SOLVING RAY-TRACING EQUATIONS WITH NEURAL NETWORKS

by

Selin Aktaş

B.S., Mathematics, Boğaziçi University, 2022

Submitted to Kandilli Observatory and Earthquake
Research Institute in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Geophysics Department
Boğaziçi University
2024

SOLVING RAY-TRACING EQUATIONS WITH NEURAL NETWORKS

APPROVED BY:

Assist. Prof. Çağrı Diner (Thesis Supervisor)	....
Prof. Dr. Bertan Badur	....
Assoc. Prof. Ayberk Zeytin	....

DATE OF APPROVAL: 10.06.2024

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Çağrı Diner for their invaluable guidance, support, and insightful mentorship throughout the entirety of this research endeavor. Their expertise and encouragement have been instrumental in shaping the course of this thesis.

Special thanks are extended to Yusuf Sezer Car and Erdem Ata for their collaboration in the development of the code essential to the implementation of neural network solutions in ray-tracing equations. Their technical expertise and collaborative spirit have significantly enriched the scope and effectiveness of this research.

I extend heartfelt appreciation to my brother Sercan Aktaş for his invaluable contributions in creating the figures that enhance the visual representation of this thesis. His artistic talent have added a unique dimension to the presentation of results.

To my family, my boyfriend and my friends, I owe a debt of gratitude for their unconditional love, encouragement, and support throughout this academic journey. Their understanding and belief in my pursuits have been a constant source of motivation.

To all those who have been a part of this academic pursuit, providing guidance, encouragement, and support, I extend my heartfelt appreciation. This achievement is a collective effort, and I am truly fortunate to have such an incredible network of individuals who have contributed to the realization of this thesis.

ABSTRACT

SOLVING RAY-TRACING EQUATIONS WITH NEURAL NETWORKS

This thesis presents an innovative approach for solving ray-tracing equations using neural networks, offering faster and more accurate solutions. The primary objective is to solve ray-tracing equations with neural networks, however, we decided to initially examine the capabilities of the network in addressing a one-dimensional problem, such as the harmonic oscillator. Firstly, we compare various neural network models for solving the harmonic oscillator equation. These include conventional neural networks, physics-informed neural networks (PINNs), and a combined model the Hamiltonian approach with the PINN. Conventional neural networks require balanced data to solve the problem and struggle when data is not provided. Conversely, PINNs are able to learn successfully with a small dataset by satisfying physical equations, for both provided and unprovided data regions. However, the aim is to solve these equations with only initial conditions, without providing any additional data. To achieve this, a neural network model integrating the Hamiltonian approach with PINN is introduced. This integrated model mainly focuses on energy conservation, enabling the model to find solutions using initial conditions. The application of this thesis is solving ray-tracing equations in linearly heterogeneous media, both isotropic and anisotropic using the integrated model. We solve ray-tracing equations for an isotropic continuum in the forward problem by providing parameters like velocity, take-off angle, and time. The model efficiently determines seismic wave ray paths. In the inverse problem, we aim to determine the angle and velocity model for an isotropic continuum by providing receiver and source locations, and travel time. We then progressively increase the complexity of the problem, starting with isotropic velocity model and advancing to constant elliptically and to generalized elliptically anisotropic models.

ÖZET

SİNİR AĞLARI İLE SİSMİK IŞIN YOL DENKLEMLERİNİN ÇÖZÜMÜ

Bu tez, sinir ağları kullanarak sismik ışın yol denklemlerini çözmek için daha hızlı ve daha doğru çözümler sunan yenilikçi bir yaklaşım sunmaktadır. Birincil amaç bu denklemleri sinir ağları ile çözmektir, ancak başlangıçta ağın harmonik osilatör gibi tek boyutlu bir problemi ele alma yeteneklerini incelemeye karar verdik. İlk olarak, harmonik osilatör denklemini çözmek için çeşitli sinir ağı modellerini karşılaştırıyoruz. Bunlar arasında geleneksel sinir ağları, fizik bilgisine sahip sinir ağları (PINN) ve PINN ile birleştirilmiş bir model olan Hamiltonian yaklaşımı yer almaktadır. Geleneksel sinir ağları problemi çözmek için dengeli verilere ihtiyaç duyar ve veri sağlanmadığında zorlanır. Buna karşılık, PINN, hem sağlanan hem de sağlanmayan veri bölgeleri için fiziksel denklemleri karşılayarak küçük bir veri kümesiyle başarılı bir şekilde öğrenebilir. Ancak amaç, bu denklemleri herhangi bir ek veri sağlamadan yalnızca başlangıç koşullarıyla çözmektir. Bunu başarmak için, Hamilton yaklaşımını PINN ile bütünleştiren bir sinir ağı modeli tanıtılmıştır. Bu entegre model temel olarak enerji korunumuna odaklanmakta ve modelin başlangıç koşullarını kullanarak çözüm bulmasını sağlamaktadır. Bu tezin uygulaması, entegre model kullanılarak hem izotropik hem de anizotropik doğrusal heterojen ortamlarda sismik ışın yol denklemlerinin çözülmesidir. İzotropik bir süreklilik için sismik ışın yol denklemlerini hız, kırılma açısı ve zaman gibi parametreler sağlayarak ileri problemde çözüyoruz. Model, sismik ışın yolunu verimli bir şekilde belirler. Ters problemde istasyon konumu, deprem kaynağı ve süreyi vererek izotropik bir süreklilik için açı ve hız modelini belirlemeyi amaçlıyoruz. Daha sonra, izotropik hız modelinden başlayarak ve heterojen ortamda sabit eliptik ve genelleştirilmiş eliptik anizotropik modellere ilerleyerek problemin karmaşıklığını aşamalı olarak artırıyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

A	Amplitude
a	Acceleration
a_{ij}	The element in the i^{th} row and j^{th} column of a matrix A.
b	Bias vector
c_{ijkl}	Fourth-order elastic stiffness tensor
det	Determinant
e	Euler number
$e^{i\omega\psi(x)}$	Complex exponential function
F	Force
$\mathcal{H}$	Hamilton
$\mathbf{h}^{[l]}$	Output of l^{th} hidden layer of deep neural network
J	Cost function
k	Spring constant
$\mathcal{L}$	Loss function
L	Lagrangian
m	Mass
p	Phase-slowness vector
p	Momentum
q	Coordinates
t	Time
t_{max}	Maximum time
T	Kinetic energy
u	Displacement
v	Speed
V	Potential energy
w	Weight matrix
$\mathbf{x}$	Input vector
$X_{\mathcal{H}}$	Symplectic gradient vector field along $\mathcal{H}$

$\mathbf{y}$	Output vector
$\hat{\mathbf{y}}$	Predicted output vector
z	Preactivation value
M^T	Transpose of a matrix M
$\frac{d}{dt}$	First derive with respect to time
∞	Infinity
α	Learning rate
χ	Parameter that indicates degree of anisotropy
δ	Kronecker delta
$\frac{\partial}{\partial t}$	Partial derivative with respect to time
$\frac{\partial^2}{\partial t^2}$	Second order partial derivative with respect to time
ε	Strain tensor
λ	Lame's first parameter
μ	Shear modulus
∇	Gradient
$\nabla\cdot$	Divergence
$\nabla\times$	Curl
∇^2	Laplace operator
ϕ	Phase angle
$\mathit{\Phi}(\mathbf{z})$	Activation function
$\mathit{\Psi}$	Rotation vector
ρ	Density
σ	Stress tensor
$\sum$	Summation
φ	Dilation
ϑ	Take-off angle
ω	Angular frequency

LIST OF ACRONYMS/ABBREVIATIONS

Adam	Adaptive Moment Estimation
AdaGrad	Adaptive Gradient Algorithm
Autograd	Automatic calculation of the gradients
HNN	Hamiltonian Neural Networks
IC	Initial Condition
MSE	Mean Squared Error
MSELoss	Mean Squared Loss
NN	Neural Networks
PDE	Partial Differential Equations
PINN	Physics-Informed Neural Networks
RMSProp	Root Mean Square Propagation
SGD	Stochastic Gradient Descent
tanh	Hyperbolic Tangent Function

1. INTRODUCTION

Ray-tracing equations are an important mathematical tool for predicting the propagation of seismic waves and modelling their paths through the layers of the Earth. These equations derived from the eikonal equation, which include Snell's law, are differential equations that take into account factors such as velocity model of the medium. In nineteenth century, mathematical foundations of classical mechanics were laid by Sir William Rowan Hamilton. Ray theory is a specific application of classical mechanics, which is solved by using Hamilton's equations of motion. The techniques used today to solve ray-tracing equations, such as finite difference and finite element methods, have been considerably enhanced by the contributions of Vlastislav Červený [1].

Artificial neural networks play a pivotal role in numerous research and innovation today. Historically, the first mathematical model of artificial neural networks was developed in 1943 by Warren McCulloch and Walter Pitts [2]. They described a simple and powerful computational model inspired by the structure of the human brain. Over the years, neural networks have evolved significantly. The real breakthrough occurred with the revelation of **the universal approximation theorem** of neural networks [3]. This theorem states that neural networks have the remarkable capability to approximate any function with a high level of accuracy, marking a significant turning point for neural networks. In the present day, neural networks have proven to be highly effective in addressing complex problems [4]. They are particularly skilled at solving complex problems defined by differential equations.

Over time, neural networks have evolved into different types, among which physics-informed neural networks (PINNs) are the most important for scientific studies. The PINN algorithm is a scientific machine learning technique introduced by Raissi, Perdikaris and Karniadakis in 2017 [5], combining the strengths of physics-based models and machine learning techniques to solve partial differential equations. It transforms the loss function into an optimization problem [6], allowing PDEs to be solved more accurately and efficiently compared to traditional numerical methods. The study conducted by Raissi, Perdikaris, and Karniadakis in 2019 [7] demonstrates that PINNs are versatile tools that can be used for both forward and inverse problems. They also require less training data compared to conventional neural networks, making them effective for solving partial differential equations using minimal data [8].

Another important type of neural network for physical problems is Hamiltonian neural networks (HNNs), which were first published in 2019 by Greydanus, Dzamba, and Yosinski [9]. HNNs are designed to preserve the underlying Hamiltonian dynamics of a physical system and to ensure that the neural network learns the physical principles, such as conservation of energy, momentum, and symplectic structure. In comparison to conventional neural network models, HNNs display enhanced long-term stability, faster training, and superior generalization. Moreover, the HNN model is time-invariant, as evidenced by the elimination of time derivatives and time dependence through the use of Hamiltonian equations and automatic differentiation [10], which results in a time-invariant energy. In 2022, Mattheakis, Sondak, Dogra, and Protopapas [4] presented a Hamiltonian neural network that solves the differential equations that govern dynamical systems. This method relies on equations rather than observational data for optimization, making it a unique equation-driven approach to machine learning.

Solving PDEs with NNs instead of traditional numerical methods is more advantageous. Some of these advantages are mentioned in the paragraphs above, but here we bring these together. Finite difference and finite element methods require a defined mesh, but PINNs are mesh-free allowing for simplification of complex geometries. NNs excel at processing high-dimensional data, learn complex patterns from data and generalize to unseen data. PINNs can incorporate both physical principles and observational data. This makes them highly adaptable to various PDEs and boundary conditions. NNs can use the automatic differentiation algorithm, which allows efficient calculation of gradients with optimization techniques to find optimal solutions to the PDE. In addition, NNs can approximate continuous and smooth functions to a high degree of accuracy.

Although ray-tracing equations dates back to the nineteenth century, it remains a fundamental tool in seismology for imaging the Earth's interior and interpreting seismic data. With this thesis, we aim to contribute to the methods of solving ray-tracing equations by adding a neural network solution method. Ray-tracing equations are PDEs by the nature of these equations. Since it is advantageous to solve PDEs with NNs, applying them to ray-tracing equations improves both the accuracy and the computational efficiency. To find the best neural network model, we create three types of neural network models. These include a conventional NN, a PINN, and a model that combines Hamiltonian approach with a PINN. With combining Hamiltonian approach and PINNs, we wish to obtain the solution without mass data, only with boundary conditions, solving ray-tracing equations with physics loss and energy loss minimization. After comparing each model in the harmonic oscillator problem, the insights gained from this comparison can then be employed to effectively solve the ray-tracing equations.

This thesis is structured as follows: In Chapter 2, the mathematical derivation of the harmonic oscillator and ray-tracing equations is explained. Additionally, the physical interpretations of these equations are discussed. In this chapter, Michael A. Slawinski's book "Waves and Rays in Elastic Continua" [11] was utilized especially in obtaining the equations. Chapter 3 introduces the fundamental concepts of neural networks. The following sections explain how a neural network is trained, the functions utilized during this process, and various regularization techniques. These concepts and techniques form the basis of physics-informed neural networks. In Chapter 4, the principles of PINNs are explained in detail and mentioned Hamiltonian neural networks in this chapter. The initial approach involved classical NNs to solve harmonic oscillator equations with training data. Consequently, a switch was made to PINNs, with the objective of solving the same problem while attempting to fit the physical equation. Therefore, the Hamiltonian approach was incorporated into the PINN model, which conserves physical quantites such as energy. After solving the harmonic oscillator equation, we focus on solving the ray-tracing equations in a linearly heterogeneous medium using forward and inverse problems, which is the main focus of this thesis. A combined model of HNNs and PINNs is used to solve these problems. In the forward model, the ray-tracing equations are solved by providing parameters such as velocity structure, take-off angle, and time. In the inverse model, the angle and velocity model is determined by providing the receiver and source locations, and travel time. Subsequently, we enhance the complexity of the velocity model by employing constant and generalized elliptical anisotropic velocity models. Chapter 5 is the conclusion, which discusses the results of these analyses.

2. PARTIAL DIFFERENTIAL EQUATIONS

Partial differential equations, abbreviated as PDEs, are mathematical equations that describe how a physical quantity changes in space and time. Unlike ordinary differential equations, which involve a single independent variable, partial differential equations account for multiple independent variables. This enables a more comprehensive understanding of complex systems. PDE's are used to analyze and predict the behavior of complex systems by expressing the relationships between the variables and their rates of change. PDE's have a significant role in seismology, particularly in modeling wave propagation and applying ray-tracing techniques. The wave equation, which is a PDE, captures the complex interaction of factors such as wave speed, density, and time in the study of seismic waves. The eikonal equation, which is derived from the wave equation using high frequency approximation, helps ray-tracing by computing travel times along varying subsurface velocities. Additionally, PDEs enable seismologists to make predictions about subsurface structures and facilitate the interpretation of seismic data by creating accurate mathematical representations of wave dynamics. The precision provided by PDE-based models makes the complexity of wave propagation in seismology solvable.

The aim of this chapter is to derive two differential equations solved with physics-informed neural networks in the subsequent chapters. The first of these equations is the harmonic oscillator, which is a second-order ordinary differential equation. The second is the ray-tracing equations in seismology, which is a PDE. The derivation of the harmonic oscillator equation will be shown first. Then, the derivation of the ray-tracing equation will commence with the wave equation, the eikonal equation, and finally the ray-tracing equation. The equations used in this work were derived from the book "Waves and Rays in Elastic Continua" written by Michael Slawinski [11].

2.1. Derivation of Harmonic Oscillator

The harmonic oscillator is one of the most fundamental models in physics, underlying many physical phenomena. This section will derive this model using two distinct mechanics: Newtonian mechanics and Hamiltonian mechanics.

2.1.1. Harmonic Oscillator in Newtonian Mechanics

In Newtonian mechanics, a simple harmonic oscillator system is described by the motion of a mass m being pulled and then released by a force F over a displacement represented by x. The relation between them is shown by Hooke's law ($F = -kx$), where k is the spring constant. The derivation of the harmonic oscillator equation begins by combining Newton's second law ($F = ma$) and Hooke's law.

$$-kx = ma, \tag{2.1}$$

where k is the spring constant, x is the displacement, m is the mass. Acceleration, denoted as a, represents the second derivative of displacement x respect to time, denoted as $a = \ddot{x}$. Substituting this relationship into Equation 2.1 and dividing both sides by mass m, we derive the equation:

$$m\ddot{x} + kx = 0. \tag{2.2}$$

This is a second-order linear homogeneous ordinary differential equation. The general solution of the motion of a harmonic oscillator can be expressed as:

$$x(t) = A\cos(\omega t + \phi). \tag{2.3}$$

Here A denotes the amplitude, ω represents the angular frequency, t represents time, ϕ represents the phase angle. The solution exhibits a sinusoidal pattern that characterizes how the system oscillates. The first and the second derivatives of the system are:

$$\dot{x}(t) = -A\omega\sin(\omega t + \phi), \tag{2.4}$$

$$\ddot{x}(t) = -A\omega^2\cos(\omega t + \phi). \tag{2.5}$$

Substituting these values into Equation 2.2, we obtain:

$$m - A\omega^2 \cos(\omega t + \phi) + kA\cos(\omega t + \phi) = 0. \tag{2.6}$$

After simplification of the terms A and $\cos(\omega t + \phi)$, the following result is obtained:

$$-m\omega^2 + k = 0, \tag{2.7}$$

Consequently, the angular frequency can be expressed as follows:

$$\omega = \sqrt{\frac{k}{m}}. \tag{2.8}$$

Thus, we have derived the equations of the harmonic oscillator in Newtonian mechanics.

2.1.2. Harmonic Oscillator in Hamiltonian Mechanics

Hamiltonian mechanics was developed by Sir William Hamilton in the nineteenth century as a mathematical reformulation of classical mechanics. It provides valuable information about the behavior of various physical systems. In Hamiltonian mechanics, position and momentum are the fundamental variables to describe the state of a system.

The Hamiltonian function represents the total energy of a dynamical system. It is defined as the sum of the kinetic energy associated with the motion of the mass T and potential energy stored in the spring-mass system V, expressed as follows:

$$\mathcal{H} = T + V, \tag{2.9}$$

where $\mathcal{H}$ denotes the Hamiltonian. Kinetic energy T and potential energy V are defined as:

$$T = \frac{1}{2}m\dot{q}^2, \tag{2.10}$$

$$V = \frac{1}{2}kq^2. \tag{2.11}$$

Here, m is the mass at the end of a frictionless spring, k is the spring constant, q is the displacement from equilibrium of the mass, and $\dot{q}$ is the velocity of the mass.

The Lagrangian, denoted as L, comprises the system's dynamics by combining its kinetic and potential energies. It is defined as :

$$L = T - V. \tag{2.12}$$

Substituting T and V, we obtain:

$$L(\dot{q}, q, t) = \frac{1}{2}m\dot{q}^2 - \frac{1}{2}kq^2. \tag{2.13}$$

The momentum p conjugate to q is defined as:

$$p_i = \frac{\partial L}{\partial \dot{q}_i} = m\dot{q}_i, \tag{2.14}$$

where q_i are the generalized coordinates, and p_i are the corresponding momenta.

The Lagrangian approach naturally leads to the Hamiltonian through the Legendre transformation. The Legendre transformation allows to turn the set of Lagrangian's variables into the set of Hamiltonian variables. The generalized Hamilton equations are:

$$\mathcal{H}(q, p, t) = \sum_i q_i p_i - L(\dot{q}_i, q_i, t). \tag{2.15}$$

Substituting the equation for the momenta from equation 2.14 into the kinetic energy equation from 2.10, and subsequently combining it with the equation 2.9, we obtain the Hamiltonian for the harmonic oscillator system as:

$$\mathcal{H} = \frac{p^2}{2m} + \frac{kq^2}{2}. \tag{2.16}$$

Therefore, Hamilton's equations of motion for a simple harmonic oscillator are expressed as:

$$\dot{q}_i = \frac{\partial \mathcal{H}}{\partial p_i} = \frac{p}{m}, \tag{2.17}$$

$$\dot{p}_i = -\frac{\partial \mathcal{H}}{\partial q_i} = -kq. \tag{2.18}$$

Consequently, the harmonic oscillator equations in Hamiltonian mechanics are derived. These equations are subsequently solved by neural networks in the following chapters.

2.2. Derivation of Wave Equation

A wave equation is defined as a partial differential equation describing the behaviour of waves in a physical medium. Wave equations play a pivotal role in understanding the propagation of seismic waves through the subsurface. These equations help to understand how seismic waves behave as they travel through different geological layers.

In isotropic and homogeneous continuum, wave equations govern the propagation of waves, which can be solved analytically. However, for an anisotropic heterogeneous continua, it is impossible to find analytical solutions for the equations of motion [11]. Instead, the high-frequency approximation is used. This approach allows the behavior of rays, wave fronts, travel times, and signal amplitudes to be studied in such continua. Although the resulting expressions are only exact for infinitely high-frequency signals, the experimental results agree well with the theoretical predictions as long as the properties of the continuum remain relatively unchanged over the wavelength of the signal. In this section, the derivation of the wave equations for both continuum will be shown.

2.2.1. Wave Equation for Isotropic and Homogeneous Medium

The aim now is to show the derivation of the wave equation for a given three-dimensional continuum that is isotropic and homogeneous. To briefly mention this continuum's features; isotropy means that the material properties remain constant in all directions, and homogeneity refers to the constancy of physical properties of the medium at any given point in space. Since the medium is isotropic and homogeneous, ρ remains constant [11]. Under these conditions, the derivation of the Cauchy equation of motion will begin by establishing the relationship between force and stress tensor. The force F can be expressed using the gradient of the stress tensor σ_{ij} as:

$$F = \nabla \cdot \sigma_{ij} = \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j}, \tag{2.19}$$

where $i, j \in \{1, 2, 3\}$. Here, ∇ represents the gradient operator.

Considering Newton's second law of motion ($F = ma$), we can utilize the previously derived equation for force. Instead of mass m, we will introduce density ρ, which represents mass per unit volume. Furthermore, we will substitute the second derivative of displacement u for acceleration a. Combining these elements, we arrive at the Cauchy equation of motion:

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j}. \tag{2.20}$$

Substituting the isotropic version of Hooke's law, that is:

$$\sigma_{ij} = c_{ijkl}\epsilon_{kl}. \tag{2.21}$$

By substituting the isotropic form of Hooke's law into the Equation 2.20, we obtain the following result:

$$\rho \frac{\partial^2}{\partial t^2} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = (\lambda + \mu) \begin{bmatrix} \dfrac{\partial(\nabla \cdot u)}{\partial x_1} \\ \dfrac{\partial(\nabla \cdot u)}{\partial x_2} \\ \dfrac{\partial(\nabla \cdot u)}{\partial x_3} \end{bmatrix} + \mu \nabla^2 \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}. \tag{2.22}$$

Here, $\nabla \cdot \mathbf{u}$ represents the divergence of $\mathbf{u}$, expressed mathematically as:

$$\nabla \cdot \mathbf{u} = \sum_{j=1}^{3} \frac{\partial u_j}{\partial x_j}, \tag{2.23}$$

and ∇^2 denotes the Laplace operator, defined as:

$$\nabla^2 = \nabla \cdot \nabla = \sum_{j=1}^{3} \frac{\partial^2}{\partial x_j^2}. \tag{2.24}$$

The details of this derivation can be found in Appendix A.1.

Since $\begin{bmatrix} \dfrac{\partial(\nabla \cdot u)}{\partial x_1} \\ \dfrac{\partial(\nabla \cdot u)}{\partial x_2} \\ \dfrac{\partial(\nabla \cdot u)}{\partial x_3} \end{bmatrix}$ is equal to $\nabla(\nabla \cdot \mathbf{u})$, we obtain:

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) + \mu \nabla^2 \mathbf{u}. \tag{2.25}$$

This is called **the equation of motion for isotropic homogeneous continua**.

Now, using the vector identity $\nabla^2\mathbf{u} = \nabla(\nabla\cdot\mathbf{u}) - \nabla\times(\nabla\times\mathbf{u})$ and substituting it to above equation:

$$\rho\frac{\partial^2\mathbf{u}}{\partial t^2} = (\lambda+2\mu)\nabla(\nabla\cdot\mathbf{u}) - \mu\nabla\times(\nabla\times\mathbf{u}). \tag{2.26}$$

Finally, the vector wave equation, also known as the elastodynamic wave equation, is represented in the following form:

$$\rho\frac{\partial^2\mathbf{u}}{\partial t^2} = (\lambda+2\mu)\nabla\varphi - \mu\nabla\times\mathbf{\Psi}. \tag{2.27}$$

Here, the dilation is denoted as $\varphi = \nabla\cdot\mathbf{u}$, representing the change in volume, and the rotation vector is $\mathbf{\Psi} = \nabla\times\mathbf{u}$, signifying the change in shape.

2.2.2. Wave equation for P waves

To derive the wave equation of P waves, we start by taking the divergence of Equation 2.27, which results in:

$$\nabla\cdot\left[\rho\frac{\partial^2\mathbf{u}}{\partial t^2}\right] = (\lambda+2\mu)\nabla\cdot\nabla\varphi - \mu\nabla\cdot\nabla\times\mathbf{\Psi}. \tag{2.28}$$

Since a homogeneous continuum is assumed, the equations $\nabla\cdot\nabla\times\mathbf{\Psi} = 0$ and $\nabla\cdot\mathbf{u} = \varphi$ hold. Furthermore, by employing the equality of the Laplace operator, namely, $\nabla^2 = \nabla\cdot\nabla$, the above equation is transformed into:

$$\rho\frac{\partial^2\varphi}{\partial t^2} = (\lambda+2\mu)\nabla^2\varphi. \tag{2.29}$$

The propagation speed of P waves v is defined as:

$$v = \sqrt{\frac{(\lambda+2\mu)}{\rho}}. \tag{2.30}$$

Hence, the wave equation for P waves is written as:

$$\nabla^2\varphi = \frac{1}{v^2}\frac{\partial^2\varphi}{\partial t^2}. \tag{2.31}$$

It is important to recall that the scalar wave equation is expressed as follows:

$$\nabla^2\mathbf{u} - \frac{1}{v^2}\frac{\partial^2\mathbf{u}}{\partial t^2} = 0. \tag{2.32}$$

Both the wave equation for P waves and the scalar wave equation share the same structural form and underlying principles.

