 4)

[62] C. Huré, H. Pham, and X. Warin. Some machine learning schemes for high-dimensional nonlinear PDEs. *Math. Comput.*, 89:1547–1579, 2020. (cited in page(s) 5)

[63] S. ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4):185–196, 1993. (cited in page(s) 6)

[64] O. Ijasana, C. Torres-Verdín, and W. E. Preeg. Inversion-based petrophysical interpretation of logging-while-drilling nuclear and resistivity measurements. *Geophysics*, 78 (6):D473–D489, 2013. (cited in page(s) 2, 10)

[65] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020. (cited in page(s) 71)

[66] B. Jan, H. Farman, M. Khan, M. Imran, I. U. Islam, A. Ahmad, S. Ali, and G. Jeon. Deep learning in big data analytics: A comparative study. *Computers & Electrical Engineering*, 75:275 – 287, 2019. (cited in page(s) 3)

[67] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017. (cited in page(s) 16)

[68] Y. Jin, X. Wu, J. Chen, Y. Huang, et al. Using a physics-driven deep neural network to solve inverse problems for lwd azimuthal resistivity measurements. In *SPWLA 60th Annual Logging Symposium*. Society of Petrophysicists and Well-Log Analysts, 2019. (cited in page(s) 19)

[69] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Dover Books on Mathematics Series. Dover Publications, Incorporated, 2012. (cited in page(s) 4, 73)

[70] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. (cited in page(s) 58)

[71] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. VPINNs: Variational Physics-Informed Neural Networks For Solving Partial Differential Equations, 2019. (cited in page(s) 5, 6, 71)

[72] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021. (cited in page(s) 71)

[73] R. Khodayi-Mehr and M. Zavlanos. Varnet: Variational neural networks for the solution of partial differential equations. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 298–307, 2020. (cited in page(s) 71)

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. (cited in page(s) 16)

[75] D. Kushnir, N. Velker, A. Bondarenko, G. Dyatlov, and Y. Dashevsky. Real-time simulation of deep azimuthal resistivity tool in 2D fault model using neural networks. In *SPE Annual Caspian Technical Conference and Exhibition*, Oct. 2018. (cited in page(s) 49)

[76] I. Lagaris, A. Likas, and D. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9:987–1000, 1998. (cited in page(s) 76)

[77] R. J. LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems.* SIAM, 2007. (cited in page(s) 4, 5)

[78] D. B. Lindell, J. N. Martel, and G. Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (cited in page(s) 6)

[79] C. R. Liu. *Theory of Electromagnetic Well Logging.* Elsevier, Amsterdam, Netherlands, 2017. (cited in page(s) 52)

[80] L. O. Loseth and B. Ursin. Electromagnetic fields in planarly layered anisotropic media. *Geophysical Journal International*, 170:44–80, 2007. (cited in page(s) 10, 23)

[81] L. Lu, X. Meng, Z. Mao, and G. Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63:208–228, 2021. (cited in page(s) 5)

[82] L. Lu, Y. Zheng, G. Carneiro, and L. Yang. *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Springer, Switzerland, 2017. (cited in page(s) 3)

[83] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11 – 24, 2014. (cited in page(s) 3)

[84] Z. Ma, D. Liu, H. Li, and X. Gao. Numerical simulation of a multi-frequency resistivity logging-while-drilling tool using a highly accurate and adaptive higher-order finite element method. *Advances in Applied Mathematics and Mechanics*, 4(4):439–453, 2012. (cited in page(s) 4)

[85] A. Malinverno and C. Torres-Verdín. Bayesian inversion of DC electrical measurements with uncertainties for reservoir monitoring. *Inverse Problems*, 16(5):1343–1356, oct 2000. (cited in page(s) 3)

[86] Z. Mao, A. D. Jagtap, and G. E. Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020. (cited in page(s) 6, 71)

[87] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis. Ppinn: Parareal physics-informed neural network for time-dependent pdes. *Computer Methods in Applied Mechanics and Engineering*, 370:113250, 2020. (cited in page(s) 71)

[88] S. Mishra and R. Molinaro. Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs. *arXiv preprint arXiv:2006.16144*, 2020. (cited in page(s) 5, 84)

[89] D. Moghadas. One-dimensional deep learning inversion of electromagnetic induction data using convolutional neural network. *Geophysical Journal International*, 222(1):247–259, 2020. (cited in page(s) 16)

[90] P. Moin. *Fundamentals of Engineering Numerical Analysis*. Cambridge University Press, 2010. (cited in page(s) 77)