2.2.3. Wave equation for S waves

To derive the wave equation of S waves, we start by taking the curl of Equation 2.27, which results in:

$$\nabla \times \left[\rho \frac{\partial^2 \mathbf{u}}{\partial t^2}\right] = (\lambda + 2\mu)\nabla \times \nabla \varphi - \mu \nabla \times \nabla \times \mathbf{\Psi}. \tag{2.33}$$

Since a homogeneous continuum is assumed, the equations $\nabla \times \nabla \varphi = 0$ and $\nabla \times \mathbf{u} = \mathbf{\Psi}$ hold. So the above equation is transformed into:

$$\rho \frac{\partial^2 \mathbf{\Psi}}{\partial t^2} = -\mu \nabla \times [\nabla \times \mathbf{\Psi}]. \tag{2.34}$$

Substituting the vector identity into the above equation:

$$\rho \frac{\partial^2 \mathbf{\Psi}}{\partial t^2} = -\mu[\nabla(\nabla \cdot \mathbf{\Psi}) - \nabla^2 \mathbf{\Psi}]. \tag{2.35}$$

The propagation speed of S waves v is defined as:

$$v = \sqrt{\frac{\lambda}{\rho}}. \tag{2.36}$$

Hence the wave equation for S waves is written as:

$$\nabla^2 \mathbf{\Psi} = \frac{1}{v^2} \frac{\partial^2 \mathbf{\Psi}}{\partial t^2}. \tag{2.37}$$

The equation for S waves exhibits a component-wise similarity to the scalar wave equation.

2.2.4. Wave Equation for Anisotropic and Heterogeneous Medium

The continuum is characterized by varying values of ρ and c_{ijkl} across the space, which are dependent on x. The presence of c_{ijkl} indicates that a continuum is anisotropic. Hence, $\rho(x)$ and $c_{ijkl}(x)$ together describe an anisotropic heterogeneous continuum [11]. To derive the wave equation for an anisotropic and heterogeneous continuum, it is necessary to insert the stress-strain equations for the heterogeneity into Cauchy's equation of motion. Remembering these equations, firstly Cauchy's equation of motion:

$$\rho(x) \frac{\partial^2 u_i}{\partial t^2} = \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j}, \tag{2.38}$$

For an heterogeneous continuum, the stress-strain equations can be expressed as:

$$\sigma_{ij} = \frac{1}{2}\sum_{k=1}^{3}\sum_{l=1}^{3} c_{ijkl}(x)\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\right), \tag{2.39}$$

where σ_{ij} represents stress, and $c_{ijkl}(x)$ is the fourth-order elasticity tensor dependent on position x. By inserting Equation 2.39 into Equation 2.38, we derive:

$$\begin{aligned} \rho(x)\frac{\partial^2 u_i}{\partial t^2} &= \sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left[\frac{1}{2}\sum_{k=1}^{3}\sum_{l=1}^{3} c_{ijkl}(x)\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\right)\right] \qquad (2.40)\\ &= \frac{1}{2}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}\left[\frac{\partial c_{ijkl}(x)}{\partial x_j}\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\right) + c_{ijkl}(x)\left(\frac{\partial^2 u_k}{\partial x_j \partial x_l} + \frac{\partial^2 u_l}{\partial x_j \partial x_k}\right)\right]. \end{aligned}$$

This is called **the wave equation for anisotropic and heterogeneous continua**.

2.3. Derivation of Eikonal Equation

The eikonal equation is a type of partial differential equation commonly utilized in ray-tracing to accurately model the travel times of seismic waves through subsurface regions with varying velocities. It describes the travel time of a wave front from a source to a receiver, taking into account the spatial variability of subsurface velocities. Mathematically, the eikonal equation is a first-order nonlinear partial differential equation. By solving this partial differential equation, geophysicists can accurately calculate travel times and predict the geometry of seismic wave fronts. This improves the ability to decipher the composition and structure of the underground. The derivation of the eikonal equation from wave equations will be shown in this section.

First, the eikonal equation for an isotropic and homogeneous continuum will be derived. In order to derive it, weak heterogeneity will be assumed because at high frequencies the weakly heterogeneous case shortens the wavelength, thereby converging to the isotropic homogeneous wave equation. In homogeneous media the speed of propagation of the wave is constant, while in weakly heterogeneous media the speed of propagation of the wave is assumed to vary very slowly. A good approximation to isotropic homogeneity will be obtained by moving to the frequency domain and taking the high frequency approximation. After some calculations, it can be proved that the derived eikonal equation is also applicable to isotropic homogeneous media.

It is necessary to recall the wave equation for an isotropic homogeneous continuum, which is given by the following expression:

$$\nabla^2 \mathbf{u}(x,t) - \frac{1}{v^2(x)} \frac{\partial^2 \mathbf{u}(x,t)}{\partial t^2} = 0, \tag{2.41}$$

where $\mathbf{u}(x,t)$ represents a vector field describing the displacement of waves at position x and time t, and $v(x)$ denotes the wave velocity at position x. Now, transitioning to the frequency domain using the Fourier transform to implement the high-frequency approach, the wave equation in the frequency domain becomes:

$$\nabla^2 \tilde{u}(x,\omega) + \left[\frac{\omega}{v(x)}\right]^2 \tilde{u}(x,\omega) = 0. \tag{2.42}$$

In this equation, $\tilde{u}(x,\omega)$ represents the Fourier transform of the displacement field $\mathbf{u}(x,t)$ with respect to time, taken at angular frequency ω. Since the wave equation is a partial differential equation, it is assumed to have a trial solution to solve it. The general trial solution formulation for the wave equation in the Fourier domain is expressed as:

$$\tilde{u}(x,\omega) = A(x) e^{i\omega\psi(x)}, \tag{2.43}$$

where $\tilde{u}$ is the Fourier's transform of function u, A is the amplitude of the displacement, ω is the frequency, ψ is the eikonal function. Inserting the trial solution into Equation 2.42, we obtain the following result:

$$\frac{\partial^2}{\partial x_i^2} A(x) e^{i\omega\psi(x)} + \left[\frac{\omega}{v(x)}\right]^2 A(x) e^{i\omega\psi(x)} = 0. \tag{2.44}$$

Since the value of $e^{i\omega\psi(x)}$ is nonzero, it can be eliminated from the equation. After elimination of the term, the equation is presented in the following form:

$$\sum_{k=1}^{3} \frac{\partial^2 A}{\partial x_i^2} + A\omega^2 \left[\frac{1}{v^2(x)} - \sum_{k=1}^{3} \frac{\partial \psi}{\partial x_i} \frac{\partial \psi}{\partial x_i}\right] + i\omega \sum_{i=1}^{3} \left(2 \frac{\partial A}{\partial x_i} \frac{\partial \psi}{\partial x_i} + A \frac{\partial^2 \psi}{\partial x_i^2}\right) = 0. \tag{2.45}$$

Utilizing the Laplace operator, the equation can be rewritten as follows:

$$\nabla^2 A + A\omega^2 \left[\frac{1}{v^2(x)} - (\nabla\psi)^2\right] + i\omega(2\nabla A \cdot \nabla\psi + A\nabla^2\psi) = 0. \tag{2.46}$$

Given that Equation 2.46 is equal to zero, it can be concluded that both the real and imaginary parts are equal to zero. The real part will be referred to as the eikonal equation, while the imaginary part is called the transport equation [11]. Considering the eikonal part, if we assume that both ω and A are nonzero, the eikonal equation can be simplified to:

$$\frac{\nabla^2 A}{A\omega^2} + \left[\frac{1}{v^2(x)} - (\nabla\psi)^2\right] = 0. \tag{2.47}$$

Substituting the trial solution and taking the frequency to infinity, the above equation transforms into:

$$\lim_{\omega\to\infty}\left(\frac{\nabla^2 A}{A\omega^2}\right) + \left[\frac{1}{v^2(x)} - (\nabla\psi)^2\right] = 0. \tag{2.48}$$

This can be simplified to:

$$\left[\frac{1}{v^2(x)} - (\nabla\psi)^2\right] = 0, \tag{2.49}$$

which resulted in the following:

$$(\nabla\psi)^2 = \frac{1}{v^2(x)}. \tag{2.50}$$

This equation is the eikonal equation for an isotropic homogeneous continuum. Since the slowness vector of the propagation of wavefront p is normal to the wavefront ($\mathbf{p} = \nabla\psi$), the above equation can also be expressed as:

$$p^2 = \frac{1}{v^2(x)}. \tag{2.51}$$

For all phases, we have the eikonal equation:

$$p^2 = \frac{1}{v^2(\mathbf{x},\mathbf{p})} \tag{2.52}$$

The details of this derivation can be found in Appendix A.2. Replacing the phase-slowness vector with the eikonal function, we obtain:

$$[\nabla\psi(x)]^2 = \frac{1}{v^2(\mathbf{x},\mathbf{p})}. \tag{2.53}$$

This expression gives rise to the eikonal equation, which describes the magnitude of phase slowness based on the properties of an anisotropic inhomogeneous medium. The propagation of the wavefront through such a continuum is explained by this equation. Expressing the gradient $\nabla\psi$ in terms of q_i and p_i:

$$\nabla\psi = \left(\frac{\partial\psi}{\partial x_1}, \frac{\partial\psi}{\partial x_2}, \frac{\partial\psi}{\partial x_3}\right) = (p_1, p_2, p_3). \tag{2.54}$$

This gradient provides information about both the direction and the rate of change of the wave's phase.

2.4. Derivation of Ray-Tracing Equations

In this section, we aim to derive the ray-tracing equations, is also named as Hamilton's ray equations, using the characteristics method of eikonal equation. We begin with the eikonal equation obtained in Equation 2.52:

$$p^2 = \frac{1}{v^2(\mathbf{x},\mathbf{p})} \tag{2.55}$$

This equation directly links the squared phase-slowness vector p^2 with the reciprocal of the squared velocity of a wave v^2. In light of the eikonal equation, $p^2v^2 = 1$, it can be rearranged as follows:

$$p^2v^2 - 1 = 0. \tag{2.56}$$

To normalize time, the equation above is divided by two. This normalization ensures that time scales align appropriately within the system, thereby enabling accurate interpretations of dynamic behavior. Consequently, the Hamiltonian $\mathcal{H}$ is defined as:

$$\mathcal{H} = \frac{p^2v^2 - 1}{2}. \tag{2.57}$$

This definition ensures that the Hamiltonian remains constant along the ray trajectory. The Hamilton's ray equations in two-dimensional case are given by:

$$\dot{x} = \frac{\partial\mathcal{H}}{\partial p_x}, \tag{2.58}$$

$$\dot{z} = \frac{\partial\mathcal{H}}{\partial p_z}, \tag{2.59}$$

$$\dot{p_x} = -\frac{\partial\mathcal{H}}{\partial x}, \tag{2.60}$$

$$\dot{p}_z = -\frac{\partial \mathcal{H}}{\partial z} \tag{2.61}$$

Here $\dot{x}_i$ represents the components of vectors tangent to curves $x(t)$, which correspond to the rays of the system, and $\dot{p}_i$ describes the rate of change of the phase slowness, establishing a relationship between ray and phase velocities. These equations illustrate how the wave's spatial coordinates q_i and momenta p_i change over time. They are guided by the symplectic gradient vector field $X_{\mathcal{H}}$, defined as $X_{\mathcal{H}} = \left(\frac{\partial \mathcal{H}}{\partial p_x}, \frac{\partial \mathcal{H}}{\partial p_z}, -\frac{\partial \mathcal{H}}{\partial x}, -\frac{\partial \mathcal{H}}{\partial z}\right)$ [4], which represents the Hamiltonian vector field. This vector field ensures that the system's energy, represented by the Hamiltonian, remains constant along its trajectory.

2.4.1. Solutions of Ray-Tracing Equations for a Linearly Heterogeneous and Isotropic Velocity Model

In an isotropic velocity model, seismic waves propagate with a constant velocity in all directions, as the velocity is independent of the angle. And, the velocity of the waves varies linearly with depth in a linearly heterogeneous medium. The visualization of this medium is depicted in Figure 2.1.

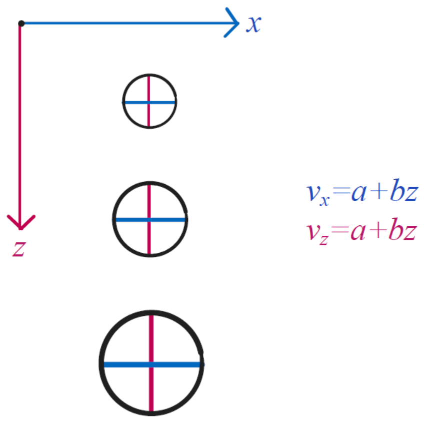

Figure 2.1. Illustration of a linearly heterogeneous and isotropic velocity model. Here, v_x denotes the horizontal velocity, while v_z denotes the vertical velocity within this medium. The parameters a and b characterize the velocity model.

The aim of this subsection is to derive Hamilton's ray equations in this linear heterogeneous and isotropic medium. The equations presented here were derived from the book cited in the reference [11]. Mathematically, the velocity in a linearly heterogeneous medium is expressed as:

$$v(\vartheta, z) = a + bz. \tag{2.62}$$

Here, v represents the velocity in this medium. The variables a and b are positive constants with specific units; a has units of velocity, while b has units of the reciprocal of time. Substituting this velocity expression into equation 2.57, we derive the Hamiltonian for an isotropic velocity model in a linearly heterogeneous medium:

$$\mathcal{H} = \frac{(a+bz)^2(p_x^2+p_z^2)-1}{2}. \tag{2.63}$$

Therefore, the Hamilton's equations of motion for this medium are expressed as follows:

$$\begin{aligned}
\dot{x} &= \frac{\partial \mathcal{H}}{\partial p_x} = (a+bz)^2 p_x, & (2.64)\\
\dot{z} &= \frac{\partial \mathcal{H}}{\partial p_z} = (a+bz)^2 p_z, & (2.65)\\
\dot{p_x} &= -\frac{\partial \mathcal{H}}{\partial x} = 0, & (2.66)\\
\dot{p_z} &= -\frac{\partial \mathcal{H}}{\partial z} = -b(a+bz)(p_x^2+p_z^2), & (2.67)
\end{aligned}$$

where $\dot{x}$ and $\dot{z}$ represent the first derivatives of the coordinates x and z with respect to time t, respectively. Here, $\dot{p_x}$ describes the first derivative of the horizontal component of momentum with respect to time. Since p_x is constant, its derivative is zero. The expression $\dot{p_z}$ describes the first derivative of the vertical component of momentum with respect to time.

To solve Hamilton's equations of motion, it is necessary to specify the initial conditions. By placing the source at the origin, the initial conditions are defined accordingly. These initial conditions are set as follows:

$$\begin{aligned}
x(0) &= 0, & (2.68)\\
z(0) &= 0, & (2.69)
\end{aligned}$$

$$\begin{aligned} p_x(0) &= \frac{\sin\vartheta_0}{v_0} = \frac{\sin\vartheta_0}{a}, && (2.70) \\ p_z(0) &= \frac{\cos\vartheta_0}{v_0} = \frac{\cos\vartheta_0}{a}. && (2.71) \end{aligned}$$

Here, ϑ_0 represents the take-off angle with respect to the z-axis. The trajectory of the ray is determined by the integral curves. These integral curves are given by [12]:

$$\begin{aligned} x(t) &= \frac{a}{b\sin\vartheta}\left(\tanh(bt - \operatorname{arctanh}(\cos\vartheta)) + \cos\vartheta\right), && (2.72) \\ z(t) &= \frac{a}{b}\left(\frac{1}{\sin\vartheta}\cosh(\operatorname{arctanh}(\cos\vartheta) - bt)\right), && (2.73) \\ p_x(t) &= \frac{\sin\vartheta}{a}, && (2.74) \\ p_z(t) &= \frac{\sin\vartheta}{a}\sinh(\operatorname{arctanh}(\cos\vartheta) - bt). && (2.75) \end{aligned}$$

In Section 4.2, we will solve the Hamilton's equations of motion for this medium using PINNs. We will validate the accuracy of our solution by comparing it to the analytical solution provided in above equation.

2.4.2. Solutions of Ray-Tracing Equations for a Linearly Heterogeneous and Constant Elliptically Anisotropic Velocity Model

In this model, the velocity model exhibits elliptical anisotropy in a linearly heterogeneous medium. This means that the velocity dependence with direction in an elliptical manner, resulting in velocity variations with both depth and angle. The visualization of this medium is shown in Figure 2.2.

The equations presented here were derived from the book [11] and the article [12] cited in the reference page.

Starting with remembering the eikonal equation, which is $p^2 = \frac{1}{v^2}$. The wavefront

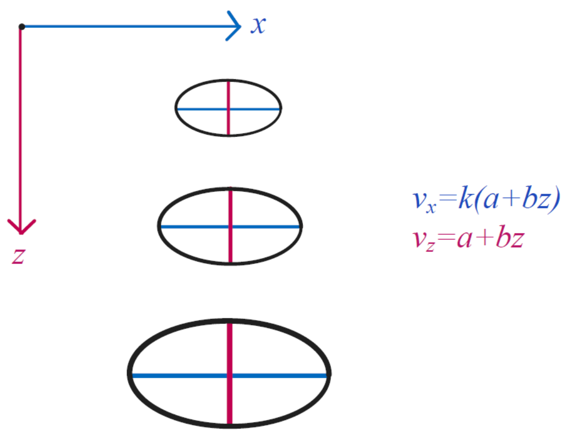

Figure 2.2. Illustration of a linearly heterogeneous and constant elliptically anisotropic velocity model. Here, v_x denotes the horizontal velocity, while v_z denotes the vertical velocity within this medium. The parameters a and b characterize the velocity model, and k is a constant that provides anisotropy to the velocity model.

velocity in an elliptically anisotropic velocity model is given by the following expression:

$$v(\vartheta, z) = \sqrt{v_x^2 \sin^2 \vartheta + v_z^2 \cos^2 \vartheta}. \tag{2.76}$$

Here, v_x represents the horizontal velocity, and v_z represents the vertical velocity. The vertical velocity v_z is expressed as:

$$v_z = a + bz, \tag{2.77}$$

The vertical velocity is characterized by the linearly hetergoneous medium. In this medium, it is assumed that the ratio of the horizontal and vertical velocities is remains constant with depth. The degree of the anisotropy is determined by the parameter χ, which is defined as follows:

$$\chi := \frac{v_x^2 - v_z^2}{2v_z^2}. \tag{2.78}$$

And thus, the horizontal velocity v_x can be expressed as:

$$v_x = (a + bz)\sqrt{1 + 2\chi}. \tag{2.79}$$

By combining the equations presented above, we derive the expression for the wavefront velocity, v, in this medium:

$$v = (a + bz)\sqrt{(1 + 2\chi) \sin^2 \vartheta + \cos^2 \vartheta}. \tag{2.80}$$

Therefore, by substituting the velocity into the eikonal equation, we obtain:

$$p^2 = \frac{1}{(a+bz)^2((1+2\chi)\sin^2\vartheta + \cos^2\vartheta)} \tag{2.81}$$

In accordance with the given velocity model, the equation becomes:

$$(a+bz)^2\left[(1+2\chi)p_x^2 + p_z^2\right] = 1. \tag{2.82}$$

The following is a definition of the Hamiltonian in the continuum:

$$\mathcal{H} = \frac{(a+bz)^2\left[(1+2\chi)p_x^2 + p_z^2\right] - 1}{2}. \tag{2.83}$$

The equations of motion derived from the Hamiltonian are as follows:

$$\begin{aligned}
\dot{x} &= (a+bz)^2(1+2\chi)p_x, && (2.84)\\
\dot{z} &= (a+bz)^2 p_z, && (2.85)\\
\dot{p_x} &= 0, && (2.86)\\
\dot{p_z} &= -\frac{b}{a+bz}. && (2.87)
\end{aligned}$$

To determine the integral curves, initial conditions are set at $t = 0$ and the source is positioned at the origin. Since the derivative of the momentum in the x-coordinate is zero, its integral curve is a constant. Thus, the initial conditions are defined as follows:

$$\begin{aligned}
x(0) &= 0, && (2.88)\\
z(0) &= 0, && (2.89)\\
p_x(0) &= \frac{\sin\vartheta_0}{a\sqrt{(1+2\chi)\sin^2\vartheta_0 + \cos^2\vartheta_0}}, && (2.90)\\
p_z(0) &= \frac{\cos\vartheta_0}{a\sqrt{(1+2\chi)\sin^2\vartheta_0 + \cos^2\vartheta_0}} = \sqrt{\frac{1}{a^2} - (1+2\chi){p_x}^2}. && (2.91)
\end{aligned}$$

For short notation, we named $p_z(0) = p_{z0}$. Hence, the general solution of the system derived from the Hamiltonian is:

$$x(t) = \frac{1}{p_x b}\left[\tanh(bt - \operatorname{arctanh}(ap_{z0})) + ap_{z0}\right], \tag{2.92}$$

$$z(t) = \frac{a}{b}\left[\frac{1}{p_x a\sqrt{1+2\chi}\cosh(\operatorname{arctanh}(ap_{z0}) - bt)} - 1\right], \tag{2.93}$$

$$p_x(t) = \frac{\sin\vartheta}{a\sqrt{(1+2\chi)\sin^2\vartheta + \cos^2\vartheta}}, \tag{2.94}$$

$$p_z(t) = p_x\sqrt{1+2\chi}\sinh(\operatorname{arctanh}(ap_{z0}) - bt). \tag{2.95}$$

These equations describe the integral curves, each of which gives a unique solution for x, z, p_x, and p_z over time, representing the trajectory of the ray.

In Section 4.2, we will use PINNs to solve Hamilton's equations of motion for this medium. To ensure the accuracy of our solution, we will compare the solution of the PINN with the analytical solution provided in the preceding equation.

2.4.3. Solutions of Ray-Tracing Equations for a Linearly Heterogeneous and Generalized Elliptically Anisotropic Velocity Model

In the previous case, the medium maintained constant ellipticity. Now, the ellipticity is not necessarily constant and thus it is called the generalized ellipticity. The visualization of this medium is given in Figure 2.3.

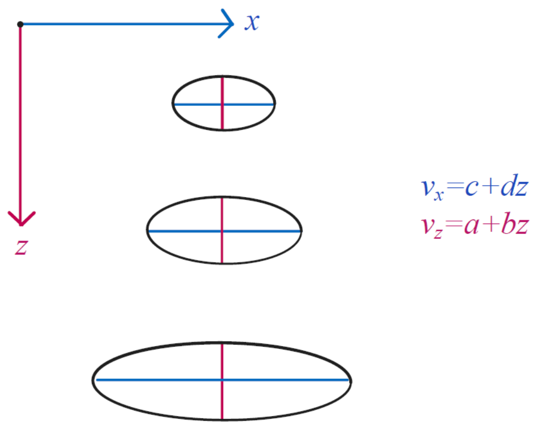

Figure 2.3. Illustration of a linearly heterogeneous and generalized elliptically anisotropic velocity model. Here, v_x denotes the horizontal velocity, while v_z denotes the vertical velocity within this medium. The parameters a, b, c and d characterize the velocity model.