[91] D. Mortari. Least-Squares Solution of Linear Differential Equations. *Mathematics*, 5, 02 2017. (cited in page(s) 73)

[92] M. J. Nam, D. Pardo, and C. Torres-Verdín. Simulation of borehole-eccentered triaxial induction measurements using a Fourier *hp* finite-element method. *Geophysics*, 78(1):D41–D52, Jan. 2013. (cited in page(s) 4, 49)

[93] D. M. Nguyen, A. Evgrafov, and J. Gravesen. Isogeometric shape optimization for electromagnetic scattering problems. *Progress In Electromagnetics Research B*, 45:117–146, 2012. (cited in page(s) 4)

[94] V. P. Nguyen, C. Anitescu, S. P. Bordas, and T. Rabczuk. Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation*, 117:89–116, 2015. (cited in page(s) 5)

[95] C. M. B. Nunes and C. Régis. GEMM3D: An edge finite element program for 3D modeling of electromagnetic fields and sensitivities for geophysical applications. *Computers & Geosciences*, 139:104477, June 2020. (cited in page(s) 4)

[96] G. Pang, L. Lu, and G. E. Karniadakis. fPINNs: Fractional Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019. (cited in page(s) 5)

[97] D. Pardo, L. Demkowicz, C. Torres-Verdín, and M. Paszynski. Two-dimensional high-accuracy simulation of resistivity logging-while-drilling (LWD) measurements using a self-adaptive goal-oriented *hp* finite element method. *SIAM Journal on Applied Mathematics*, 66(6):2085–2106, Jan. 2006. (cited in page(s) 4)

[98] D. Pardo, P. J. Matuszyk, V. Puzyrev, C. Torres-Verdin, M. J. Nam, and V. M. Calo. *Modeling of Resistivity and Acoustic Borehole Logging Measurements Using Finite Element Methods*. Elsevier, 2021. (cited in page(s) 4)

[99] D. Pardo and C. Torres-Verdin. Fast 1D inversion of logging-while-drilling resistivity measurements for the improved estimation of formation resistivity in high-angle and horizontal wells. *Geophysics*, 80 (2):E111–E124, 2014. (cited in page(s) 2, 4, 10)

[100] D. Pardo and C. Torres-Verdín. Fast 1D inversion of logging-while-drilling resistivity measurements for improved estimation of formation resistivity in

high-angle and horizontal wells. *Geophysics*, 80(2):E111–E124, Mar. 2015. (cited in page(s) 49)

[101] D. Pardo, C. Torres-Verdín, M. J. Nam, M. Paszynski, and V. M. Calo. Fourier series expansion in a non-orthogonal system of coordinates for the simulation of 3D alternating current borehole resistivity measurements. *Computer Methods in Applied Mechanics and Engineering*, 197(45-48):3836–3849, Aug. 2008. (cited in page(s) 4, 49)

[102] J. A. Parker, R. V. Kenyon, and D. E. Troxel. Comparison of interpolating methods for image resampling. *IEEE Transactions on Medical Imaging*, 2(1):31–39, 1983. (cited in page(s) 17)

[103] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. (cited in page(s) 3)

[104] M. Paszyński, R. Grzeszczuk, D. Pardo, and L. Demkowicz. Deep learning driven self-adaptive hp finite element method. In *Computational Science – ICCS 2021*, pages 114–121. Springer International Publishing, 2021. (cited in page(s) 5)

[105] C. G. Petra, O. Schenk, and M. Anitescu. Real-time stochastic optimization of complex energy systems on high-performance computers. *Computing in Science & Engineering*, 16(5):32–42, 2014. (cited in page(s) 58)

[106] C. G. Petra, O. Schenk, M. Lubin, and K. Gäertner. An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization. *SIAM Journal on Scientific Computing*, 36(2):C139–C162, 2014. (cited in page(s) 58)

[107] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag, New York, NY, 2nd edition, 1997. (cited in page(s) 54)

[108] S. Purushotham, C. Meng, Z. Che, and Y. Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of Biomedical Informatics*, 83:112–134, 2018. (cited in page(s) 3)

[109] V. Puzyrev. Deep learning electromagnetic inversion with convolutional neural networks. *Geophysical Journal International*, 218(2):817–832, 2019. (cited in page(s) 16)

[110] T. M. Quan, D. G. C. Hildebrand, and W.-K. Jeong. Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics, 2016. (cited in page(s) 16)

[111] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville. On the Spectral Bias of Neural Networks, 2019. (cited in page(s) 5)

[112] M. Raissi, P. Perdikaris, and G. Karniadakis. 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, 2019. (cited in page(s) 6, 71)

[113] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. 2017. (cited in page(s) 5)

[114] S. Ranjan and S. Senthamilarasu. *Applied Deep Learning and Computer Vision for Self-Driving Cars: Build autonomous vehicles using deep neural networks and behavior-cloning techniques.* Packt Publishing, 2020. (cited in page(s) 3)

[115] J. N. Reddy. *Introduction to the finite element method.* McGraw-Hill Education, 2019. (cited in page(s) 4)

[116] W. Ritz. Über eine neue methode zur lösung gewisser variationsprobleme der mathematischen physik. *Journal für die reine und angewandte Mathematik*, 135:1–61, 1909. (cited in page(s) 73)

[117] J. A. Rivera, D. Pardo, and E. Alberdi. Design of loss functions for solving inverse problems using deep learning. In *Computational Science – ICCS 2020*, pages 158–171. Springer International Publishing, 2020. (cited in page(s) 3)

[118] J. A. Rivera, J. M. Taylor, Ángel J. Omella, and D. Pardo. On quadrature rules for solving partial differential equations using neural networks. *Computer Methods in Applied Mechanics and Engineering*, 393:114710, 2022. (cited in page(s) 5)

[119] Á. Rodríguez-Rozas and D. Pardo. A priori Fourier analysis for 2.5D finite elements simulations of logging-while-drilling (LWD) resistivity measurements. *Procedia Computer Science*, 80:782–791, 2016. (cited in page(s) 58)

[120] Á. Rodríguez-Rozas, D. Pardo, and C. Torres-Verdín. Fast 2.5D finite element simulations of borehole resistivity measurements. *Computational Geosciences*, 22(5):1271–1281, 2018. (cited in page(s) 4, 52, 53, 58, 60, 65)

[121] L. Ruthotto and E. Haber. Deep Neural Networks Motivated by Partial Differential Equations. *Journal of Mathematical Imaging and Vision*, 62, 2020. (cited in page(s) 5)

[122] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8, 2020. (cited in page(s) 71)

[123] E. Samaniego, C. Anitescu, S. Goswami, V. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362:112790, 2020. (cited in page(s) 5)

[124] A. Sarmiento, A. Côrtes, D. Garcia, L. Dalcin, N. Collier, and V. Calo. PetIGA-MF: A multi-field high-performance toolbox for structure-preserving B-splines spaces. *Journal of Computational Science*, 18:117–131, 2017. (cited in page(s) 57)

[125] K. Scaman and A. Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3839–3848, 2018. (cited in page(s) 84)

[126] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487, Apr. 2004. (cited in page(s) 58)

[127] O. Schenk, K. Gärtner, and W. Fichtner. Efficient sparse LU factorization with left-right looking strategy on shared memory multiprocessors. *BIT Numerical Mathematics*, 40(1):158–176, 2000. (cited in page(s) 58)

[128] D. J. Seifert, S. A. Dossary, R. E. Chemali, M. S. Bittar, A. A. Lotfy, J. L. Pitcher, and M. A. Bayrakdar. Deep electrical images, geosignal, and real-time inversion help guide steering decisions. *Society of Petroleum Engineers*, pages 1–9, 2009. (cited in page(s) 2)

[129] M. Shahriari and D. Pardo. Borehole resistivity simulations of oil-water transition zones with a 1.5D numerical solver. *Computational Geosciences*, 24(3):1285–1299, Mar. 2020. (cited in page(s) 4, 49)

[130] M. Shahriari, D. Pardo, B. Moser, and F. Sobieczky. A deep neural network as surrogate model for forward simulation of borehole resistivity measurements. *Procedia Manufacturing*, 42:235 – 238, 2020. International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019). (cited in page(s) 4, 13)

[131] M. Shahriari, D. Pardo, A. Picón, A. Galdran, J. D. Ser, and C. Torres-Verdín. A deep learning approach to the inversion of borehole resistivity measurements. *Computational Geosciences*, 2020. (cited in page(s) 2, 4, 13)

[132] M. Shahriari, D. Pardo, J. Rivera, C. Torres-Verdín, A. Picon, J. Del Ser, S. Ossandón, and V. Calo. Error control and loss functions for the deep learning inversion of borehole resistivity measurements. *International Journal for Numerical Methods in Engineering*, 122, 2020. (cited in page(s) 3)

[133] M. Shahriari, S. Rojas, D. Pardo, A. Rodríguez-Rozas, S. A. Bakr, V. M. Calo, and I. Muga. A numerical 1.5D method for the rapid simulation of geophysical resistivity measurements. *Geosciences*, 8(6):1–28, 2018. (cited in page(s) 2, 4, 8, 10)