In this medium, the vertical and horizontal velocities are defined as:

$$v_z = a + bz. \tag{2.96}$$

$$v_x = c + dz. \tag{2.97}$$

where a, b, c, d are positive constant, not required to be proportional to each other. Combining the above equations for v_z and v_x with the velocity expression for an elliptically anisotropic medium, as presented in Equation 2.76, we derive the velocity for a generalized elliptically anisotropic medium as follows:

$$v(\vartheta, z) = \sqrt{(c + dz)^2 \sin^2 \vartheta + (a + bz)^2 \cos^2 \vartheta}. \tag{2.98}$$

Substituting the velocity expression into the eikonal equation, we obtain:

$$p^2 = \frac{1}{(c + dz)^2 \sin^2 \vartheta + (a + bz)^2 \cos^2 \vartheta}. \tag{2.99}$$

Replacing the equation:

$$(c + dz)^2 p_x^2 + (a + bz)^2 p_z^2 - 1 = 0. \tag{2.100}$$

Hence, the Hamiltonian in this medium is defined as:

$$\mathcal{H} = \frac{(c + dz)^2 p_x^2 + (a + bz)^2 p_z^2 - 1}{2}. \tag{2.101}$$

Derived from the Hamiltonian, Hamilton's equations of motion for this medium are:

$$\begin{aligned} \dot{x}(t) &= (c + dz)^2 p_x, && (2.102) \\ \dot{z}(t) &= (a + bz)^2 p_z, && (2.103) \\ \dot{p}_x(t) &= 0, && (2.104) \\ \dot{p}_z(t) &= -d(c + dz)p_x^2 - b(a + bz)p_z^2. && (2.105) \end{aligned}$$

After the Hamilton's equations of motion have been identified, the initial conditions are given in a similar manner. The source is positioned at the origin, and the time is set to zero. Hence, the initial conditions are:

$$x(0) = 0, \tag{2.106}$$

$$\begin{aligned}
z(0) &= 0, && (2.107)\\
p_x(0) &= \frac{\sin\vartheta_0}{\sqrt{c^2\sin^2\vartheta_0 + a^2\cos^2\vartheta_0}}, && (2.108)\\
p_z(0) &= \frac{\cos\vartheta_0}{\sqrt{c^2\sin^2\vartheta_0 + a^2\cos^2\vartheta_0}}. && (2.109)
\end{aligned}$$

Finally, the integral curves that determine the trajectory of the ray are as follows:

$$\begin{aligned}
x(t) &= \frac{2}{bp_x}\left(\frac{\operatorname{arctanh}(Y)}{AB} + \frac{\arctan(AY+B)}{B} - \frac{1}{1+(AY+B)^2}\right) + c_2, && (2.110)\\
z(t) &= \frac{2(AY+B)}{dp_x(1+(AY+B)^2)} - \frac{c}{d}, && (2.111)\\
p_x(t) &= \frac{\sin\vartheta}{\sqrt{c^2\sin^2\vartheta + a^2\cos^2\vartheta}}, && (2.112)\\
p_z(t) &= \frac{dp_x B(1-(AY+B)^2)}{bA^2(1-Y^2)}. && (2.113)
\end{aligned}$$

The given equations are quite complex and lengthy, so we introduce some notations called A, B, Y to simplify our expressions. These notations are as follows:

$$\begin{aligned}
Y(t,\vartheta) &= \tanh\left(\frac{bA}{2B}t + c_1(\vartheta)\right), && (2.114)\\
A(\vartheta) &= \frac{\sqrt{b^2-(ad-bc)^2p_x^2}}{(ad-bc)p_x}, && (2.115)\\
B(\vartheta) &= \frac{-b}{(ad-bc)p_x} = \sqrt{A^2+1}. && (2.116)
\end{aligned}$$

The integration constants can be found by using the initial condition as:

$$\begin{aligned}
Y(0,\vartheta) &= \frac{ad-(ad-bc)\sqrt{1-c^2p_x^2}}{c\sqrt{b^2-(ad-bc)^2p_x^2}} = \tanh(c_1(\vartheta)), && (2.117)\\
c_1(\vartheta) &= \operatorname{arctanh}\left(\frac{ad-(ad-bc)\sqrt{1-c^2p_x^2}}{c\sqrt{b^2-(ad-bc)^2p_x^2}}\right), && (2.118)\\
c_2(\vartheta) &= \frac{-2}{bp_x}\left(\frac{\operatorname{arctanh}(Y(0,\vartheta))}{AB} + \frac{\arctan(AY(0,\vartheta)+B)}{B} - \frac{1}{1+(AY(0,\vartheta)+B)^2}\right). && (2.119)
\end{aligned}$$

In Section 4.2, we will employ PINNs to solve Hamilton's equations of motion for this medium. We will verify the accuracy of the PINN solution by comparing it to the analytical solution presented in the preceding equation.

3. NEURAL NETWORKS

The first paper that initiated the development of artificial neural networks was published in 1943 by Warren McCulloch and Walter Pitts [2]. In this paper, they describe a simple model of an artificial neural and develop a mathematical computational model using this model. Nowadays, artificial neural networks used in machine learning are computational models inspired by the interconnected neurons in the human brain. The purpose of using these networks in machine learning is to enable computers in performing tasks that humans can do easily, such as recognizing patterns, classification, prediction and decision making. However to accomplish these tasks with neural networks, numerous algorithms need to be designed, which can be computationally intensive, and a significant amount of data needs to be used for training [13]. There are two major problems with neural networks learning that have yet to be solved. One is that they can suffer from over-fitting during training and their decisions can be difficult to interpret, known as the black box problem. The other major problem is their safety and robustness in critical applications. Despite these problems, neural networks excel at learning complex patterns from large datasets, and their adaptive nature allows them to generalize well to unseen data and handle nonlinear relationships in a variety of applications. **The universal approximation theorem** of neural networks states that a neural network can approximate any function with arbitrary accuracy [3]. Therefore, neural networks are also capable of solving complex problems expressed by differential equations [4]. Using neural network solvers instead of traditional numerical methods to solve differential equations is more advantageous, making neural networks promising for the future and worthy of further study.

With the aim of explaining the methodology used in the thesis, first the fundamentals of neural networks are discussed without going into the details, followed by a more detailed discussion of the meaning of the terms used and the operation of the mechanism.

3.1. Fundamentals of Neural Networks

The purpose of this section is to examine the fundamental components of a neural network, including the functioning of neurons, layers, weights, and biases. Neurons form the foundational elements of neural networks, with a neural network being a complex system characterized by the interconnection of these neurons. The neuron holds a number between 0 and 1, this number in the neuron is the activation of that neuron [14]. The adjustable parameters in neurons are called weights and biases. The connection between neurons is called weight, which determines the strength of the connection. Biases are additional parameters used to tune the output of each neuron. In simple terms, a neuron calculates the weighted sum of input, adds a bias and then makes a decision [15]. Layers in neural networks make up this decision mechanism and the algorithms of these layers work as follows: Neurons receive inputs and put them into the "Input Layer", and the "Output Layer" produces an output, which provides the prediction. The layers that exist between the input and output layers are called "Hidden Layers" because the computations they perform are invisible to the user [13]. The neural network is designed to determine which features to use in the hidden layers, making it an efficient learning algorithm [16]. In the hidden layers, "Activation Functions" are applied to give the model a nonlinearity, enabling it to learn more complex relationships within the data [17]. The prediction accuracy is then evaluated using the "Loss Function". Minimizing loss is crucial for the neural network model to approach the target values. The loss function updates the weights and biases in the neurons to minimize them based on the calculated distance in the "Backpropagation Algorithm" step. This step is one of the most important steps for the model to learn the complex structure, and will explain in Section 3.1.2.

The terms related to the neural network mechanism have been briefly covered. The next section will mention three types of neural networks to enhance understanding and visualization: the perceptron, shallow, and deep neural networks.

3.1.1. Shallow Neural Networks vs Deep Neural Networks

To ease of understanding, the perceptron is used as a starting point. Then, a single hidden layer configuration is presented to illustrate the operation of shallow neural networks. Following this, the number of hidden layers is increased and the operation of deep neural networks is discussed. Finally, this section will introduce the concepts of model capacity and circuit theory and compare shallow and deep neural networks. The notations used in this thesis are drawn from multiple sources, as indicated by the references [13, 16].

The Perceptron: The perceptron is the simplest type of neural networks, consisting of a single input layer and an output layer. In our illustration, the input layer consists of multivariate inputs, and the output layer consists of a single neuron. This neuron processes inputs by multiplying each input variable with a corresponding weight from the weight matrix. The weighted inputs are then summed with the addition of a bias term, resulting in a weighted sum. By passing this weighted sum through an activation function, the model generates a prediction. The schematic representation in Figure 3.1 provides a visual overview of this process.

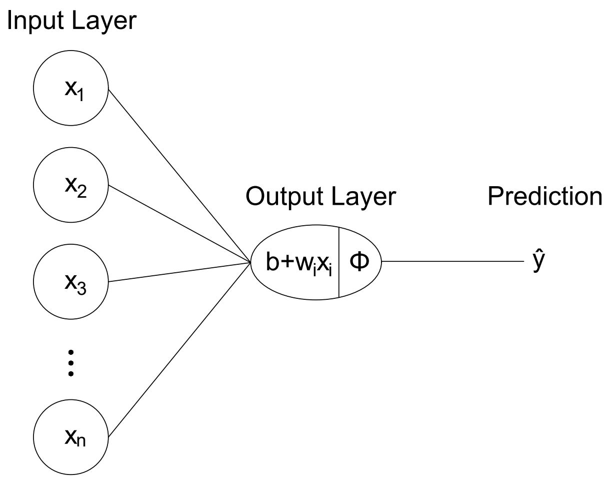

Figure 3.1. The illustration of a perceptron. This neural network can approximate a function mapping from $\mathbb{R}^n$ to $\mathbb{R}$.

In a mathematical representation using **Einstein's summation convention** (repeated indices imply summation), the prediction for a single value can be expressed as follows:

$$\begin{aligned} z &= b + w_i x_i, && (3.1) \\ \hat{y} &= \mathit{\Phi}(z). && (3.2) \end{aligned}$$

Here, $\mathit{\Phi}$ represents the activation function, the weighted sum z is preactivation value, $\hat{y}$ denotes the prediction of model, w_i is the i^{th} term of weight matrix, x_i is the i^{th} term of input vector, and b is the bias term.

Figure 3.1 illustrates a neural network with a single output neuron. If the output is a multivariate output, the mathematical notation for this case is presented below:

$$\begin{aligned} z_j &= b_j + w_{ji} x_i, && (3.3) \\ \hat{\mathbf{y}} &= \mathit{\Phi}(\mathbf{z}). && (3.4) \end{aligned}$$

In these equations, z_j signifies the j^{th} term of the vector $\mathbf{z}$, where $\mathbf{z}$ is composed of terms generated by summing the product of input values x_i with their corresponding weights w_{ji}, and adding the bias term b_j. The resulting vector $\mathbf{z}$ is then processed through the activation function $\mathit{\Phi}$, producing the output vector denoted as $\hat{\mathbf{y}}$.

Shallow Neural Networks: A shallow neural network is composed of one input layer, one hidden layer, and one output layer. In our illustration, the input layer contains the multivariate inputs, and the output layer contains the multivariate outputs. Each neuron in the hidden layer receives inputs from the input layer and calculates the weighted sum of the inputs, adds a bias, and then applies an activation function. The same process is then carried out by the output layer, which receives its inputs from the hidden layer, in order to make a prediction. Mathematically it is defined as:

$$\begin{aligned} z_j^{[1]} &= b_j^{[1]} + w_{ji}^{[1]} x_i, && (3.5)\\ \mathbf{h}^{[1]} &= \Phi^{[1]}(b_j + w_{ji} x_i), && (3.6)\\ z_j^{[2]} &= b_j^{[2]} + w_{jk}^{[2]} h_k^{[1]}, && (3.7)\\ \mathbf{h}^{[2]} &= \Phi^{[2]}(\mathbf{z}^{[2]}), && (3.8)\\ \hat{\mathbf{y}} &= \mathbf{h}^{[2]}. && (3.9) \end{aligned}$$

Here, x, w, b, Φ represent multivariate input, weight, bias, and activation function, respectively. $\mathbf{h}^{[1]}$ represents the output vector of the first hidden layer and $\hat{\mathbf{y}}$ denotes the output vector of the overall network, which is the second hidden layer $\mathbf{h}^{[2]}$ in this case. The visual representation of this shallow neural network can be found in Figure 3.2.

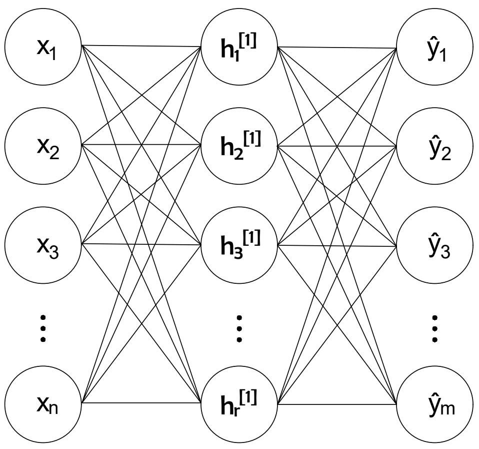

Figure 3.2. The illustration of a shallow neural network. This neural network can approximate a function mapping from $\mathbb{R}^n$ to $\mathbb{R}^r$, and subsequently from $\mathbb{R}^r$ to $\mathbb{R}^m$, representing a composite function from $\mathbb{R}^n$ to $\mathbb{R}^m$.

Deep Neural Networks: Shallow neural networks, which are a subset of deep neural networks, were illustrated above. The scope is now extended to formulation of a deep neural network with L layers. In the context of neural networks, the number of layers is defined as the sum of the number of hidden layers and the output layer, with the input layer excluded since only computational layers are counted [13]. A deep neural network with L layers includes $L-1$ hidden layers, and can be represented by a combination of $L-1$ shallow neural networks. For a visual representation, refer to Figure 3.3, which illustrates the architecture of deep neural networks.

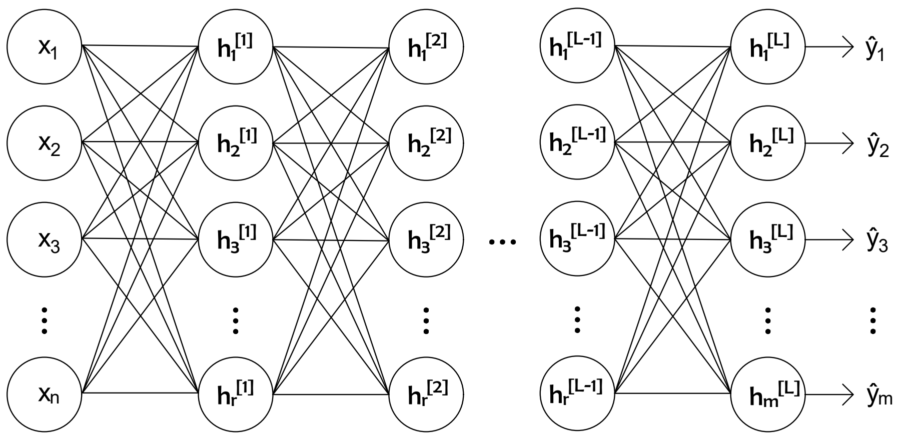

Figure 3.3. The illustration of a deep neural network with L layers. This network can approximate a function mapping from $\mathbb{R}^n$ to $\mathbb{R}^r$, followed by $L-2$ mappings within $\mathbb{R}^r$ to $\mathbb{R}^r$, and finally mapping from $\mathbb{R}^r$ to $\mathbb{R}^m$.

The neural network models utilized in this thesis have hidden layers with an equal number of hidden units. As a result, they are processed and represented similarly in the deep neural network visualization. In reality, deep neural networks do not necessarily have an equal number of units in the hidden layers.

$$h_j^{[l]} = \varPhi^{[l]}\left(b_j^{[l]} + \sum_{k=1}^{n} w_{jk}^{[l]} h_k^{[l-1]}\right), \tag{3.10}$$

$$\hat{\mathbf{y}} = \mathbf{h}^{[L]}, \tag{3.11}$$

where $[l]$ signifies the l^{th} layer of the L-layered deep neural network. The activation of each neuron in a given layer is determined by the weighted sum of its inputs, incorporating biases and the application of the activation function $\varPhi^{[l]}$. The prediction vector, $\hat{\mathbf{y}}$, corresponds to the output of the last layer, $\mathbf{h}^{[L]}$.

Comparison of Shallow and Deep Neural Networks: The performance of neural networks is significantly influenced by their capacity, which is determined by multiplying the number of hidden layers by the number of units in each layer [18]. The relationship between architecture and capacity is complex and can be better understood through **circuit theory** [16]. This theory clarifies the computational difficulties of shallow networks and the advantages of deeper ones [16]. In this context, deep networks are demonstrating superior performance to shallow networks due to their greater capacity and layered structure, which allows them to capture complex data relationships and generate many linear regions per parameter [18]. Their ability to approximate functions efficiently and manage large, structured inputs like images makes them the optimal choice for practical applications. Empirical evidence consistently supports the superior performance of deep neural networks, highlighting the intricate relationship between architecture, capacity, and computational efficiency in machine learning.

3.1.2. Backpropagation Algorithm

The backpropagation algorithm is a fundamental technique for optimizing the performance of neural networks and enabling efficient learning. This algorithm consists of two main stages, the forward pass and the backward pass. During the forward pass, a neural network processes input data using randomly initialized weights and biases to generate predictions. The concepts of neural networks mentioned above describe the forward propagation. Rewriting Equation 3.10 which represents the forward propagation step of the deep neural network:

$$h_j^{[l]} = \Phi^{[l]}\left(b_j^{[l]} + \sum_{k=1}^{n} w_{jk}^{[l]} h_k^{[l-1]}\right), \tag{3.12}$$

$$\hat{\mathbf{y}} = \mathbf{h}^{[L]}. \tag{3.13}$$

During forward propagation, the loss is calculated using the existing weights and biases. The least squares loss function is denoted as:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2. \tag{3.14}$$

Here, L represents the loss function, $\mathbf{y}$ refers to the true output data, and $\hat{\mathbf{y}}$ refers to the predicted output data. The loss function represents the individual term associated with a data point, while the cost function encapsulates the overall quantity that needs to be minimized [18]. The cost function is calculated by averaging the loss function, which measures the mismatch between the predicted and target values. The cost function is denoted as:

$$J(w, b) = \frac{1}{N} \sum_{i=1}^{m} L(y_i, \hat{y}_i). \tag{3.15}$$

Here, J represents the cost function, L represents the loss function, w denotes the weight, b denotes the bias, $y^{(i)}$ refers to the i^{th} term of the true output data, and $\hat{y}^{(i)}$ refers to the i^{th} term of the predicted output data, and N represents the number of data. As mentioned above, the cost function computes the average loss of the training dataset. After calculating the loss and cost in forward propagation, it is time to proceed with backward propagation. Subsequently, the network calculates the local derivatives of the cost function with respect to the output values by using the chain rule of calculus. Backward propagation entails the use of a gradient descent algorithm to iteratively adjust the weights and biases based on these gradients. To minimize the cost function and improve the performance of the model, the gradient descent algorithm sets the parameters as follows: These gradients are multiplied by the learning rate, which is roughly the step size, and subtracted from the weights and biases to bring the weights and biases closer to their optimal values. The process of updating parameters is expressed mathematically as:

$$w \leftarrow w - \alpha \nabla_w J, \tag{3.16}$$

$$b \leftarrow b - \alpha \nabla_b J, \tag{3.17}$$

where w denotes the weight, b denotes the bias, α represents the learning rate, and ∇ indicates the gradient operation.

The fundamentals of a neural network have been covered so far. The next step is to introduce the key components that are employed during the training process of neural networks, which include the loss function, optimizer, hyperparameters and their tuning, the challenges that arise during the training process and regularization techniques.

3.2. Training of a Neural Network

The process of training a model starts with selecting initial parameter values and a specific loss function. Then, the gradients of the loss with respect to the parameters are calculated, and the model parameters are updated based on these gradients to minimize the overall loss. In machine learning, the objective function and loss function are closely interconnected. The objective function sets the overall goal of the problem, while the loss function measures the model's performance in achieving that goal during training by measuring the mismatch between predicted and actual values. Training a neural network can be viewed as the process of solving an optimization problem, where the objective is to minimize a defined loss function by adjusting the weights and biases of the network.

Before starting to train a neural network, I would like to mention three datasets that are commonly used at different stages of model building: training set, validation set, and test sets. To train a neural network model, it is initially fitted with a training dataset. The **training dataset** is used to set the parameters of the model, and the model learns from the data in the training set to make predictions. Secondly, there is the **validation set**, which is a distinct subset of the dataset. This set is used to evaluate the model's performance during training, helping to tune hyperparameters and providing an estimate of how well the model will perform on unseen data. Finally, the **test dataset** is used to assess the final model's fit on the training data. It helps evaluate the model's ability to generalize to new, unseen data, and is not utilized during training or validation [19].

This section discusses the training of a neural network, key components such as activation functions, loss functions, optimizers, hyperparameter selection and tuning, and challenges faced during training. Additionally, various regularization techniques to overcome these challenges will be explored.

3.2.1. Activation Functions

Previously, the activation function was denoted by Φ without an explanation of its role. The purpose of this section is to discuss its importance and how it works. Subsequently, the preferred activation function will be explained, and reasons for its preference will be given.

Activation functions play a crucial role in the performance of neural networks by determining the output of each neuron. These functions add nonlinearity to the model, so that the model can learn more complex relationships within the data [17].

Choosing the appropriate activation function is important in training the model effectively. The most commonly used functions are Sigmoid, ReLU and Hyperbolic Tangent (tanh) [15]. Each activation function performs optimally in specific scenarios. The sigmoid activation function is used for binary classification and outputs the interval $[0, 1]$, so it do not blow-up the activations [15]. However, it has the problem of a vanishing gradient near the horizontal part of its curve. ReLu is computationally easier than tanh and sigmoid. The ReLU function gives a value in the range $[0, \infty)$, in other words, the activation is blow-up. It is popular because of its simplicity and efficiency in dealing with the vanishing gradient problem [15]. However, it cannot be used for higher order derivatives. Using the hyperbolic tangent function, tanh, instead of the sigmoid function is generally more efficient. This is due to the fact that it generates activations with a mean closer to zero when the values are situated within the interval $[-1, 1]$. When the mean of the data is closer to zero than 0.5, it facilitates the learning process for the next layer. However, tanh also exhibits the vanishing gradient problem, similar to the sigmoid. Nevertheless, the use of tanh alleviates this problem, although it does not solve it entirely. Consequently, it is more effective than other alternatives for higher order partial derivatives. In general, the aim of this thesis is to solve partial differential equations with physics-informed neural networks. In order to achieve this aim, the activation function tanh was selected, as it has been observed to produce superior results when attempting to solve problems involving higher-order derivatives.

Figure 3.4 depicts the hyperbolic tangent function, and Figure 3.5 depicts the first derivative of this function. The mathematical expression of the hyperbolic tangent function is as follows:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{3.18}$$

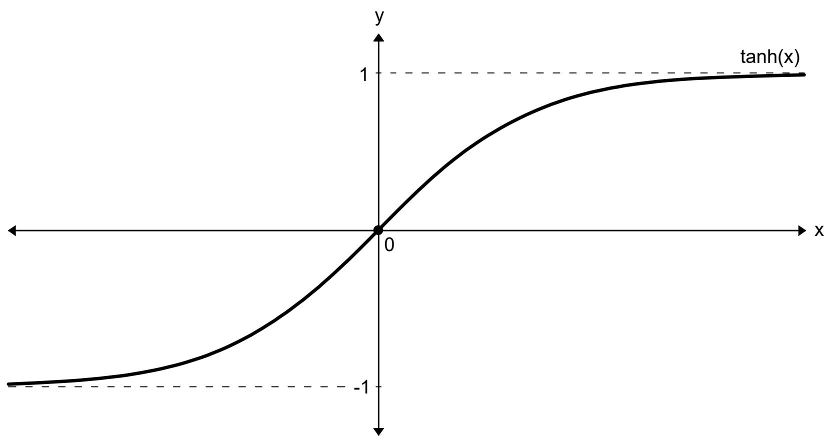

Figure 3.4. The hyperbolic tangent function.

Given that the derivative of the activation function is crucial for solving partial differential equations and the training of neural networks using backpropagation, it is necessary to calculate the derivative of the tanh function. The first derivative of the tanh function is:

$$\frac{d}{dz}\tanh(z) = \frac{4e^{-2z}}{(1+e^{-2z})^2} = 1 - (\tanh(z))^2. \tag{3.19}$$

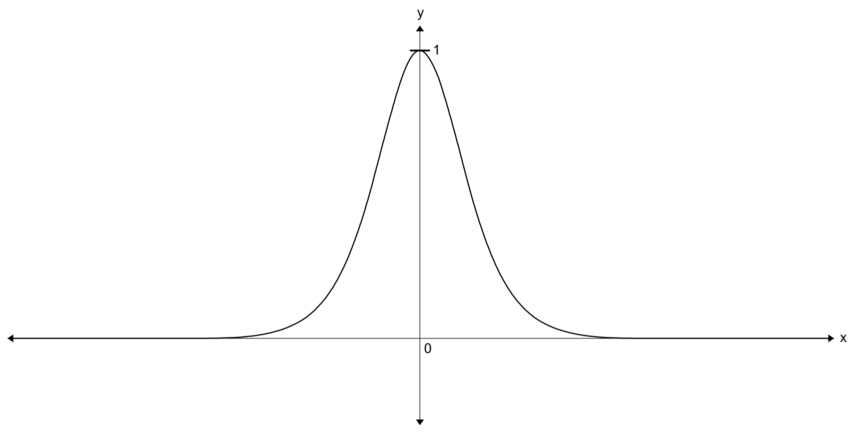

Figure 3.5. The first derivative of the hyperbolic tangent function.

3.2.2. Loss Functions

During the training, we look for parameter values that minimize the loss and therefore match the training outputs with the target values as closely as possible [18]. The choice of the loss function depends on the nature of the problem being solved and specific requirements. Various loss functions are used in machine learning to handle different types of problems. The most common ones are binary cross-entropy, categorical cross-entropy, and mean squared error loss function. This text examines the three functions, each serving a different purpose. The binary cross-entropy loss is an effective method for addressing binary classification tasks, making it a prevalent choice in the field of binary classification. It is frequently employed in conjunction with the sigmoid activation function. On the other hand, the categorical cross-entropy excels in multi-class classification problems. It is often paired with the softmax activation function in the output layer. Finally, the MSE loss is the preferred function for regression problems. Since the MSE loss function is utilized in all of our neural network models, this loss function will be examined in detail.

Mean Squared Error Loss Function: The mean squared error loss function is a commonly used for regression tasks due to its simplicity, mathematical properties, and effectiveness. It is often preferred for predicting continuous numeric values. Its non-negativity and differentiability make it compatible with gradient-based optimization algorithms, such as gradient descent, which facilitates efficient model training. Moreover, the MSE loss function is convex, which guarantees convergence to the global minimum. The mathematical expression of the mean squared error loss function is as follows:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{3.20}$$

Here, N is the total number of samples in the dataset, y_i is the true target value for the i^{th} sample, $\hat{y}_i$ is the predicted value for the i^{th} sample.

In our models, we combined the MSE loss and the hyperbolic tangent function. This combination enables the network to learn complex relationships using tanh activation functions in the hidden layers while optimizing the model parameters, weights, and biases to minimize the MSE loss between the predicted and actual values in the output layer.

3.2.3. Optimizers

Optimizing a neural network model for enhanced performance is to influence the convergence speed. This is achieved through multiple iterations of the training loop, which converges towards the global minimum of the loss function [18]. Therefore, choosing an optimizer is crucial in the training process. The selection of an optimizer is typically based on the characteristics of the data and the complexity of the neural network structure. The aim is to analyze the advantages and disadvantages of common optimizers, including SGD (Stochastic Gradient Descent) [20], AdaGrad (Adaptive Gradient Algorithm) [21], RMSProp (Root Mean Square Propagation) [22], and Adam (Adaptive Moment Estimation) [23].