[134] S. Shahrokhabadi, T. D. Cao, and F. Vahedifard. Isogeometric analysis through Bézier extraction for thermo-hydro-mechanical modeling of saturated porous media. *Computers and Geotechnics*, 107:176–188, Mar. 2019. (cited in page(s) 4)

[135] J. Shen and W. Sun. 2.5-D modeling of cross-hole electromagnetic measurement by finite element method. *Petroleum Science*, 5(2):126–134, May 2008. (cited in page(s) 4, 49)

[136] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *Journal of Nondestructive Evaluation*, 39(3):61, 2020. (cited in page(s) 71)

[137] A. Simona, L. Bonaventura, C. de Falco, and S. Schöps. IsoGeometric approximations for electromagnetic problems in axisymmetric domains. *Computer Methods in Applied Mechanics and Engineering*, 369:113211, Sept. 2020. (cited in page(s) 4)

[138] R. N. Simpson, Z. Liu, R. Vázquez, and J. A. Evans. An isogeometric boundary element method for electromagnetic scattering with compatible B-spline discretizations. *Journal of Computational Physics*, 362:264–289, June 2018. (cited in page(s) 4)

[139] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339 – 1364, 2018. (cited in page(s) 6, 71)

[140] G. D. Smith, G. D. Smith, and G. D. S. Smith. *Numerical solution of partial differential equations: finite difference methods.* Oxford university press, 1985. (cited in page(s) 4)

[141] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation.* SIAM, 2005. (cited in page(s) 2)

[142] P. Thévenaz, T. Blu, and M. Unser. Chapter 28-image interpolation and resampling. In I. Bankman, editor, *Handbook of MedicalImage Processing and Analysis, 2nd ed*, volume 4, pages 465–493. Academic Press: Cambridge, MA, USA, 2009. (cited in page(s) 17)

[143] B. Ursin and A. Stovas. Reflection and transmission responses of a layered isotropic viscoelastic medium. *Geophysics*, 67(1):307–323, 2002. (cited in page(s) 24)

[144] R. Vargas, A. Mosavi, and R. Ruiz. Deep learning: A review. Working paper, January 2017. (cited in page(s) 3)

[145] C. Vogel. *Computational Methods for Inverse Problems.* SIAM, 2002. (cited in page(s) 2)

[146] H. Wang, G. Tao, and K. Zhang. Wavefield simulation and analysis with the finite-element method for acoustic logging while drilling in horizontal and deviated wells. *Geophysics*, 78(6):D525–D543, Nov. 2013. (cited in page(s) 4)

[147] L. Wang, H. Li, and Y. Fan. Bayesian inversion of logging-while-drilling extra-deep directional resistivity measurements using parallel tempering markov chain monte carlo sampling. *IEEE Transactions on Geoscience and Remote Sensing*, 57(10):8026–8036, 2019. (cited in page(s) 3)

[148] S. Wang, X. Yu, and P. Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective, 2020. (cited in page(s) 5)

[149] E. Weinan and Y. Bing. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Communications in Mathematics and Statistics*, 6:1–12, 2018. (cited in page(s) 6, 71)

[150] S. Weinzierl. Introduction to Monte Carlo Methods. *ArXiv High Energy Physics - Phenomenology e-prints*, 07 2000. (cited in page(s) 81)

[151] T. Wiatowski and H. Bölcskei. A mathematical theory of deep convolutional neural networks for feature extraction. *IEEE Transactions on Information Theory*, 64(3):1845–1866, 2018. (cited in page(s) 16)

[152] K. Wu and D. Xiu. Data-driven deep learning of partial differential equations in modal space. *Journal of Computational Physics*, 408:109307, 2020. (cited in page(s) 5)

[153] L. Yang, X. Meng, and G. E. Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021. (cited in page(s) 71)

[154] D. Yu and L. Deng. *Automatic Speech Recognition: A Deep Learning approach*. Springer, London, 2017. (cited in page(s) 3)

[155] Z. Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2, 2018. (cited in page(s) 88)

[156] Z. Zhang, N. Yuan, and C. R. Liu. 1-D inversion of triaxial induction logging in layered anisotropic formation. *Progress In Electromagnetics Research B*, 44:383–403, 2012. (cited in page(s) 2)

[157] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213 – 237, 2019. (cited in page(s) 3)

[158] J. Zhou. *LWD/MWD Resistivity Tool Parameters.* Maxwell Dynamics, 2016. (cited in page(s) 59)