The SGD algorithm is widely used for optimizing neural networks during training due to its simplicity. Unlike the gradient descent algorithm, which is highly sensitive to the starting point, SGD adds noise to the gradients at each step and updates the model parameters in the opposite direction of the loss function gradient [18]. Instead of computing the gradients exactly, each iteration estimates the gradient based on a single randomly selected sample. Since the samples are drawn randomly from the ground truth distribution, the expected risk is directly optimized [20]. However, stochastic gradient descent may converge slowly, particularly in the presence of saddle points or local minima. Although computationally efficient, the use of a fixed learning rate may potentially lead to zigzag convergence [13]. The utilization of momentum reduces and smooths oscillations by adding part of the previous update vector to the current update [13]. This helps stochastic gradient descent to maintain its direction, overcome local minima and plateaus, and reach the optimal solution with fewer updates. Neural Networks and Deep Learning: However, it is important to note that SGD with momentum can overshoot the minimum in areas with steep slopes due to momentum.

The AdaGrad algorithm is an adaptive optimisation algorithm that adjusts the learning rate of each parameter based on the history of gradients. It scales the learning rates according to the total squared magnitude of the partial derivative with respect to each parameter [13]. It uses a subgradient method that dynamically incorporates geometry information from data observed in previous iterations to perform more informative gradient based learning [21]. AdaGrad works well on problems with sparse data or sparse gradients, but the learning rate decreases over time, leading to slow convergence in deep learning scenarios.

The RMSProp algorithm is an adaptive optimization algorithm that has been demonstrated to perform well in non-stationary environments. The algorithm shares the same motivation as AdaGrad, but it addresses the problem of diminishing learning rates by using exponential averaging instead of simply adding quadratic gradients [13]. This makes it computationally efficient and requires less memory than some other adaptive methods. However, it may have difficulty converging in some cases.

The Adam algorithm is also an adaptive optimization algorithm renowned for its effectiveness in optimising stochastic objective functions. It calculates separate adaptive learning rates for each parameter, and adapts the optimization process to specific data characteristics. Adam combines the strengths of AdaGrad and RMSProp by taking advantage of exponentially decreasing moving averages, preserving both the mean and the uncentered variance of gradients. Adam improves convergence speed, and additionally, it exponentially smooths the first-order gradient, seamlessly incorporating momentum into the update process. Thus further increasing its efficiency and effectiveness in optimization tasks [23]. The Adam update rule is as follows:

$$t \leftarrow t+1 \tag{3.21}$$

$$g_t = \nabla_{\theta_t} J(\theta_{t-1}) \tag{3.22}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \tag{3.23}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \tag{3.24}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{3.25}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3.26}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon} \cdot \hat{m}_t. \tag{3.27}$$

Here, t denotes the current time step, and at each iteration t increases by 1. g_t denotes the gradient of the objective function with respect to the parameters to find the direction of steepest ascent. m_t is the mean of past gradients, and v_t is variance of past gradients. Using $\hat{m}_t$ and $\hat{v}_t$ terms that account for initial bias, the bias corrected. η represents the learning rate, and ϵ is a small constant added to the denominator to prevent division by zero and stabilize the computation. β_1 and β_2 determine the decay rates with default values of 0.9 and 0.999, respectively.

Using the Adam as an optimizer in a model offers several advantages. It is easy to implement, computationally efficient, and requires minimal memory. Additionally, it is invariant to the rescaling of gradients and is suitable for handling large-scale problems in terms of both data and parameters. Moreover, Adam is suitable for non-stationary objectives and problems with sparse gradients. Furthermore, the paper on the Adam algorithm [23] provided a convergence proof for convex loss functions, reinforcing its theoretical underpinnings and reliability.

3.2.4. Hyperparameters and Their Tuning

The choice of hyperparameters can have a significantly impact the performance of a machine learning model. Hyperparameter tuning involves evaluating the performance of the model by attempting different combinations of hyperparameter values and selecting the hyperparameter set that gives the optimal performance on a validation dataset. The aim is to enhance the accuracy and predictive efficacy of the model.

As previously stated, the **learning rate** is a hyperparameter that determines the step size in each iteration when updating the model parameters weights and biases to move towards the minimum cost, affecting the accuracy of the training process. It is crucial to choose the correct learning rate, which can only be determined empirically for each model. A low learning rate is typically associated with slow convergence, although it can result in a more precise model when the number of iterations is increased. On the other hand, a high learning rate leads to faster convergence but may result in a less accurate model. The learning rate is between 10^{-2} and 10^{-4} typically but actual values can vary based on the specific problem and dataset. It is denoted by α in this thesis.

Instead of using a fixed learning rate, adjusting the learning rate through a **learning rate scheduler** during training can improve the efficiency of the model's learning process. This approach involves starting with a higher learning rate to quickly approach the minimum loss, and then gradually decreasing the learning rate to prevent the model from diverging and ensure a more precise convergence towards the optimal loss.

In each iteration of the training process, the algorithm randomly selects a subset of training data, known as a **batch**, and computes the gradient based on this subset alone. The **batch size** represents the number of training examples used in one iteration [18].

Smaller batch sizes offer computational efficiency, whereas larger batch sizes contribute to more stable convergence.

An **epoch** refers to a single pass through the entire training dataset [18]. The number of epochs indicates how many times the learning algorithm processes the complete training dataset. Insufficient epochs may result in underfitting, as the model may not capture the underlying patterns, while excessive epochs can lead to overfitting, where the model becomes too specific to the training data and performs poorly on unseen data.

Experimental tuning of hyperparameters is necessary to balance between underfitting and overfitting. Iterative adjustments may be required to find the optimal combination of hyperparameter values. Hyperparameter combinations can be tested using methods such as grid search, random search, and Bayesian optimization, which have been shown to produce superior results. This can improve the generalization ability and performance of models.

3.2.5. Challenges in Training and Regularization Techniques

During neural network training, many challenges arise that require effective regularization strategies to improve model performance. This section discusses the challenges faced during training and explains the various regularization techniques used to overcome them.

The most important of these challenges is **overfitting**. Overfitting means that fitting a model to a given training dataset does not guarantee good predictive performance on unseen test data, even if the model predicts the targets perfectly on the training data. In other words, the model does not generalize well to unseen test data [13]. In contrast, **underfitting** is when the model does not fit the training data well enough and the model does not understand the dataset, which means it is complex. In this case, it can be said that the model cannot learn. **Insufficient or unbalanced**

training data represents another factor that negatively affects the accuracy of the model. In this case, the overfitting or underfitting may occur.

Another issue is related to the **capacity and structure of the model**. Deep learning models may require large datasets and complex architectures. This means high computing power and time. Furthermore, the presence of an excessive number of hyperparameters in the model can make it difficult to find the optimal combination.

After discussing the challenges faced during training, we will mention the three most commonly used and important regularization techniques employed in our models to smooth the function: L2, dropout, and early stopping techniques.

L2 regularization is a commonly used technique to reduce the complexity of neural networks. L2 regularization is typically employed in neural networks on the weights rather than the biases, which is commonly referred to as a **weight decay** term [18]. This technique brings the weights closer to zero, and in the limit, if all weights are forced to be zero, the network will produce a constant output determined by the final bias parameter [18].

To prevent the model from overfitting, a **dropout** technique can be applied. This technique randomly sets some of the hidden units to zero during training. This reduces the network's dependence on any particular hidden unit, resulting in smaller weight magnitudes [18]. As a result, the effect of the presence or absence of a hidden unit on the function is minimized.

Another technique employed to prevent overfitting is **early stopping**. Early stopping means stopping the training procedure before it has fully converged [18]. It usually stops training if the performance on the validation set does not improve over a certain period of time. Since the weights are initialized to small values, they do not have enough time to grow; as a result, early stopping achieves a result comparable to L2 regularization [18].

4. PHYSICS-INFORMED NEURAL NETWORKS

The physics-informed neural network, abbreviated as PINN, is a scientific machine learning technique that employs the strengths of physics-based models. PINNs were first introduced in 2017 by Raissi, Perdikaris, and Karniadakis [5]. Since the PINN algorithm combines the strengths of physics-based models and machine learning techniques, it results in a more accurate and efficient solution of partial differential equations compared to traditional numerical methods. This algorithm is also more efficient than conventional neural networks. Deep neural networks typically involve a significant number of independent and dependent parameters, necessitating a large amount of training data. In contrast, PINNs can operate effectively with small datasets and using only initial or boundary conditions for solving partial differential equations [8]. These neural networks ensure that their predictions are consistent with physical laws observed over time and represented by nonlinear partial differential equations. They are designed to respect the symmetries, invariances and conservation principles that arise from these laws [7]. The concept behind PINNs is to train a neural network to approximate the solution of a partial differential equation while incorporating the physics of the problem directly into the learning process. It efficiently finds approximate solutions to PDEs by transforming the loss function into an optimization problem [6].

PINNs are versatile tools that can be used for both forward and inverse problems [7]. In forward problems, PINNs can predict the behavior of physical systems described by PDEs without requiring for an explicit analytical solution, even in scenarios where only boundary or initial condition data are available. In inverse problems, PINNs can learn the relationships between input-output data pairs, even if the system's behavior is complex and nonlinear, and determine the best model parameters. Due to the PINN algorithm's efficacy in solving PDEs and its ability to preserve physical laws, it was chosen for the study in the thesis to solve the harmonic oscillator and ray-tracing equations using neural networks.

Hamiltonian Neural Network, abbreviated as HNN, is a type of neural network that inspire from Hamiltonian mechanics. This algorithm is designed to conserve energy, momentum, and angular momentum in a specific manner [9]. The HNN model demonstrates quicker training and superior generalization compared to a standard neural network. Additionally, it exhibits perfect reversibility in time [9]. Since it is important to conserve energy when solving our problem, we used the definition of the energy loss function of HNNs.

In order to construct neural network models, the Python programming language was employed, with "PyTorch" [24] selected as the library. Due to PyTorch's dynamic computational graph, that allows easy and intuitive model building and debugging, it is a popular open source machine learning library. PyTorch provides "tensor operations", which are crucial for the implementation of neural network algorithms. Tensors are multidimensional arrays that represent numerical data in PyTorch, making numerical calculations efficient. Among other benefits, one of the most important features of PyTorch is "Autograd", a powerful automatic differentiation system [10]. Autograd enables the automatic calculation of the gradients required to train neural networks using gradient-based optimization algorithms such as SGD (stochastic gradient descent) and Adam (adaptive moment estimation) [23]. In this way, it simplifies the backpropagation process and makes it easier to implement and experiment with complex neural network architectures [17].

4.1. Hamiltonian Harmonic Oscillator

Hamiltonian harmonic oscillator is an important model in physics, and the detailed mathematical derivation of its equations is given in Section 2.1. Equation 2.3 for position is given, and the momentum associated with the harmonic oscillator system can be expressed as follows:

$$x(t) = A\cos(\omega t + \phi), \tag{4.1}$$

$$p(t) = -A\omega\sin(\omega t + \phi). \tag{4.2}$$

To construct the analytical solution function that generated the true data, we used above equations and the following conditions.

$$\begin{aligned} \text{initial position :}\quad & x_0 = 1, \\ \text{initial momentum :}\quad & p_0 = 0, \\ \text{mass :}\quad & m = 1, \\ \text{angular momentum :}\quad & \omega = 10, \\ \text{phase angle :}\quad & \phi = 0, \\ \text{amplitude :}\quad & A = 1, \\ \text{spring constant :}\quad & k = 100, \\ \text{maximum time :}\quad & t_{max} = 6. \end{aligned}$$

The increase in the value of k, which is the spring constant in the harmonic oscillator, represents a significant challenge for the network model in solving the problem. To ensure the performance of the model, the value of k is therefore chosen to be 100.

The generation of true data serves two purposes: First, to verify that the model has learned correctly. Second, the data is used as training data. To generate the training data, the true data is sliced and 30 data points are selected.

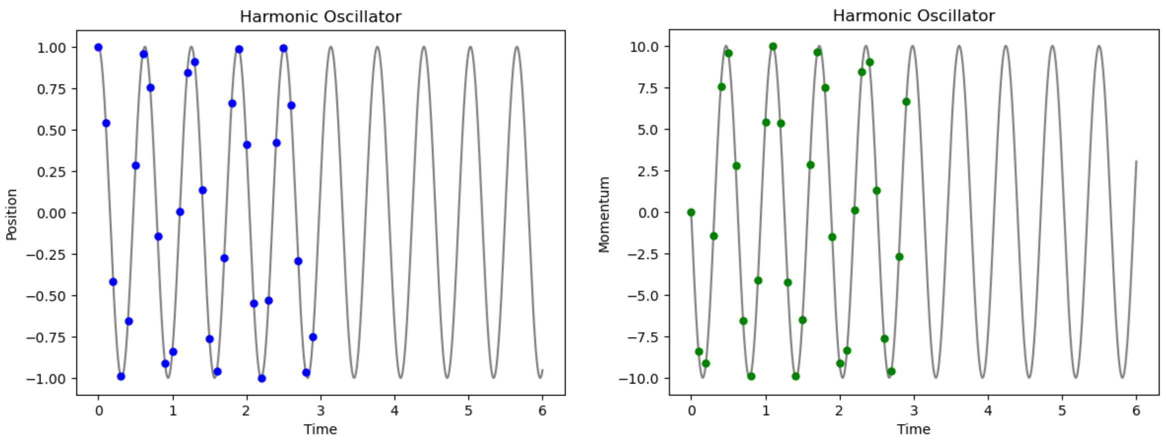

Figure 4.1. The illustration of the training data for the harmonic oscillator. In both figures, the grey curve represents the true solution. The left-hand side figure with blue dots illustrates true position data, while the right-hand side figure with green dots illustrates true momentum data generated by analytic solution.

4.1.1. Utilizing a Neural Network for Solving the Harmonic Oscillator

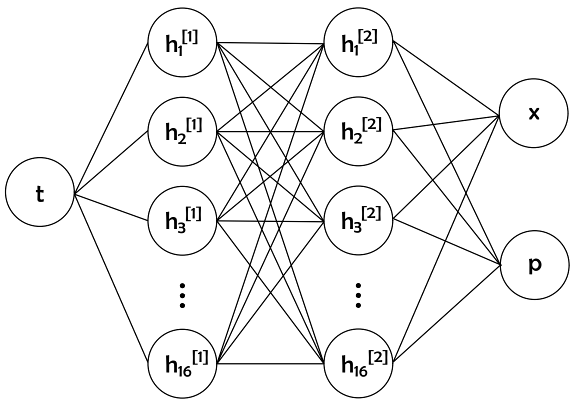

Figure 4.2. Illustration of the neural network.

To solve harmonic oscillator, we first create a conventional neural network model. The model is 3-layer neural network consists of an input layer, 2 hidden layers, and an output layer. The input layer receives time data, and the output layer gives two outputs representing position and momentum. In both hidden layers, there are 16 hidden units, and the hyperbolic tangent activation function is employed as the activation function. For optimizer in the network we employ the Adam optimizer with a learning rate of 0.01. To prevent overfitting and improve generalization, we use weight decay set at 0.1. To enhance adaptability, we implement a learning rate scheduler with a step size of

1000 and a decay factor gamma of 0.8. This scheduler gradually reduces the learning rate every 1000 steps. The new learning rate is calculated by multiplying the current learning rate by the decay factor. The model was iterated over a total of $3,000$ epochs.

During the training, we compute the data losses for both position and momentum data. The criterion is calculated using the Mean Squared Error (MSE) loss, which evaluates the difference between the model's predictions for both position and momentum, and the corresponding training data. Writing them in equation form:

$$\mathcal{L}_{data_x} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} (x^i_{pred} - x^i_{training})^2, \tag{4.3}$$

$$\mathcal{L}_{data_p} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} (p^i_{pred} - p^i_{training})^2. \tag{4.4}$$

Here, x^i_{pred} denotes the i^{th} term of the prediction of model for position, and $x^i_{training}$ denotes the i^{th} term of the training data for position. Similarly, p^i_{pred} denotes the i^{th} term of the prediction of model for momentum, and $p^i_{training}$ denotes the i^{th} term of the training data for momentum. $N_{data} = 30$ denotes the number of data points.

Although initial conditions are included in the data loss, we define another loss called "initial conditions loss". This is because we wished to ensure that this model satisfies the initial conditions well. Similarly, the MSE loss function was employed to calculate the difference between the model's predictions for the initial data and the initial training data for both position and momentum. This is mathematically represented as:

$$\mathcal{L}_{IC_x} = \left(x_{0_{pred}} - 1\right)^2, \tag{4.5}$$

$$\mathcal{L}_{IC_p} = \left(p_{0_{pred}} - 0\right)^2. \tag{4.6}$$

The model was run and the weights of the losses were estimated in accordance with the loss results. Since we want the model to satisfy the initial conditions well, we gave more weight to them. Subsequently, adjustments were made in accordance with the loss weights obtained through a trial and error method. The weights obtained through this method are as follows:

$$\begin{aligned} \mathcal{L}_{data} &= 10^{10} \cdot \mathcal{L}_{data_x} + 10^7 \cdot \mathcal{L}_{data_p}, & (4.7) \\ \mathcal{L}_{IC} &= 10^3 \cdot \mathcal{L}_{IC_x} + 10^2 \cdot \mathcal{L}_{IC_p}. & (4.8) \end{aligned}$$

Here, the IC is the abbreviation of initial condition. The total loss comprises these two losses and is expressed as:

$$\mathcal{L}_{total} = \mathcal{L}_{data} + \mathcal{L}_{IC}. \tag{4.9}$$

Backpropagation is applied to the total loss to minimize by iteratively adjusting the model's parameters in the direction that reduces the loss. Figure 4.3 illustrates this process of minimizing losses.

Figure 4.4 and Figure 4.5 show the model's predictions for position and momentum. Based on these figures of the network's predictions, it is evident that this neural network model can learn effectively when provided with well-balanced training data but unable to learn when the training data is not available. It can be concluded that the only way to solve the harmonic oscillator equation with a neural network is to provide balanced data from all sources.

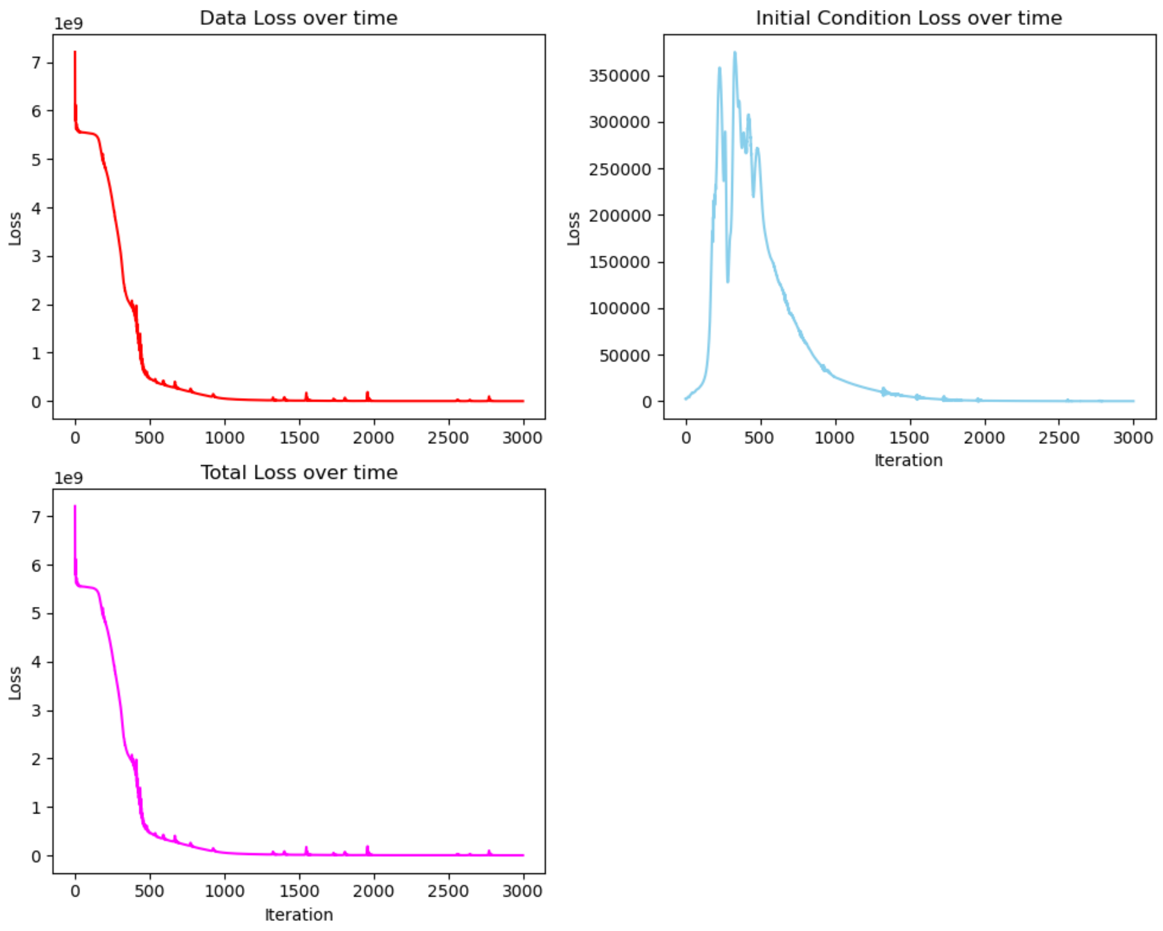

Figure 4.3. The loss results of the harmonic oscillator forward model solved with a conventional neural network are presented. The data loss over iterations is represented by the red curve, the initial condition loss by the sky blue curve, and the total loss by the magenta curve.

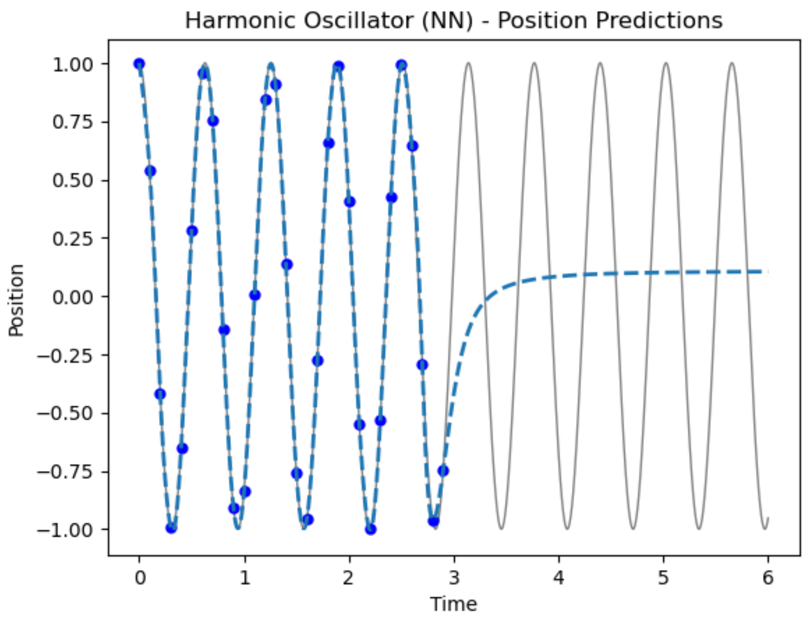

Figure 4.4. Position prediction of harmonic oscillator neural network. Blue dashed line represent the position prediction of the model. Blue dots are training data that we train the model. The gray line represents the true position.

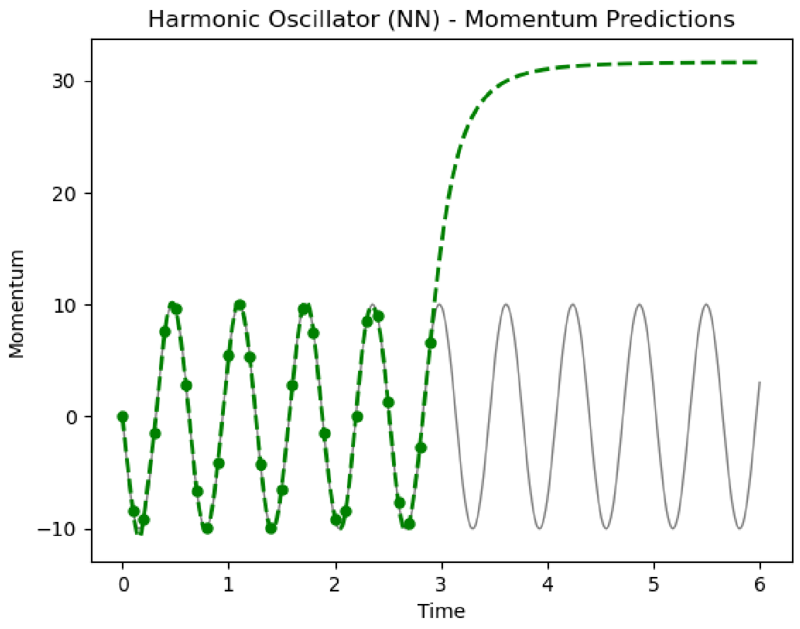

Figure 4.5. Momentum prediction of harmonic oscillator neural network. Green dashed line represent the position prediction of the model. Green dots are training data that we train the model. The gray line represents the true position.

4.1.2. Utilizing a Physics-Informed Neural Network for Solving the Harmonic Oscillator

Now the aim is to extend the model to more than only data-driven solutions and tackle scenarios where data is not given. In order to achieve this, the capacity of the model was increased and the physical equations were followed.

In this enhanced model, the capacity has been expanded. This has been achieved by increasing the number of hidden layers to 4, each comprising 32 units, and extending the training duration to $40,000$ epochs. As in the previous model, we employ the hyperbolic tangent activation function and the Adam optimizer with a learning rate of 0.01, while implementing a learning rate scheduler with a step size of 1000 and a decay factor gamma of 0.8. However, in this iteration, we increase the weight decay to 0.3.

We also add the final data points of each position and momentum into the training dataset. This inclusion ensures that the model considers the current state of the system, which contributes to its ability to make accurate predictions and simulate the dynamics of the physical system.

To strengthen the loss function by adding an additional component called "physics loss". The physics loss aims to ensure that the learned solution is consistent with the differential equation. This loss evaluates whether the predicted matches the physics formulas the by MSE loss between the automatic differentiation of models output and the derivative obtained with the physical equation using models output. To derive the physics loss function, we use Hamilton's equations of motion as defined in Equation 2.17, we rewrite these equations as follows:

$$\dot{x} = \frac{\partial \mathcal{H}}{\partial p} = \frac{p}{m}, \tag{4.10}$$

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial x} = -kx. \tag{4.11}$$

Accordingly, the physics loss functions for position and momentum are defined as follows:

$$\mathcal{L}_{physics_x} = \frac{1}{N_{physics}} \sum_{i=1}^{N_{physics}} (\dot{x}^i_{autograd} - \frac{p^i_{pred}}{m})^2, \tag{4.12}$$

$$\mathcal{L}_{physics_p} = \frac{1}{N_{physics}} \sum_{i=1}^{N_{physics}} (\dot{p}^i_{autograd} + kx^i_{pred})^2. \tag{4.13}$$

Here, $N_{physics}$ representes the number of iteration which is 600 in this case. The i^{th} derivative of position obtained by automatic differentiation [10] in Python usig the "autograd" function denoted as $\dot{x}^i_{autograd}$, and similarly for momentum $\dot{p}^i_{autograd}$. The physical equations obtained in Equation 2.17 in Hamiltonian Harmonic Oscillator Section. The mass m is 1, and the spring constant k is 100 as they are given before. The data loss and initial conditions loss are similar to the convetional neural network model. In this case, however, the number of data points has been increased by one, corresponding to the final data points. Adding physics losses to the total loss, we obtain:

$$\mathcal{L}_{total} = \mathcal{L}_{physics} + \mathcal{L}_{data} + \mathcal{L}_{IC}. \tag{4.14}$$

Here, loss functions with the obtained weights:

$$\mathcal{L}_{physics} = 10^7 \cdot \mathcal{L}_{physics_x} + 10^5 \cdot \mathcal{L}_{physics_p} \tag{4.15}$$

$$\mathcal{L}_{data} = 10^8 \cdot \mathcal{L}_{data_x} + 10^6 \cdot \mathcal{L}_{data_p} \tag{4.16}$$

$$\mathcal{L}_{IC} = 10^3 \cdot \mathcal{L}_{IC_x} + 10^2 \cdot \mathcal{L}_{IC_p} \tag{4.17}$$

In the PINN model for the harmonic oscillator, we have observed in predictions shown in Figures 4.7 and 4.8, and losses shown in Figure 4.6 a remarkable capability: Even when provided with only a limited amount of data in conjunction with the physical equations, the model is able to generalizes correctly for unseen data with these parameters and initial conditions. This indicates that the model's ability has increased

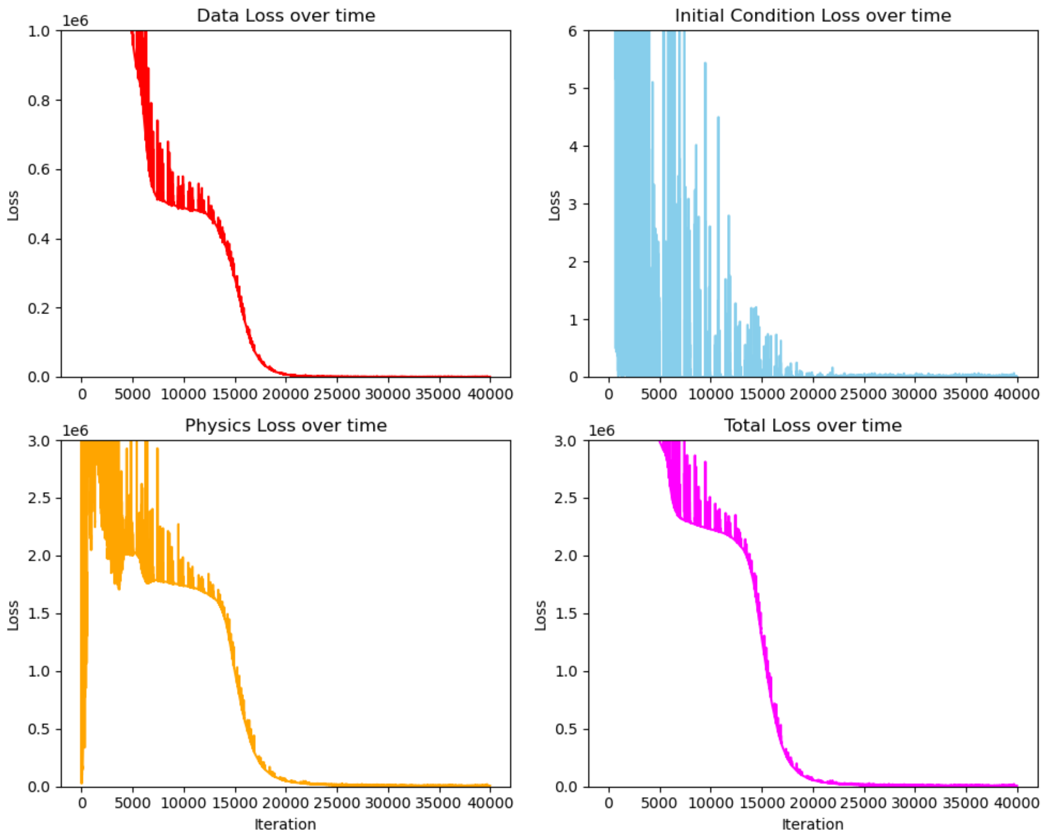

Figure 4.6. The loss results of the harmonic oscillator forward model solved with a physics-informed neural network are presented. The data loss over iterations is represented by the red curve, the initial condition loss by the sky blue curve, the physics loss by the orange curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

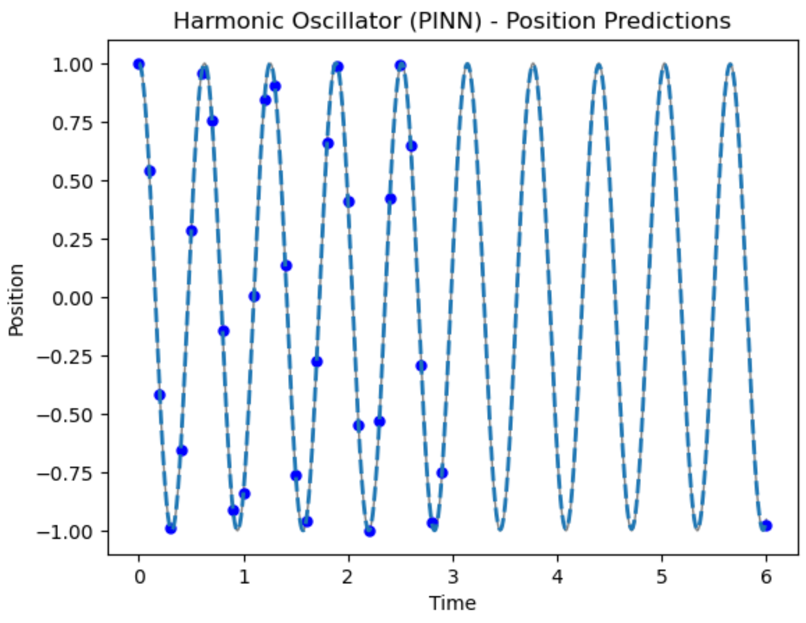

Figure 4.7. The predicted position of the harmonic oscillator model, solved with PINN, is represented by the blue curve. The blue dots represent the training position data.

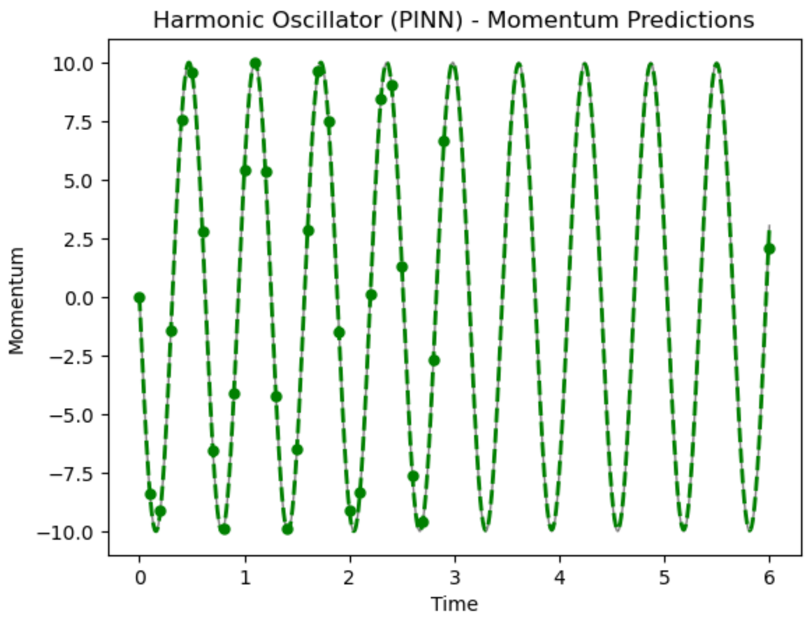

Figure 4.8. The predicted momentum for the harmonic oscillator model, solved with PINN, is represented by the green curve. The green dots represent the training momentum data.

and that it is capable of effectively solving problems even in scenarios where complete data is unavailable.

4.1.3. Utilizing the Hamiltonian Approach and Physics-Informed Neural Network for Solving the Harmonic Oscillator

This model aims to solve the problem using only initial data as input and making use of the conservation of energy, in a similar manner to the approach employed in HNNs. To achieve this, we enhance the capacity of the previous PINN model and add the "energy loss".

The capacity of the model is enhanced by increasing the number of hidden layers to six, with each layer containing 64 units. The previous weight decay of 0.3 is not modified. However, the learning rate is adjusted to 0.001 and the Adam optimizer is employed. Furthermore, the learning rate scheduler was enhanced by increasing the decay parameter gamma to 0.9 and setting the step size to $3,000$. It is noteworthy that the training duration has been significantly extended to $120,000$ epochs.

In this model, the total loss function is designed to fit three distinct components: physics loss, initial data loss, and energy loss. Equation 2.16 forms the basis for constructing the energy loss equation, rewriting this equation:

$$\mathcal{H} = \frac{kx^2}{2} + \frac{p^2}{2m}. \tag{4.18}$$

Accordingly, the energy loss function is defined as:

$$\mathcal{L}_{energy} = \frac{1}{N_{energy}} \sum_{i=1}^{N_{energy}} \left[\left(\frac{k(x^i_{pred})^2}{2} + \frac{(p^i_{pred})^2}{2m} \right) - \left(\frac{kx_0^2}{2} + \frac{p_0^2}{2m} \right) \right]^2 . \tag{4.19}$$

where x^i_{pred} denotes the i^{th} term of the prediction of model for position and x_0 is the initial position data which is given as 1. Similarly, p^i_{pred} denotes the i^{th} term of the prediction of model for momentum, and p_0 is the initial momentum data which is given as 0. The spring constant k is given 100 in this case, m is the mass of the system is given 1. The number of data points, N_{energy}, utilized for the energy loss calculation is 600. By incorporating energy loss into the total loss function, the conservation of

Hamiltonian is enforced.

$$\mathcal{L}_{total} = \mathcal{L}_{physics} + \mathcal{L}_{IC} + 10^6 \cdot \mathcal{L}_{energy}, \tag{4.20}$$

where:

$$\mathcal{L}_{physics} = 10^6 \cdot \mathcal{L}_{physics_x} + 10^5 \cdot \mathcal{L}_{physics_p}, \tag{4.21}$$

$$\mathcal{L}_{IC} = 10^4 \cdot \mathcal{L}_{IC_x} + 10^3 \cdot \mathcal{L}_{IC_p}. \tag{4.22}$$

The progression of losses over iterations is shown in Figure 4.9.

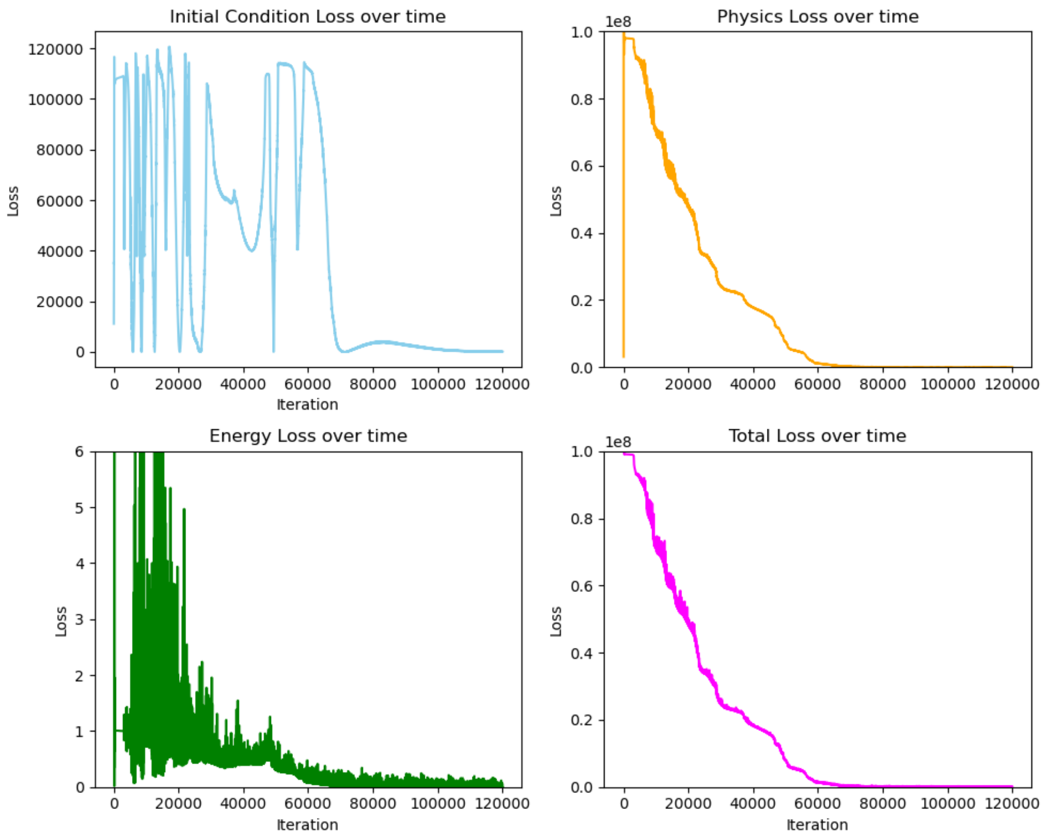

Figure 4.9. The loss results of the harmonic oscillator forward model solved with the combination of Hamiltonian approach and PINN are presented. The initial condition loss over iterations is represented by the sky blue curve, the physics loss by the orange curve, the energy loss by the green curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

Figures 4.10 and 4.11 demonstrate that this PINN model, which employs the

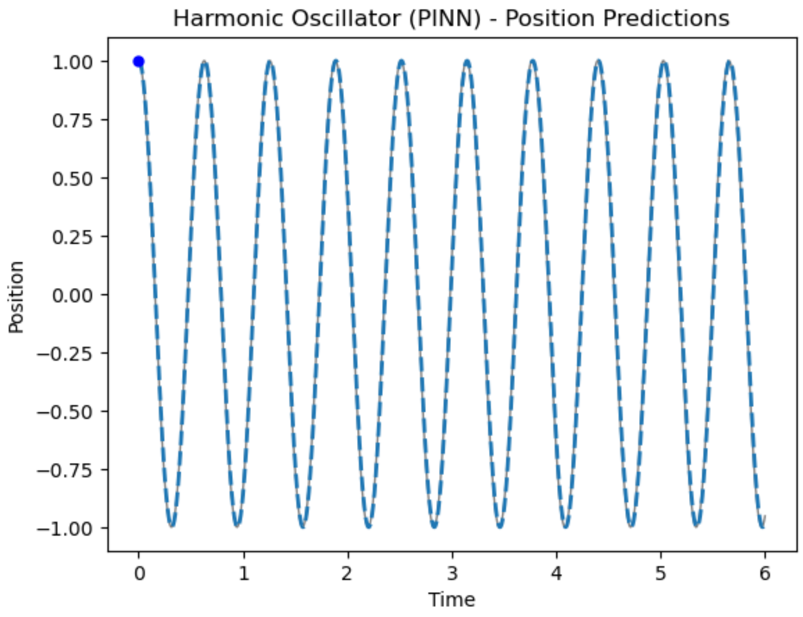

Figure 4.10. The position prediction for the harmonic oscillator model, solved with the combination of Hamiltonian approach and PINN, are represented by the blue curve. The blue dot represents initial position data.

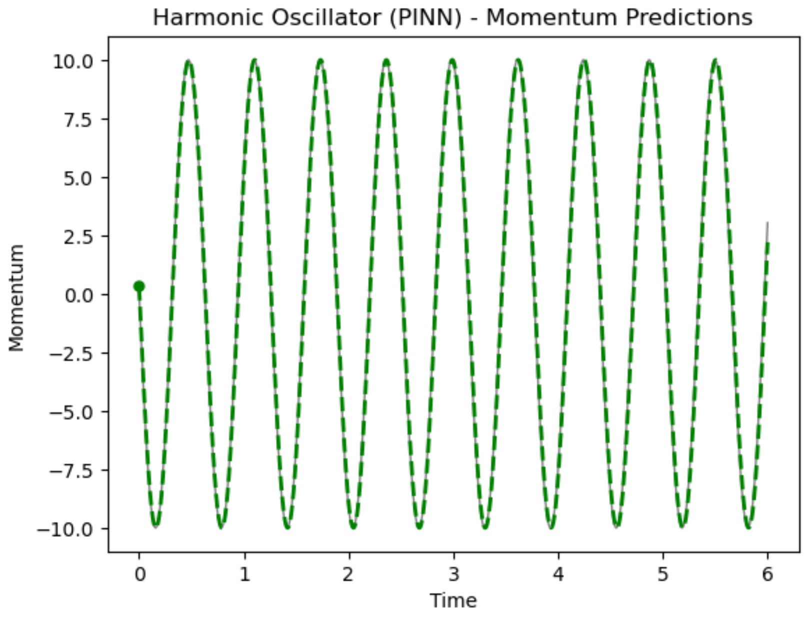

Figure 4.11. The momentum prediction for the harmonic oscillator model, solved with the combination of Hamiltonian approach and PINN, are represented by the green curve. The green dot represents the initial momentum data.

Hamiltonian approach, exhibits superior predictive capabilities compared to previous models. The model's predictions are accurate, and it is capable of accounting for physical phenomena and energy losses, which depend on initial conditions, without requiring additional data. It is important to point out that when the same model is run without consideration of energy loss, it fails to solve the problem.

4.2. Ray-Tracing Equations

In Section 2.4, we derive the ray-tracing equations in the linearly heterogeneous medium and examine various scenarios, including isotropic, elliptically anisotropic, and generalized elliptically anisotropic velocity models. In this section, we will use physics-informed neural networks to solve the ray-tracing equations for each of the scenarios. A comparison will be made between the PINN result and the analytic equations derived in previous sections.

4.2.1. Solving the Ray-Tracing Equations for Isotropic Velocity Model with PINNs

The ray-tracing equations for isotropic velocity model is derived in Subsection 2.4.1. Here, our initial goal is to solve the ray-tracing equations as the forward problem. In this problem we provide the take-off angle, velocity, and time as inputs, from which we derive the ray path. Furthermore, we provided initial conditions and selected parameter values for the system to predict correctly. These provided values are listed below together:

$$\begin{aligned}
\text{initial coordinates}: &\quad x(0)=0, \quad z(0)=0,\\
\text{initial momenta}: &\quad p_x(0)=\frac{\sin\vartheta_0}{v_0}, \quad p_z(0)=\frac{\cos\vartheta_0}{v_0},\\
\text{initial velocity}: &\quad v_0=a,\\
\text{velocity model parameters}: &\quad a=2, \quad b=1,\\
\text{take}-\text{off angle}: &\quad \vartheta_0=\frac{\pi}{12},
\end{aligned}$$

$$\text{travel time :} \qquad t_{max} = 5.$$

The decrease of the take-off angle of the ray represents a further complicating factor in the problem as it approaches the singularity. Consequently, the value of ϑ is set to $\frac{\pi}{12}$ in order to guarantee the efficacy of the model.

In order to determine the capacity of the model for learning, the analytic solution function of ray equations is constructed to generate true data in accordance with the initial conditions. The ray equations referenced in Equation 2.72 were employed in generating true data, rewriting these equations as:

$$x(t) = \frac{a}{bsin\vartheta}\left(\tanh(bt - \operatorname{arctanh}(\cos\vartheta)) + \cos\vartheta\right), \tag{4.23}$$

$$z(t) = \frac{a}{b}\left(\frac{1}{\sin\vartheta}\cosh(\operatorname{arctanh}(\cos\vartheta) - bt)\right), \tag{4.24}$$

$$p_x(t) = \frac{\sin\vartheta}{a}, \tag{4.25}$$

$$p_z(t) = \frac{\sin\vartheta}{a}\sinh(\operatorname{arctanh}(\cos\vartheta) - bt). \tag{4.26}$$

These equations use time t as the input variable and have four outputs: two coordinates x and z, and two momenta p_x and p_z. So, when constructing our neural network model, we give the input as time and the outputs as x, z, p_x and p_z. And the network comprises 6 hidden layers, with each layer containing 48 neurons. The hyperbolic tangent activation function is employed as the activation function. The Adam optimizer is employed for model parameter optimization, with a learning rate of 0.001. A learning rate scheduler is implemented with a step size of 1000 and a decay factor gamma of 0.9. The weight decay is set at 0.1. The model was then iterated over $30,000$ epochs.

In light of the findings from the investigation of the harmonic oscillator with neural networks, we have applied the Hamiltonian and PINNs to the ray-tracing equations. This includes the incorporation of the physics loss, initial loss, and energy loss components within the total loss function. Explanation of the specific types of losses

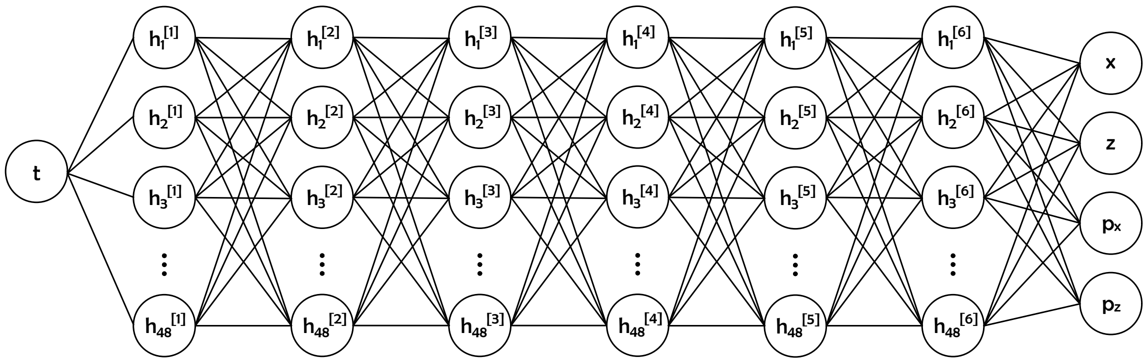

Figure 4.12. Illustration of the neural network.

is provided below. It begins with an examination of initial conditions losses:

$$\mathcal{L}_{IC_x} = \left(x_{0pred} - 0\right)^2, \tag{4.27}$$

$$\mathcal{L}_{IC_z} = \left(z_{0pred} - 0\right)^2, \tag{4.28}$$

$$\mathcal{L}_{IC_{px}} = \left(p_{x0pred} - \frac{\sin \vartheta_0}{a}\right)^2, \tag{4.29}$$

$$\mathcal{L}_{IC_{pz}} = \left(p_{z0pred} - \frac{\cos \vartheta_0}{a}\right)^2. \tag{4.30}$$

The physics loss contains the Hamilton equations of motion. For the sake of clarity, we will repeat the equations presented in Equation 2.64:

$$\dot{x} = (a + bz)^2 p_x, \tag{4.31}$$

$$\dot{z} = (a + bz)^2 p_z, \tag{4.32}$$

$$\dot{p_x} = 0, \tag{4.33}$$

$$\dot{p_z} = -b(a + bz)(p_x^2 + p_z^2). \tag{4.34}$$

The physics losses can be quantified as follows:

$$\mathcal{L}_{physics_x} = \sum_{i=1}^{N_{physics}} \left(\dot{x}^i_{autograd} - (a + bz^i_{pred})^2 p_{x\,pred}^{\;i}\right)^2, \tag{4.35}$$

$$\begin{aligned}
\mathcal{L}_{physics_z} &= \sum_{i=1}^{N_{physics}} \left(\dot{z}^i_{autograd} - (a+bz^i_{pred})^2 p_{z}{}^i_{pred}\right)^2, && (4.36)\\
\mathcal{L}_{physics_{p_x}} &= \sum_{i=1}^{N_{physics}} \left(\dot{p_x}{}^i_{autograd} - 0\right)^2, && (4.37)\\
\mathcal{L}_{physics_{p_z}} &= \sum_{i=1}^{N_{physics}} \left(\dot{p_z}{}^i_{autograd} + b(a+bz^i_{pred})((p_{x}{}^i_{pred})^2 + (p_{z}{}^i_{pred})^2)\right)^2. && (4.38)
\end{aligned}$$

Consequently, to conserve the energy of the system, it is necessary to define a loss function. This will be constructed using the Equation 2.63, which is rewritten as follows:

$$\mathcal{H} = \frac{(a+bz)^2(p_x^2+p_z^2)-1}{2}. \tag{4.39}$$

Finally, the energy loss is defined as:

$$\mathcal{L}_{energy} = \sum_{i=1}^{N_{energy}} \left(\frac{(a+bz^i_{pred})^2[(p_{x}{}^i_{pred})^2 + (p_{z}{}^i_{pred})^2]-1}{2}\right)^2. \tag{4.40}$$

In the equations physics loss and energy loss, the length of training time, which is represented by the symbols $N_{physics}$ and N_{energy}, is equal to 501.

The total loss can be represented as:

$$\mathcal{L}_{total} = 10^2 \cdot \mathcal{L}_{physics} + 10^2 \cdot \mathcal{L}_{IC} + 10^2 \cdot \mathcal{L}_{energy}, \tag{4.41}$$

where:

$$\begin{aligned}
\mathcal{L}_{physics} &= \mathcal{L}_{physics_x} + \mathcal{L}_{physics_y} + 10^2 \cdot \mathcal{L}_{physics_{p_x}} + 10^2 \cdot \mathcal{L}_{physics_{p_z}}, && (4.42)\\
\mathcal{L}_{IC} &= \mathcal{L}_{IC_x} + \mathcal{L}_{IC_z} + 10^2 \cdot \mathcal{L}_{IC_{p_x}} + 10^2 \cdot \mathcal{L}_{IC_{p_z}}. && (4.43)
\end{aligned}$$

Here, we mentioned total loss and the loss weights obtained through a trial and error method.

The loss results over iterations of this model is shown in Figure 4.13. It is evident that the losses are consistently minimized until they reach zero.

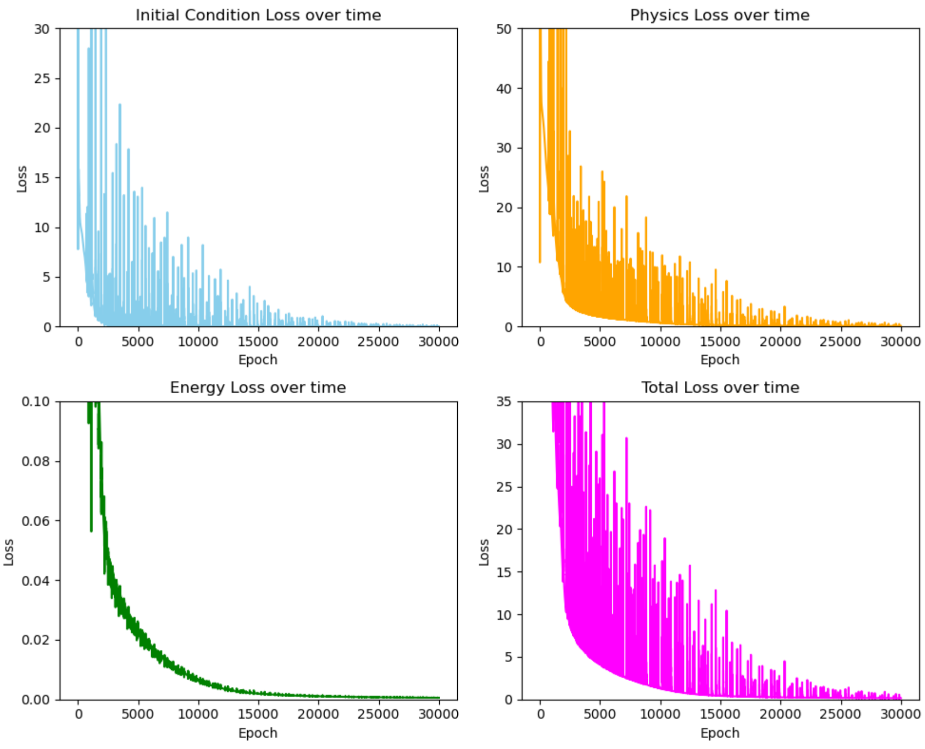

Figure 4.13. The loss results of the ray-tracing equations forward model solved with the combination of Hamiltonian approach and PINN are presented. The initial condition loss over iterations is represented by the sky blue curve, the physics loss by the orange curve, the energy loss by the green curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

Figure 4.14 illustrates the ray path prediction of this neural network model. The prediction aligns with the analytical solution for the ray path.

This model is applicable to any take-off angle. To provide further evidence, we will examine the results for different angle values. In the initial example, by slightly increasing the angle value, we obtain $\frac{\pi}{9}$. Figure 4.15 demonstrates the output of the model, and Figure 4.17 shows the loss results. Similarly, by slightly increasing the angle value, we obtain $\frac{\pi}{6}$. Figure 4.16 demonstrates the output of the model, and Figure 4.18 shows the loss results.

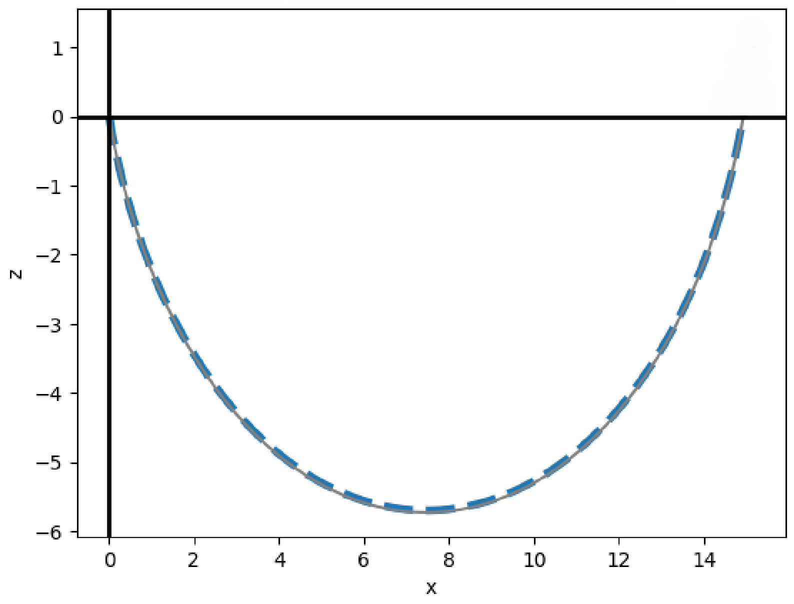

Figure 4.14. Ray path prediction of the neural network model. The blue dashed curve depicts the neural network's predictive model, aligning with the gray curve, representing the analytical solution of the ray path. This alignment emphasizes the accuracy and reliability of our predictive model.

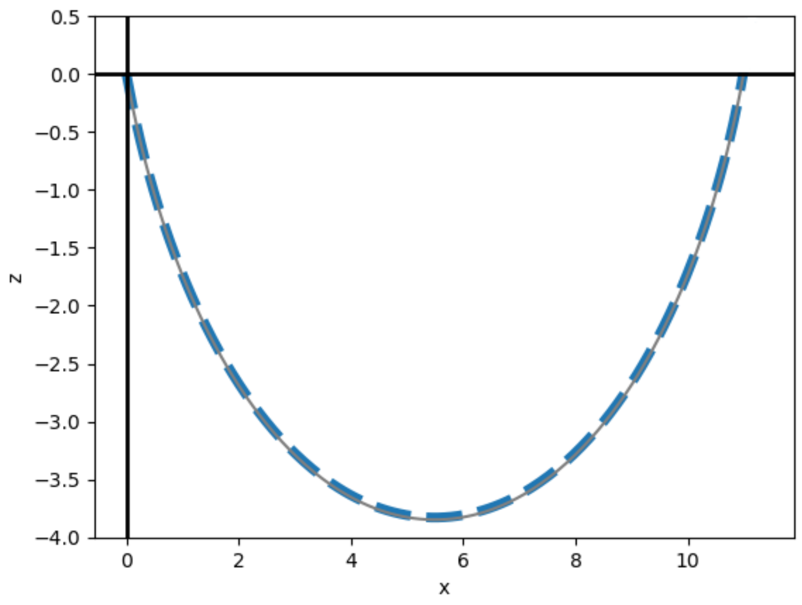

Figure 4.15. The prediction of this PINN model for the take-off angle $\frac{\pi}{9}$. The blue dashed curve depicts the neural network's predictive model, perfectly aligning with the gray curve, representing the analytical solution of the ray path.

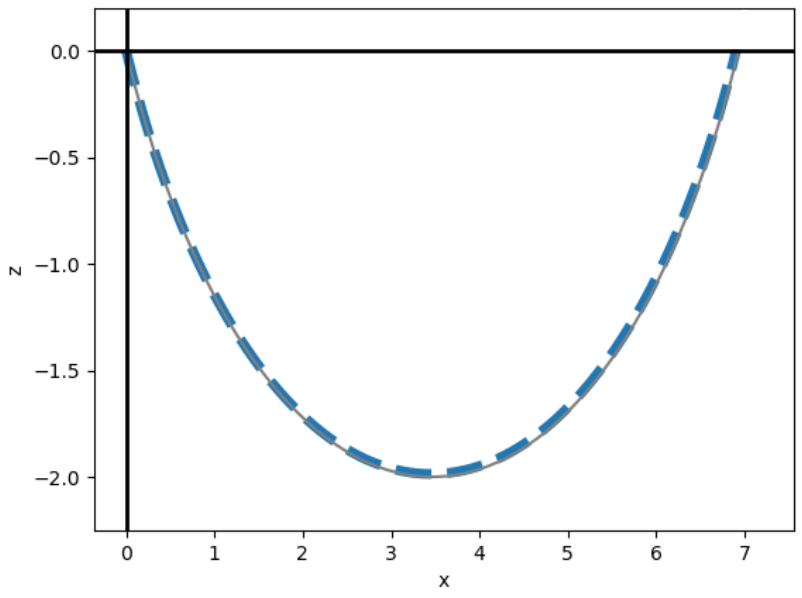

Figure 4.16. The prediction of this PINN model for the take-off angle $\dfrac{\pi}{6}$. The blue dashed curve depicts the neural network's predictive model, aligning with the gray curve, representing the analytical solution of the ray path.

4.2.2. Solving the Inverse Problem of the Ray-Tracing Equations for Isotropic Velocity Model with PINNs

In the inverse model of ray-tracing equations, we provide the coordinates of the station and source, along with the travel time. While the travel time duration of $t_{max} = 5$ is sufficient for the forward method, we prefer a longer training time for inversion, so we set $t_{max} = 100$. Then, we employ a neural network model to solve for the take-off angle ϑ and the velocity model, represented by parameters a and b. The network model is similar to the forward method. However, the number of epoch is increased to $40,000$, and the step size of the learning rate scheduler is increased to $2,500$. The Adam optimizer is used not only to optimize the model parameters, but also to refine the estimates for variables such as the take-off angle ϑ and the parameters a and b within the model.

To verify the accuracy of a model, it is necessary to generate a true solution. In this inverse model, two rays we want to solve. This is the reason why two rays were

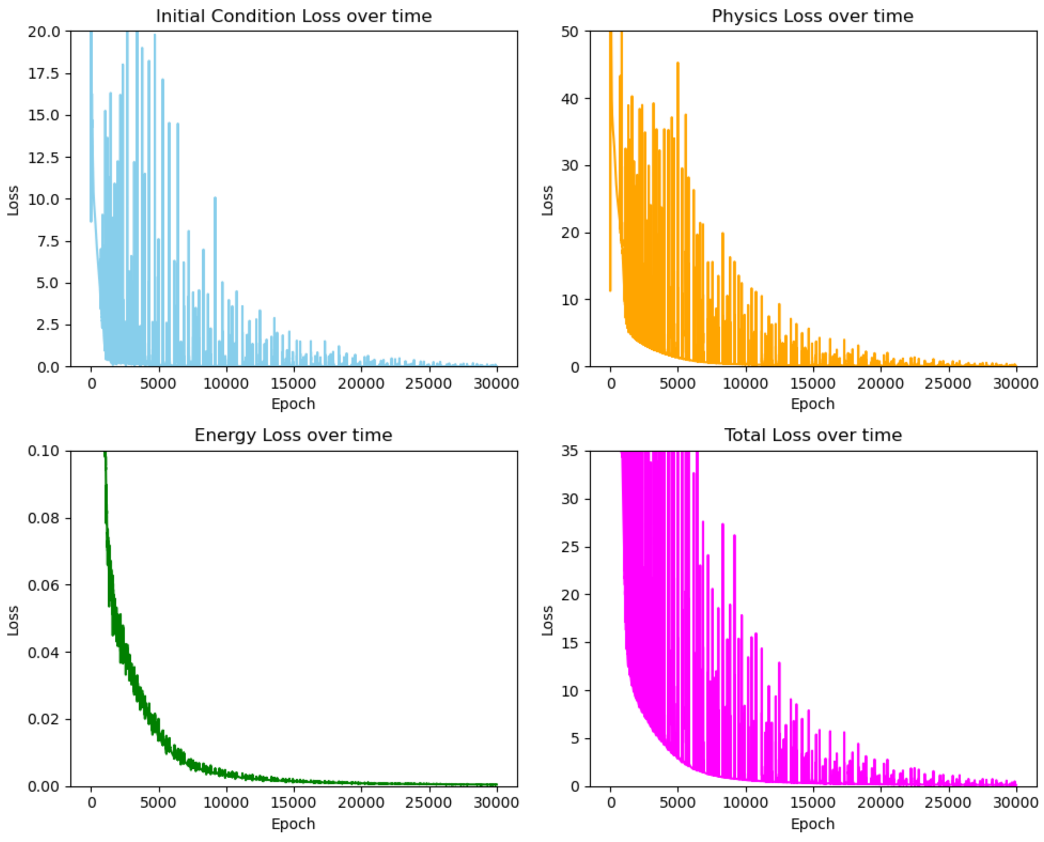

Figure 4.17. The loss results of the ray-tracing equations for the take-off angle $\frac{\pi}{9}$ solved with this PINN are presented. The initial condition loss over iterations is represented by the sky blue curve, the physics loss by the orange curve, the energy loss by the green curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

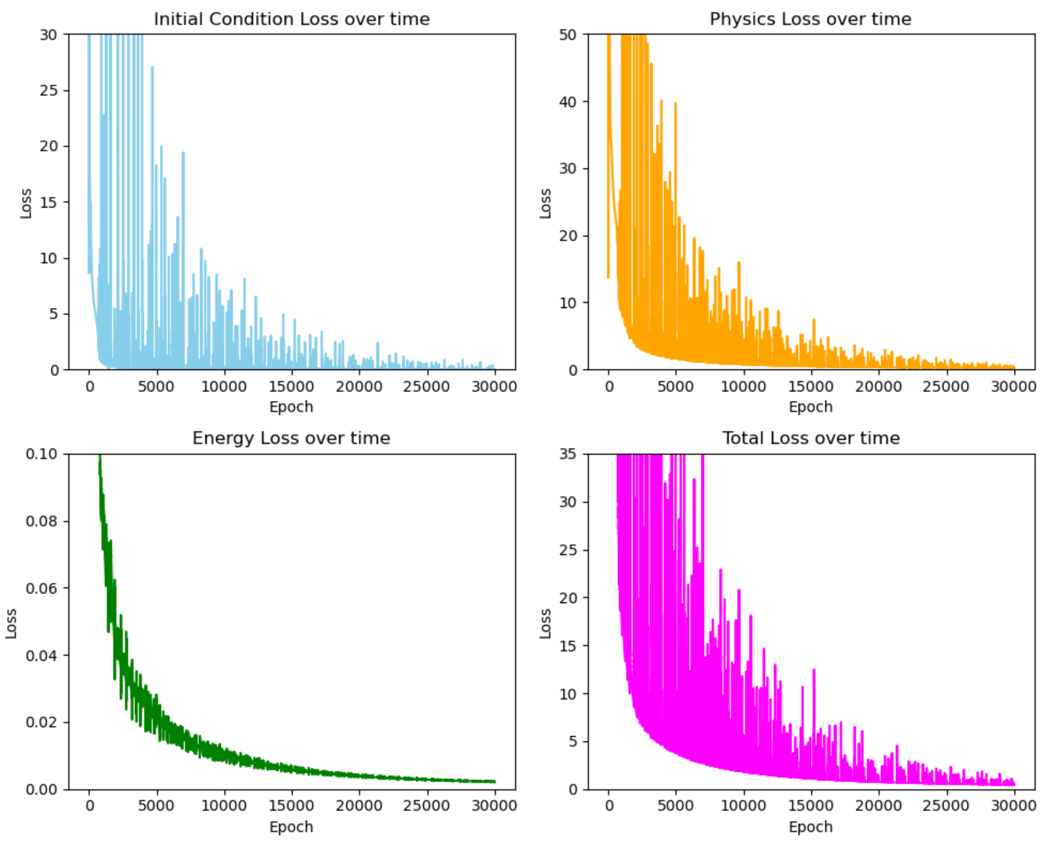

Figure 4.18. The loss results of the ray-tracing equations for the take-off angle $\dfrac{\pi}{9}$ solved with this PINN are presented. The initial condition loss over iterations is represented by the sky blue curve, the physics loss by the orange curve, the energy loss by the green curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

set up, each defined by its respective angles:

$$\begin{aligned}\vartheta_1 &= \frac{\pi}{8},\\ \vartheta_2 &= \frac{\pi}{6}.\end{aligned}$$

And, the true velocity parameters are as follows:

$$\begin{aligned}a_{true} &= 2,\\ b_{true} &= 1.\end{aligned}$$

Following this, we define incorrect initial guesses:

$$\begin{aligned}a_{guess} &= 1,\\ b_{guess} &= 0.5,\\ \vartheta_{guess} &= \frac{\pi}{2}.\end{aligned}$$

These predictions serve as a test to evaluate the model's ability to converge to the correct solutions throughout the optimization process.

The calculation of the loss functions remains similar to the forward method, with one notable change; the velocity term is modified as follows:

$$v^i = a_{guess} + b_{guess} z^i. \tag{4.44}$$

Here we substitute the values of a and b with those a_{guess} and b_{guess}. Additionally, we introduce another loss function aimed at minimizing the final coordinate points, computed as:

$$\mathcal{L}_{shoot_x} = \left(x_{pred} - x_{final}\right)^2, \tag{4.45}$$

$$\mathcal{L}_{shoot_z} = \left(z_{pred} - z_{final}\right)^2. \tag{4.46}$$

We positioned the station at the coordinates (x_{final}, z_{final}), where $x_{final} = 6.92$ and $z_{final} = 0$. Using these coordinates, the model predicts the source location.

Another change has been made to the training duration. The physics loss and energy loss, the number of training data, denoted as $N_{physics}$ and N_{energy}, respectively, are updated to 264. Consequently, the total loss function is defined as:

$$\mathcal{L}_{total} = 10^2 \cdot \mathcal{L}_{physics} + 10^2 \cdot \mathcal{L}_{IC} + 10^2 \cdot \mathcal{L}_{shoot} + 10^2 \cdot \mathcal{L}_{energy}, \tag{4.47}$$

where:

$$\begin{aligned} \mathcal{L}_{physics} &= \mathcal{L}_{physics_x} + \mathcal{L}_{physics_y} + 10^2 \cdot \mathcal{L}_{physics_{px}} + 10^2 \cdot \mathcal{L}_{physics_{pz}}, & (4.48) \\ \mathcal{L}_{IC} &= \mathcal{L}_{IC_x} + \mathcal{L}_{IC_z} + 10^2 \cdot \mathcal{L}_{IC_{px}} + 10^2 \cdot \mathcal{L}_{IC_{pz}}, & (4.49) \\ \mathcal{L}_{shoot} &= \mathcal{L}_{shoot_x} + \mathcal{L}_{shoot_z}. & (4.50) \end{aligned}$$

The progress of these losses are depicted in Figure 4.19, with each gradually converging to zero.

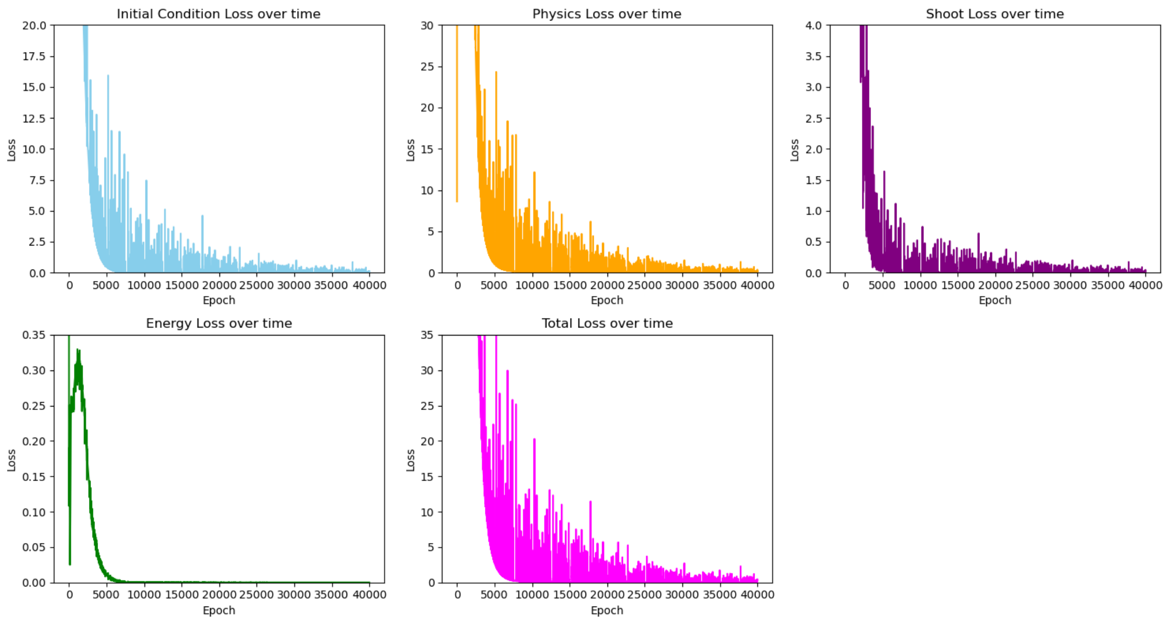

Figure 4.19. The loss results of the ray-tracing equations inverse model solved with the combination of Hamiltonian approach and PINN are presented. The initial condition loss over iterations is represented by the sky blue curve, the physics loss by the orange curve, the shoot loss by the purple curve, the energy loss by the green curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

Observing the convergence of both a and b in Figure 4.20 provides insight into how they gradually approach their true values. Furthermore, examining the convergence of the take-off angles ϑ_1 and ϑ_2 in Figure 4.21 provides insights into the evolution of these angles over time. These convergences demonstrate the efficacy of the optimization process in converging to the true values and indicate that the model is making progress towards minimizing the overall loss.

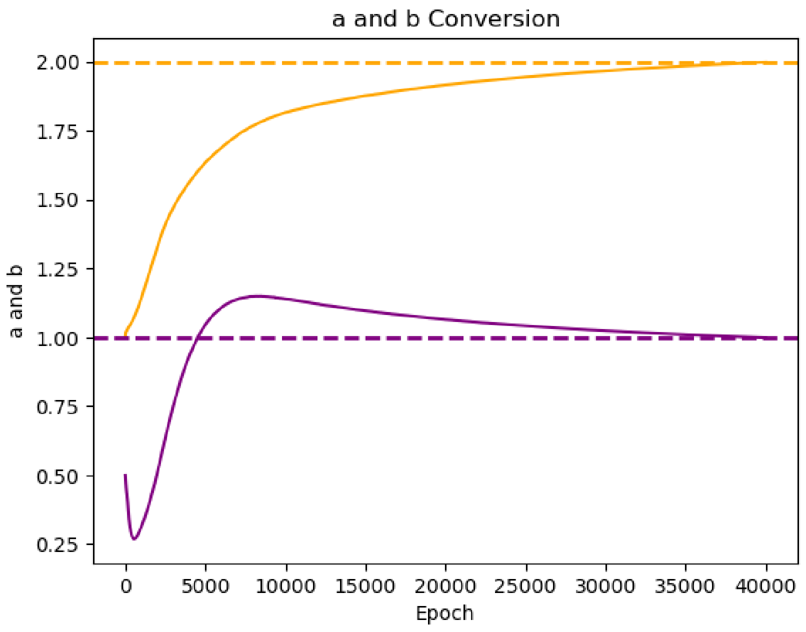

Figure 4.20. The convergence trajectories of a and b are plotted over the course of the optimization process. The orange curve illustrates the evolution of a, while the purple curve tracks the changes in b. The initial guesses for these parameters, $a_{guess} = 1$ and $b_{guess} = 0.5$, serve as the starting points for the optimization. As the optimization progresses, the model refines its estimates, resulting in $a_{pred} = 1.9992$ and $b_{pred} = 0.9985$. These predicted values are close to the true parameters, $a_{true} = 2$ and $b_{true} = 1$.

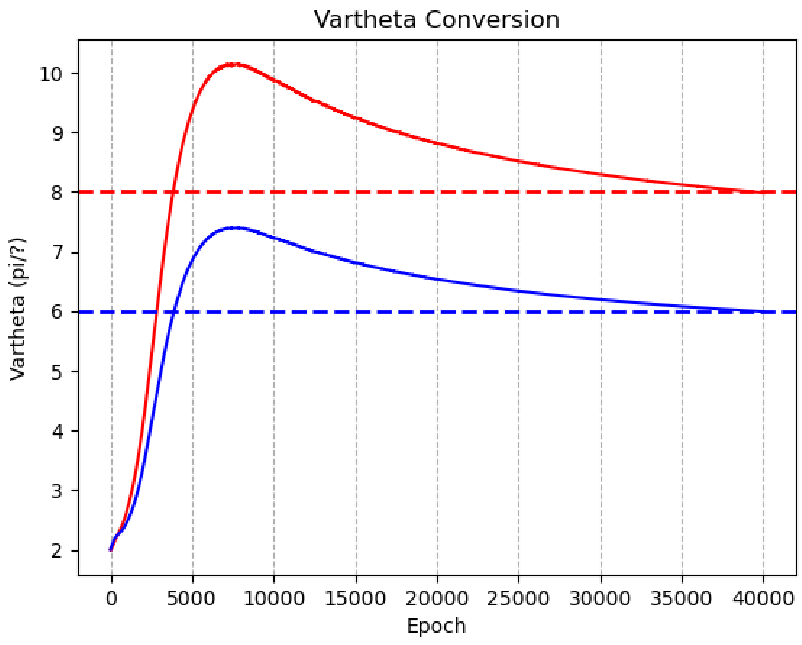

Figure 4.21. The convergence trajectories of take-off angles ϑ_1 and ϑ_2 are illustrated in this figure. The initial guess for both angles, $\vartheta_{guess} = \dfrac{\pi}{2}$ serves as the starting point. As the optimization iterates, the model refines its estimates. For the red curve, the model's predictions are $\vartheta_1^{pred} = \dfrac{\pi}{7.9816}$, which closely approximates the true parameter $\vartheta_1^{true} = \dfrac{\pi}{8}$. Similarly, for the blue curve, the model predicts $\vartheta_2^{pred} = \dfrac{\pi}{5.9918}$, which is close to the true parameter $\vartheta_2^{true} = \dfrac{\pi}{6}$.

The ray path predictions of each take-off angles are as depicted in Figure 4.22. This figure demonstrates that this network model effectively solve the inverse problem of ray-tracing equations. Initially, incorrect guesses were made for velocity model, and take-off angle. By incorporating the shoot loss into the total loss and engaging in iterative tuning, the model proves its ability to accurately predict the ray-path by identifying and converging to the correct values.

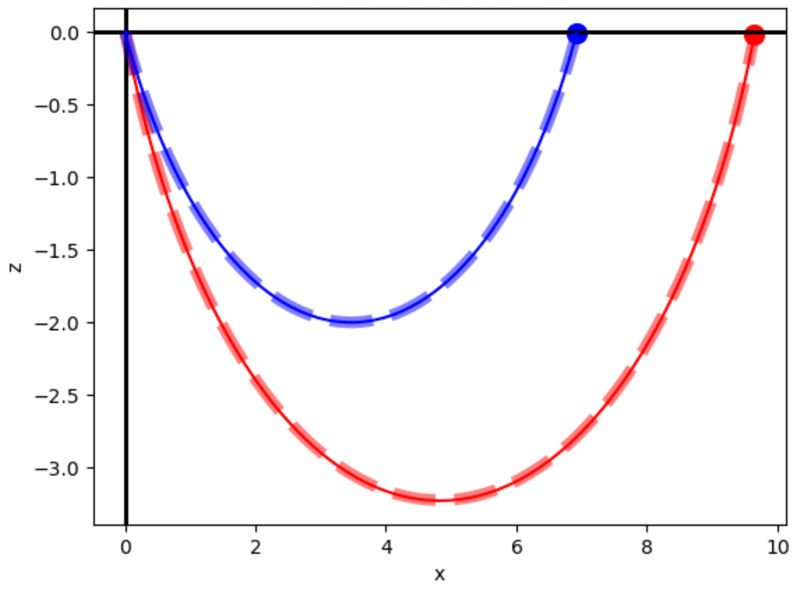

Figure 4.22. The ray path predictions of the inverse problem of the ray-tracing equations are illustrated in this figure. For the red curves, the thin curve represents the true ray path of the true angle $\vartheta_1^{true} = \frac{\pi}{8}$, while the dashed curve represents the predicted ray path of the predicted angle $\vartheta_1^{pred} = \frac{\pi}{7.9816}$. The red dot represents the receiver for the corresponding ray path. Similarly, for the blue curves, the thin curve represents the true ray-path of the true angle $\vartheta_2^{true} = \frac{\pi}{6}$, while the dashed curve represents the predicted ray-path of the predicted angle $\vartheta_2^{pred} = \frac{\pi}{5.9918}$. The blue dot represents the receiver for the corresponding ray path.

4.2.3. Solving the Ray-Tracing Equations for Constant Elliptical Anisotropic Velocity Model with PINNs

The model employing ray-tracing equations for an isotropic velocity model in a linearly heterogeneous medium is adapted for an anisotropic velocity model with the velocities depending on elliptically in the medium. The details of this explanation can be found in Section 2.4.2. This model maintains the same structure and layers. The input layer receives input t, and the output layer generates four outputs, namely x, z, p_x, and p_z. There is six hidden layers, each containing 48 neurons. The hyperbolic tangent activation function is used as the activation function. The Adam optimizer is employed to optimize the model parameters, with a learning rate of 0.001. A learning rate scheduler is implemented with a step size of 1000 and a decay factor gamma of 0.9. The weight decay is set to 0.1. However, in this model, the number of epochs is increased to $45,000$.

To check accuracy of the model, an analytic solution is generated utilising the integral curves derived in Equation 2.92. The integral curves are as follows:

$$\begin{aligned}
x(t) &= \frac{1}{p_x b}\left[\tanh(bt - \operatorname{arctanh}(ap_{z0}) + ap_{z0}\right], && (4.51)\\
z(t) &= \frac{a}{b}\left[\frac{1}{p_x a\sqrt{1+2\chi}\cosh(\operatorname{arctanh}(ap_{z0}-bt)} - 1\right], && (4.52)\\
p_x(t) &= \frac{\sin\vartheta}{a\sqrt{(1+2\chi)\sin^2\vartheta + \cos^2\vartheta}}, && (4.53)\\
p_z(t) &= p_x\sqrt{1+2\chi}\sinh(\operatorname{arctanh}(ap_{z0}-bt). && (4.54)
\end{aligned}$$

Initial conditions written in Equation 2.88 given to model listed below:

$$\begin{aligned}
\text{initial coordinates :}\quad & x(0)=0,\quad z(0)=0,\\
\text{initial momenta :}\quad & p_x(0)=\frac{\sin\vartheta_0}{v_0},\quad p_z(0)=\frac{\cos\vartheta_0}{v_0},\\
\text{initial velocity :}\quad & v_0 = a\sqrt{(1+2\chi)\sin^2\vartheta_0+\cos^2\vartheta_0},
\end{aligned}$$

$$\begin{aligned}
\text{velocity model parameters}: &\quad a = 2, \quad b = 1,\\
\text{degree of anisotropy}: &\quad \chi = 0.3,\\
\text{take} - \text{off angle}: &\quad \vartheta_0 = \frac{\pi}{12},\\
\text{travel time}: &\quad t_{max} = 5.
\end{aligned}$$

The initial conditions loss functions are defined by utilizing the aforementioned initial conditions, and these functions are written as follows:

$$\mathcal{L}_{IC_x} = \left(x_{0pred} - 0\right)^2, \tag{4.55}$$

$$\mathcal{L}_{IC_z} = \left(z_{0pred} - 0\right)^2, \tag{4.56}$$

$$\mathcal{L}_{IC_{p_x}} = \left(p_{x0pred} - \frac{\sin\vartheta_0}{a\sqrt{(1+2\chi)\sin^2\vartheta_0 + \cos^2\vartheta_0}}\right)^2, \tag{4.57}$$

$$\mathcal{L}_{IC_{p_z}} = \left(p_{z0pred} - \frac{\cos\vartheta_0}{a\sqrt{(1+2\chi)\sin^2\vartheta_0 + \cos^2\vartheta_0}}\right)^2. \tag{4.58}$$

Utilizing the Hamilton's equations of motion mentioned in Equation 2.84 to evaluate physics loss function. Recalling these equations:

$$\dot{x} = (a+bz)^2(1+2\chi)p_x, \tag{4.59}$$

$$\dot{z} = (a+bz)^2 p_z, \tag{4.60}$$

$$\dot{p_x} = 0, \tag{4.61}$$

$$\dot{p_z} = -\frac{b}{a+bz}. \tag{4.62}$$

The physics loss functions are defined as follows:

$$\mathcal{L}_{physics_x} = \sum_{i=1}^{N_{physics}} \left(\dot{x}^i_{autograd} - (a+bz^i_{pred})^2(1+2\chi)p_{x}{}^i_{pred}\right)^2, \tag{4.63}$$

$$\mathcal{L}_{physics_z} = \sum_{i=1}^{N_{physics}} \left(\dot{z}^i_{autograd} - (a+bz^i_{pred})^2 p_{z}{}^i_{pred}\right)^2, \tag{4.64}$$

$$\mathcal{L}_{physics_{p_x}} = \sum_{i=1}^{N_{physics}} \left(\dot{p_x}^i_{autograd} - 0\right)^2, \tag{4.65}$$

$$\mathcal{L}_{physics_{p_z}} = \sum_{i=1}^{N_{physics}} \left(\dot{p_z}^i_{autograd} + \frac{b}{(a + bz^i_{pred})}\right)^2. \tag{4.66}$$

The energy of this system is given by the Hamiltonian derived in Equation 2.83. This can be rewritten as:

$$\mathcal{H} = \frac{(a+bz)^2\left[(1+2\chi)p_x^2 + p_z^2\right] - 1}{2}. \tag{4.67}$$

Consequently, the energy loss function is defined as:

$$\mathcal{L}_{energy} = \sum_{i=1}^{N_{energy}} \left(\frac{(a + bz^i_{pred})^2\left((1+2\chi)(p_x{}^i_{pred})^2 + (p_z{}^i_{pred})^2\right) - 1}{2}\right)^2. \tag{4.68}$$

Once the individual loss functions have been identified, the total loss function is defined as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{physics} + \mathcal{L}_{IC} + 10^5 \cdot \mathcal{L}_{energy}, \tag{4.69}$$

where:

$$\mathcal{L}_{physics} = 10^3 \cdot \mathcal{L}_{physics_x} + 10^3 \cdot \mathcal{L}_{physics_y} + 10^5 \cdot \mathcal{L}_{physics_{p_x}} + 10^5 \cdot \mathcal{L}_{physics_{p_z}}, \tag{4.70}$$

$$\mathcal{L}_{IC} = 10^3 \cdot \mathcal{L}_{IC_x} + 10^3 \cdot \mathcal{L}_{IC_z} + 10^5 \cdot \mathcal{L}_{IC_{p_x}} + 10^5 \cdot \mathcal{L}_{IC_{p_z}}. \tag{4.71}$$

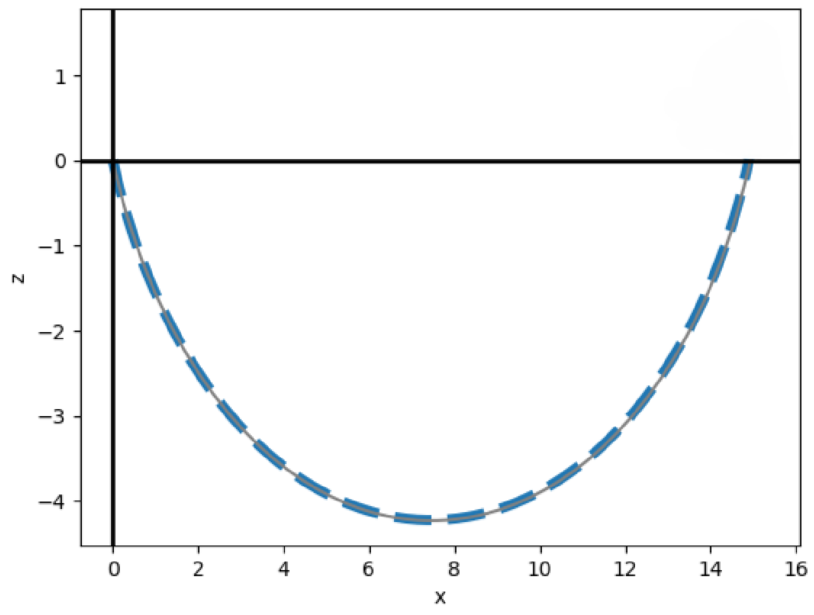

Figure 4.23. The PINN model's prediction of ray-path in a linearly heterogenous medium for elliptical anisotropic velocity model with the take-off angle of $\frac{\pi}{12}$ is shown in this figure. The blue dashed curve represents the neural network's prediction, which matches the gray curve depicting the analytical solution of the ray path.

Observations were conducted using a range of take-off angles, including $\frac{\pi}{9}$, $\frac{\pi}{6}$ and $\frac{\pi}{3}$. During these observations, it was determined that to achieve successful conversion for the angle $\vartheta = \frac{\pi}{3}$, a decrease in the maximum time was necessary. Consequently, t_{max} was set to 4 to ensure the desired conversion. The results of the neural network model for each take-off angle are presented in the figures below.

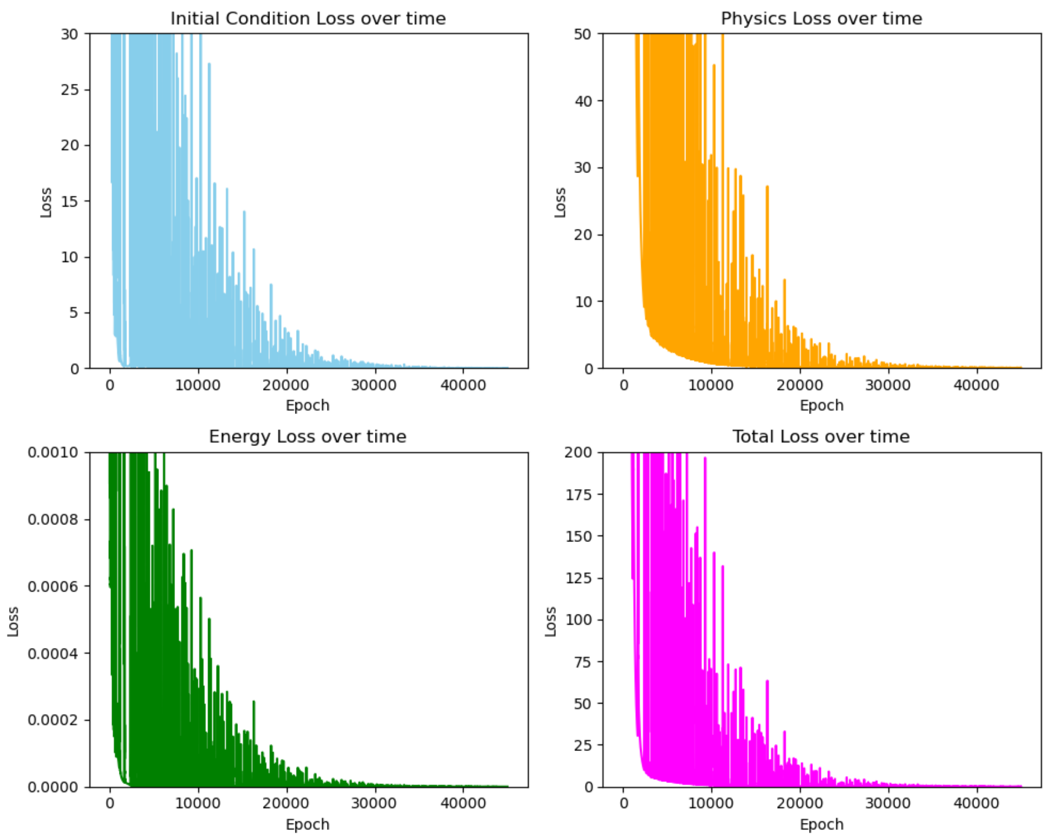

Figure 4.24. The iterative loss results of the model's prediction of ray-path in a linearly heterogenous medium for elliptical anisotropic velocity model with the take-off angle of $\frac{\pi}{12}$ are presented. The initial condition loss over iterations is represented by the sky blue curve, the physics loss by the orange curve, the energy loss by the green curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

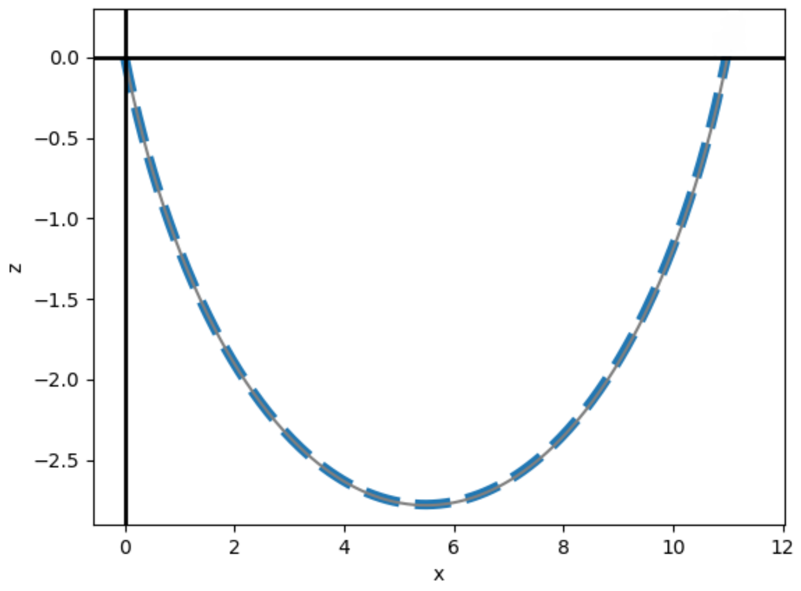

Figure 4.25. The model's prediction of ray-path for the velocity model with the take-off angle of $\frac{\pi}{9}$ is shown in this figure. The blue dashed curve represents the neural network's prediction, which matches the gray curve depicting the analytical solution of the ray path.

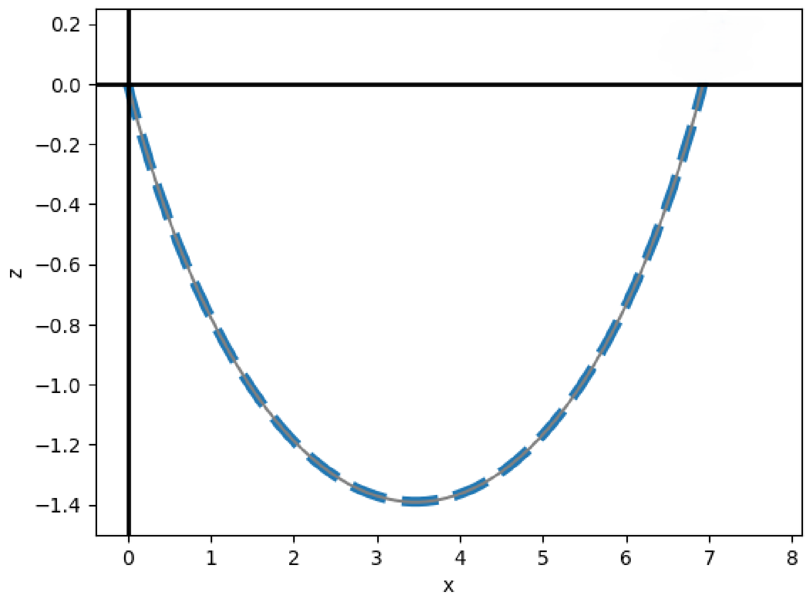

Figure 4.26. The model's prediction of ray-path for the velocity model with the take-off angle of $\frac{\pi}{6}$ is shown in this figure. The blue dashed curve represents the neural network's prediction, which matches the gray curve depicting the analytical solution of the ray path.

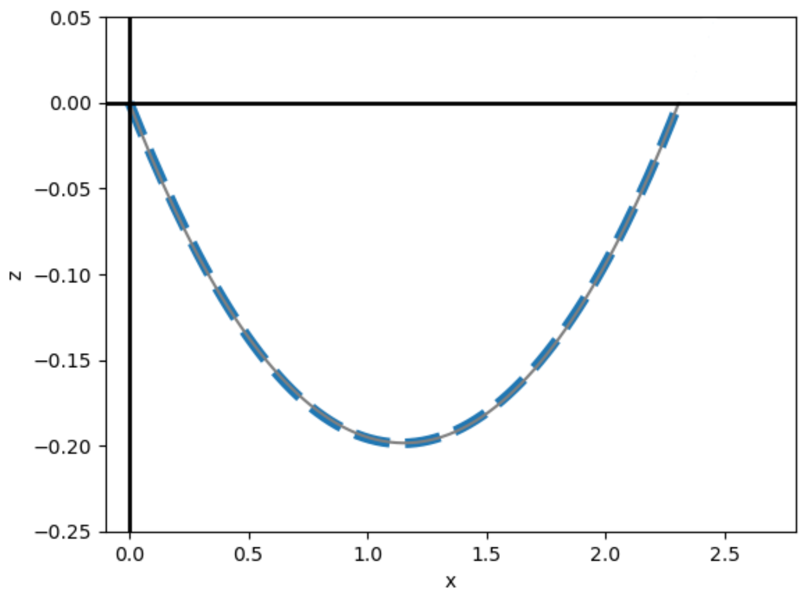

Figure 4.27. The model's prediction of ray-path for the velocity model with the take-off angle of $\frac{\pi}{3}$ is shown in this figure. The blue dashed curve represents the neural network's prediction, which matches the gray curve depicting the analytical solution of the ray path.

4.2.4. Solving the Ray-Tracing Equations for Generalized Elliptical Anisotropic Velocity Model with PINNs

In this model, there is a transition from a constant elliptical anisotropy to a generalized elliptical anisotropy. This model employs ray-tracing equations for an anisotropic medium with velocities depending elliptically on direction. The details of this explanation can be found in Section 2.4.3. The neural network architecture is similar with the constant elliptical anisotropic case, the input layer receives time, and the output layer predicts x, z, p_x, and p_z. The network consists of 6 hidden layers, each containing 48 neurons, and employs the hyperbolic tangent activation function. The Adam optimizer is used to fine-tune the model parameters with a learning rate of 0.001. Additionally, a learning rate scheduler with a step size of 1000 and a decay factor gamma of 0.9 is implemented. Weight decay is set to 0.1, and the number of training epochs is fixed to $45,000$.

The initial conditions of this model are defined in Equation 2.106 above. In addition to this, the velocity parameters and take-off angle are defined below. The complete list is provided below.

$$\begin{aligned}
\text{initial coordinates}: &\quad x(0) = 0, \quad z(0) = 0, \\
\text{initial momenta}: &\quad p_x(0) = \frac{\sin\vartheta_0}{v_0}, \quad p_z(0) = \frac{\cos\vartheta_0}{v_0}, \\
\text{initial velocity}: &\quad v_0 = \sqrt{c^2\sin^2\vartheta_0 + a^2\cos^2\vartheta_0}, \\
\text{velocity model parameters}: &\quad a = 2, \quad b = 1, \quad c = 2.1, \quad d = 0.85, \\
\text{take}-\text{off angle}: &\quad \vartheta_0 = \frac{\pi}{12}, \\
\text{travel time}: &\quad t_{max} = 5.
\end{aligned}$$

The initial conditions loss functions are defined by utilizing the aforementioned initial conditions, and these functions are written as follows:

$$\mathcal{L}_{IC_x} = \left(x_{0pred} - 0\right)^2, \tag{4.72}$$

$$\mathcal{L}_{IC_z} = \left(z_{0pred} - 0\right)^2, \tag{4.73}$$

$$\mathcal{L}_{IC_{p_x}} = \left(p_{x0pred} - \frac{\sin\vartheta_0}{\sqrt{c^2\sin(\vartheta_0)^2 + a^2\cos(\vartheta_0)^2}}\right)^2, \tag{4.74}$$

$$\mathcal{L}_{IC_{p_z}} = \left(p_{z0pred} - \frac{\cos\vartheta_0}{\sqrt{c^2\sin(\vartheta_0)^2 + a^2\cos(\vartheta_0)^2}}\right)^2. \tag{4.75}$$

The physics loss functions are defined by utilizing the Hamilton's equations of motion, which are derived in Equation 2.102. Recalling them:

$$\dot{x}(t) = (c + dz)^2 p_x, \tag{4.76}$$

$$\dot{z}(t) = (a + bz)^2 p_z, \tag{4.77}$$

$$\dot{p}_x(t) = 0, \tag{4.78}$$

$$\dot{p}_z(t) = -d(c + dz)p_x^2 - b(a + bz)p_z^2. \tag{4.79}$$

Consequently, the physics loss functions are defined as follows:

$$\mathcal{L}_{physics_x} = \sum_{i=1}^{N_{physics}} \left(\dot{x}^i_{autograd} - (c + dz^i_{pred})^2 p_{x\,pred}^{\;i}\right)^2, \tag{4.80}$$

$$\mathcal{L}_{physics_z} = \sum_{i=1}^{N_{physics}} \left(\dot{z}^i_{autograd} - (a + bz^i_{pred})^2 p_{z\,pred}^{\;i}\right)^2, \tag{4.81}$$

$$\mathcal{L}_{physics_{p_x}} = \sum_{i=1}^{N_{physics}} \left(\dot{p}_{x\,autograd}^{\;i} - 0\right)^2, \tag{4.82}$$

$$\mathcal{L}_{physics_{p_z}} = \sum_{i=1}^{N_{physics}} \left(\dot{p}_{z\,autograd}^{\;i} + \left(d(c + dz^i_{pred})^2 (p_{x\,pred}^{\;i})^2 + b(a + bz^i_{pred})(p_{z\,pred}^{\;i})^2\right)\right)^2. \tag{4.83}$$

The Hamiltonian is employed to describe the energy of this system, as illustrated in Equation 2.101. This equation can be rewritten as follows:

$$\mathcal{H} = \frac{(c + dz)^2 p_x^2 + (a + bz)^2 p_z^2 - 1}{2}. \tag{4.84}$$

Finally, the energy loss function is presented:

$$\mathcal{L}_{energy} = \sum_{i=1}^{N_{energy}} \left(\frac{(c + dz^i_{pred})^2 (p_{x\,pred}^{\;i})^2 + (a + bz^i_{pred})^2 (p_{z\,pred}^{\;i})^2 - 1}{2}\right)^2. \tag{4.85}$$

After identifying the individual loss functions, the total loss function is defined as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{physics} + \mathcal{L}_{IC} + 10^5 \cdot \mathcal{L}_{energy}, \tag{4.86}$$

where:

$$\mathcal{L}_{physics} = 10^3 \cdot \mathcal{L}_{physics_x} + 10^3 \cdot \mathcal{L}_{physics_y} + 10^5 \cdot \mathcal{L}_{physics_{p_x}} + 10^5 \cdot \mathcal{L}_{physics_{p_z}}, \tag{4.87}$$

$$\mathcal{L}_{IC} = 10^3 \cdot \mathcal{L}_{IC_x} + 10^3 \cdot \mathcal{L}_{IC_z} + 10^5 \cdot \mathcal{L}_{IC_{p_x}} + 10^5 \cdot \mathcal{L}_{IC_{p_z}}. \tag{4.88}$$

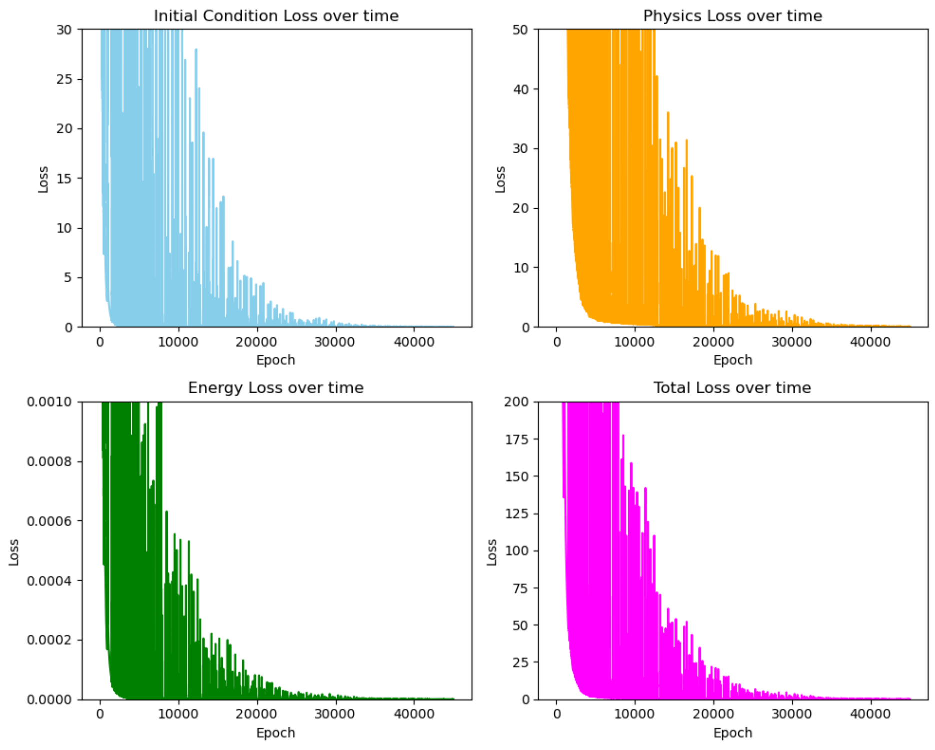

Figure 4.28. The iterative loss results of the model's prediction of ray-path in a linearly heterogenous medium for generalized elliptical anisotropic velocity model with the take-off angle of $\frac{\pi}{12}$ are presented. The initial condition loss over iterations is represented by the sky blue curve, the physics loss by the orange curve, the energy loss by the green curve, and the total loss by the magenta curve. To enhance clarity, the y-axis range has been limited.

The illustration of iterative loss results is presented in Figure 4.28.

To obtain an analytical solution, we employed the equations derived in Equation 2.110, which describe the trajectory of the ray. The objective of this analytical solution is to check the accuracy of the model. In the figures below, the gray curve depicting the analytical solution of the ray path.

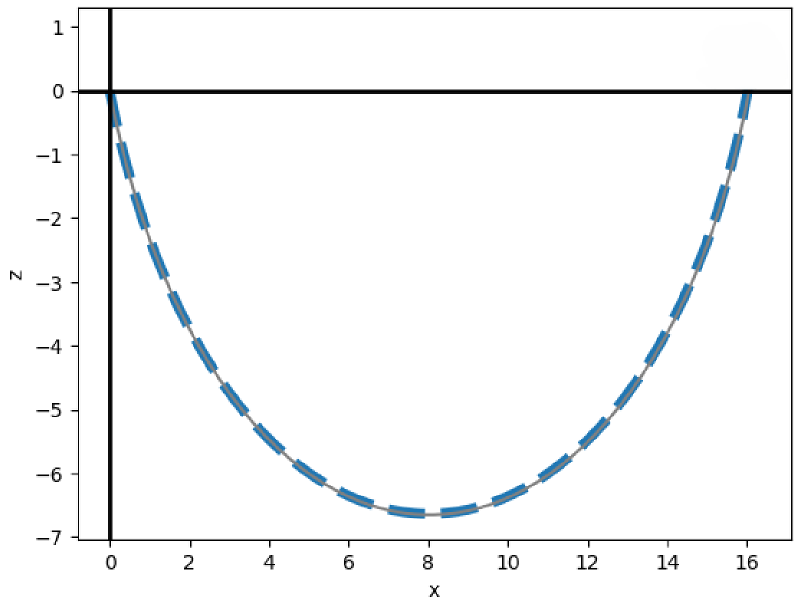

Figure 4.29. The illustration depicts the prediction of the ray path in a linearly heterogeneous medium for a generalized elliptical anisotropic velocity model by the PINN model, considering a take-off angle of $\frac{\pi}{12}$. The blue dashed curve represents the neural network's prediction, and the gray curve depicts the analytical solution of the ray path.

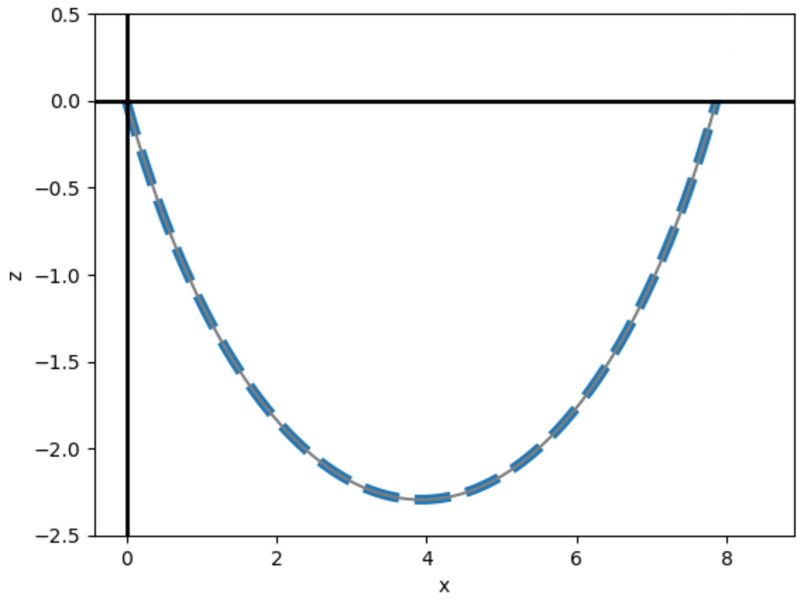

Figure 4.30. The illustration depicts the prediction of the ray path in a linearly heterogeneous medium for a generalized elliptical anisotropic velocity model by the PINN model, considering a take-off angle of $\frac{\pi}{6}$. The blue dashed curve represents the neural network's prediction, and the gray curve depicts the analytical solution of the ray path.

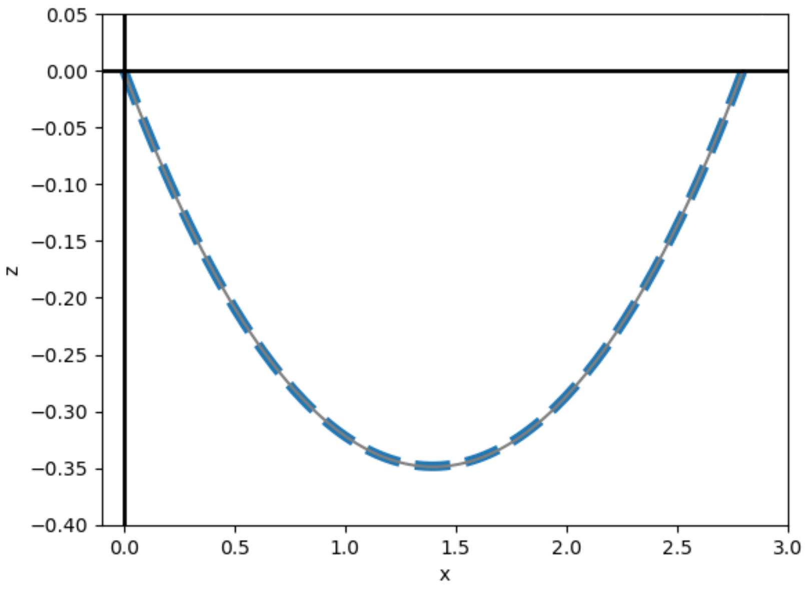

Figure 4.31. The illustration depicts the prediction of the ray path in a linearly heterogeneous medium for a generalized elliptical anisotropic velocity model by the PINN model, considering a take-off angle of $\frac{\pi}{3}$. To achieve a successful conversion for this angle, the maximum time is decreased $t_{max} = 4$. The blue dashed curve represents the neural network's prediction, and the gray curve depicts the analytical solution of the ray path.

5. CONCLUSION

This thesis attempts to improve the methods used to solve ray-tracing equations by introducing an innovative approach using neural networks. Firstly, we compare different types of neural network models, to observe which works best for solving the harmonic oscillator equation. These models include a conventional neural network, a physics-informed neural network, and a model that combines Hamiltonian approach and PINN. Subsequently, we attempt to solve the ray-tracing equations using the model that combines the Hamiltonian approach and PINN. Our goal is to solve these equations with as small dataset as possible. By using initial conditions and minimizing physics and energy losses, we solve ray-tracing equations easier and faster with neural networks.

The first approach uses a conventional neural network to solve the harmonic oscillator equations. In this model, we provide balanced data (i.e., data collected at equal intervals) for approximately half of the problem, while providing no data for the other half. The result of the model indicates that the model is able to learn where balanced data is provided, but is unable to solve the problem where no data is provided. In conclusion, the only practical approach for solving the harmonic oscillator equation with a conventional neural network is to provide balanced data everywhere.

Furthermore, an additional observation was made in this model. The increase in the value of the spring constant, k, in the harmonic oscillator makes it challenging for the network model to solve this problem. Consequently, the value of k is set to 100 in order to guarantee the efficacy of the model.

We then tried to solve the same problem with PINNs by satisfying partial differential equations. We added the final data values x and p obtained for time tmax to the dataset we created for the previous problem. In order to ensure the model's ability to solve correctly, it was necessary to increase the model's capacity, thereby extending its runtime. It was observed that the PINN model was capable of solving the problems that the previous model was able to solve where data is given, and in addition, the PINN model was able to learn the unseen data regions that the previous model was unable to solve. In conclusion, it can be stated that when a partial differential equation and half of the data from the PINN model are provided, the PINN model is capable of solving the problem.

To obtain a solution with a small dataset, we mainly focus on the initial conditions of the system. Our key aim is to ensure the energy of the system remains constant. Thus, we integrate a method called the Hamiltonian approach into the PINN model. This helps us to maintain important physics principles, such as the conservation of energy. When we expand the model and include energy loss considerations, something interesting happens: even if we don't provide any data other than the initial conditions, the model can still find the solution. However, when energy loss is not accounted for in the model, it struggles to learn under the same conditions. This highlights the significance of considering energy conservation when addressing such problems.

After solving the harmonic oscillator equation and comparing the models, we focus on solving the ray-tracing equations in a linearly heterogeneous and both for isotropic and anisotropic media, which is the main focus of this thesis. To solve these equations, we used the Hamiltonian approach in the PINN model. In other words, we used initial condition loss, physics loss, and energy loss. In the forward model, the ray-tracing equations are solved by providing parameters such as velocity, take-off angle and time. Then, the model determines the ray path followed by the seismic waves.

Another observation is that narrowing of the take-off angle in the trace is a complicating factor in the problem. Therefore, its value is set to $\frac{\pi}{12}$ to ensure the efficiency of the model. In the case of larger take-off angles, such as $\frac{\pi}{9}$, $\frac{\pi}{6}$, and $\frac{\pi}{3}$, its efficacy remains unchanged. However, the maximum time should be decreased to achieve optimal performance for the angle of $\frac{\pi}{3}$.

Inverse modeling aims to determine the angle and velocity model by providing the receiver location, earthquake source, and travel time. The network model and the total loss function are similar to those used in the forward method. However, in this case, we introduce another loss function called the "shoot loss". This loss function is used to fit the model to the last data point through loss minimization.

We proceed in a gradual manner to enhance the complexity of the velocity model. Initially, we solve the ray-tracing equations for an isotropic velocity model in a linearly heterogeneous medium. Subsequently, we increase the complexity of the problem with a constant elliptically anisotropic velocity model. Finally, we employ a generalized elliptically anisotropic velocity model in a linearly heterogeneous medium, resulting in a more complex problem.

In this study, the aim was to enhance the solution methods for ray-tracing equations by introducing an innovative approach, facilitating faster and more accurate results. The outcomes obtained highlight the potential contribution to geophysical research and seismic data interpretation. The results demonstrate the efficacy of the PINN method employing the Hamiltonian approach in enhancing the efficiency and accuracy of ray-tracing equation resolution. This efficient network model is initially applied to the isotropic velocity model, and then to a generalized elliptically anisotropic velocity model to increase the complexity of the problem. As a future work, the underground model could be expanded by increasing the number of layers, which would offer promising for further exploration and refinement of the methodology.

REFERENCES

1. Cervenỳ, V., *Seismic ray theory*, Vol. 110, Cambridge university press Cambridge, 2001.

2. McCulloch, W. S. and W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *The bulletin of mathematical biophysics*, Vol. 5, pp. 115–133, 1943.

3. Hornik, K., “Approximation capabilities of multilayer feedforward networks”, *Neural networks*, Vol. 4, No. 2, pp. 251–257, 1991.

4. Mattheakis, M., D. Sondak, A. S. Dogra and P. Protopapas, “Hamiltonian neural networks for solving equations of motion”, *Physical Review E*, Vol. 105, No. 6, p. 065305, 2022.

5. Raissi, M., P. Perdikaris and G. E. Karniadakis, “Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations”, *arXiv preprint arXiv:1711.10561*, 2017.

6. Cuomo, S., V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi and F. Piccialli, “Scientific machine learning through physics–informed neural networks: Where we are and what’s next”, *Journal of Scientific Computing*, Vol. 92, No. 3, p. 88, 2022.

7. Raissi, M., P. Perdikaris and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”, *Journal of Computational physics*, Vol. 378, pp. 686–707, 2019.

8. Blechschmidt, J. and O. G. Ernst, “Three ways to solve partial differential equations with neural networks—A review”, *GAMM-Mitteilungen*, Vol. 44, No. 2, p. e202100006, 2021.

9. Greydanus, S., M. Dzamba and J. Yosinski, "Hamiltonian neural networks", *Advances in neural information processing systems*, Vol. 32, 2019.

10. Baydin, A. G., B. A. Pearlmutter, A. A. Radul and J. M. Siskind, "Automatic differentiation in machine learning: a survey", *Journal of Marchine Learning Research*, Vol. 18, pp. 1–43, 2018.

11. Slawinski, M. A., *Waves and rays in elastic continua*, World Scientific, 2010.

12. Rogister, Y. and M. A. Slawinski, "Analytic solution of ray-tracing equations for a linearly inhomogeneous and elliptically anisotropic velocity model", *Geophysics*, Vol. 70, No. 5, pp. D37–D41, 2005.

13. Charu, C. A., *Neural networks and deep learning: a textbook*, Spinger, 2018.

14. Sanderson, G., "Neural Networks", 2017, `https://www.3blue1brown.com/`, accessed on December 19, 2023.

15. Sanderson, G., "Understanding Activation Functions in Neural Networks", 2017, `https://medium.com/`, accessed on December 20, 2023.

16. Ng, A., "Neural Networks and Deep Learning", 2019, `https://www.coursera.org/`, accessed on January 20, 2024.

17. V, A. S., "What are Neural Networks?", 2017, `https://www.datacamp.com/`, accessed on December 19, 2023.

18. Prince, S. J., *Understanding Deep Learning*, MIT Press, 2023, `http://udlbook.com`.

19. Brownlee, J., "What is the difference between test and validation datasets", *Machine Learning Mastery*, Vol. 14, 2017.

20. Bottou, L., “Stochastic gradient descent tricks”, *Neural Networks: Tricks of the Trade: Second Edition*, pp. 421–436, Springer, 2012.

21. Duchi, J., E. Hazan and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.”, *Journal of machine learning research*, Vol. 12, No. 7, 2011.

22. Hinton, G., “Neural networks for machine learning”, , 2012, `https://www.youtube.com/watch?v=defQQqkXEfE&list=PLoRl3Ht4JOcdU872GhiYWf6jwrk_SNhz9&index=29`.

23. Kingma, D. P. and J. Ba, “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, 2014.

24. Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library”, *Advances in neural information processing systems*, Vol. 32, 2019.

25. Strauss, W. A., *Partial differential equations: An introduction*, John Wiley & Sons, 2007.

APPENDIX A: Further Mathematical Calculations

A.1. Further Mathematical Details on Wave Equations for Isotropic and Homogeneous Medium

According to the stress-strain equation, it is known that [11]:

$$\sigma_{ij} = \lambda\delta_{ij}\sum_{k=1}^{3}\varepsilon_{kk} + 2\mu\varepsilon_{ij}, \tag{A.1}$$

where ε represents the strain tensor, describing the deformation of a material under stress σ, and δ represents the Kronecker delta which is the identity matrix. Here, λ and μ respectively represent the Lamé parameters controlling the material's response to volumetric changes and shear deformation. Substituting the stress-strain equation, Equation A.1, into Cauchy's equation of motion, we obtain:

$$\begin{aligned}\rho\frac{\partial^2 u_i}{\partial t^2} &= \sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\lambda\delta_{ij}\sum_{k=1}^{3}\varepsilon_{kk} + 2\mu\varepsilon_{ij}\right)\\ &= \sum_{k=1}^{3}\left(\delta_{ij}\lambda\sum_{k=1}^{3}\frac{\partial\varepsilon_{kk}}{\partial x_j} + 2\mu\frac{\partial\varepsilon_{ij}}{\partial x_j}\right).\end{aligned} \tag{A.2}$$

Utilizing the equations for strain terms $\varepsilon_{kk} = \frac{\partial u_k}{\partial x_k}$ and $\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$, and substituting them into the above equation:

$$\rho\frac{\partial^2 u_i}{\partial t^2} = \sum_{k=1}^{3}\left[\lambda\delta_{ij}\sum_{k=1}^{3}\frac{\partial}{\partial x_j}\left(\frac{\partial u_k}{\partial x_k}\right) + \mu\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]. \tag{A.3}$$

It is worth remembering the properties of the Kronecker delta.

$$\delta_{ij} = \begin{cases}1, & \text{if } i = j\\ 0, & \text{if } i \neq j\end{cases} \tag{A.4}$$

According to the property of Kronecker's delta, we obtain:

$$\delta_{ij}\sum_{k=1}^{3}\frac{\partial}{\partial x_j}\left(\frac{\partial u_k}{\partial x_k}\right) = \begin{cases}\frac{\partial}{\partial x_i}\left(\frac{\partial u_k}{\partial x_k}\right), & \text{if } i = j\\ 0, & \text{if } i \neq j\end{cases} \tag{A.5}$$

In Equation A.3, both cases ($i = j$ and $i \neq j$) are included. When i is not equal to j, the result is zero, so this case has no effect on the summation. Since the Kronecker's delta is nonzero when i is equal to j, Equation A.3 can be simplified to:

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \lambda \sum_{k=1}^{3} \frac{\partial}{\partial x_j}(\frac{\partial u_k}{\partial x_k}) + \mu \sum_{j=1}^{3} \frac{\partial}{\partial x_j}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}). \tag{A.6}$$

Applying linearity to the given equations, we can rewrite them as:

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \lambda \sum_{j=1}^{3} \frac{\partial^2 u_j}{\partial x_i \partial x_j} + \mu \sum_{j=1}^{3} \frac{\partial^2 u_i}{\partial x_j^2} + \mu \sum_{j=1}^{3} \frac{\partial^2 u_j}{\partial x_j \partial x_i}. \tag{A.7}$$

Recollecting the terms:

$$\begin{aligned} \rho \frac{\partial^2 u_i}{\partial t^2} &= (\lambda + \mu) \sum_{j=1}^{3} \frac{\partial^2 u_j}{\partial x_i \partial x_j} + \mu \sum_{j=1}^{3} \frac{\partial^2 u_i}{\partial x_j^2} \\ &= (\lambda + \mu) \frac{\partial}{\partial x_j} \sum_{j=1}^{3} \frac{\partial u_j}{\partial x_j} + \mu \left(\sum_{j=1}^{3} \frac{\partial^2}{\partial x_j^2} \right) u_i. \end{aligned} \tag{A.8}$$

A.2. Further Mathematical Details on Derivation of Eikonal Equation

The next step is the derivation of the eikonal equation for an anisotropic and inhomogeneous continuum. This derivation is similar to the derivation of the eikonal equation for an isotropic homogeneous continuum. We begin with a reminder of the wave equation for the anisotropic inhomogeneous continuum:

$$\rho(x) \frac{\partial^2 u_i}{\partial t^2} = \frac{1}{2} \sum_{j=1}^{3} \sum_{k=1}^{3} \sum_{l=1}^{3} \left[\frac{\partial c_{ijkl}(x)}{\partial x_j} \left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k} \right) + c_{ijkl}(x) \left(\frac{\partial^2 u_k}{\partial x_j \partial x_l} + \frac{\partial^2 u_l}{\partial x_j \partial x_k} \right) \right]. \tag{A.9}$$

The process of switching to the frequency domain involves the application of the Fourier transform to Equation A.9. Subsequently, the Fourier transformed wave equation becomes:

$$\rho(x)(i\omega)^2 \tilde{u}(x,\omega) = \frac{1}{2} \sum_{j=1}^{3} \sum_{k=1}^{3} \sum_{l=1}^{3} \left[\frac{\partial c_{ijkl}(x)}{\partial x_j} \left(\frac{\partial \tilde{u}_k}{\partial x_l} + \frac{\partial \tilde{u}_l}{\partial x_k} \right) + c_{ijkl}(x) \left(\frac{\partial^2 \tilde{u}_k}{\partial x_j \partial x_l} + \frac{\partial^2 \tilde{u}_l}{\partial x_j \partial x_k} \right) \right], \tag{A.10}$$

where $\tilde{u}(x,\omega)$ denotes the Fourier transform of the displacement field $\mathbf{u}(x,t)$. Substituting the trial solution from Equation 2.43 into Equation A.10 above gives:

$$\begin{aligned}
-\rho(x)\omega^2 A_i e^{i\omega\psi(x)} &= \frac{1}{2}e^{i\omega\psi(x)}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}\Bigg[\frac{\partial c_{ijkl}(x)}{\partial x_j}\left(\frac{\partial A_k}{\partial x_l}+\frac{\partial A_l}{\partial x_k}+i\omega\Big(\frac{\partial\psi}{\partial x_l}A_k+\frac{\partial\psi}{\partial x_k}A_l\Big)\right)\\
&+ c_{ijkl}(x)\Bigg(\Big(\frac{\partial^2 A_k}{\partial x_j\partial x_l}+\frac{\partial^2 A_l}{\partial x_j\partial x_k}\Big)+i\omega\Big(\frac{\partial^2\psi}{\partial x_j\partial x_l}A_k+\frac{\partial\psi}{\partial x_j}\frac{\partial A_l}{\partial x_k}+\frac{\partial^2\psi}{\partial x_j\partial x_k}A_l\\
&+ \frac{\partial\psi}{\partial x_l}\frac{\partial A_k}{\partial x_j}+\frac{\partial\psi}{\partial x_k}\frac{\partial A_l}{\partial x_j}+\frac{\partial\psi}{\partial x_j}\frac{\partial A_l}{\partial x_k}\Big)-\omega^2\left(\frac{\partial\psi}{\partial x_l}\frac{\partial\psi}{\partial x_j}A_k+\frac{\partial\psi}{\partial x_k}\frac{\partial\psi}{\partial x_j}A_l\right)\Bigg)\Bigg].
\end{aligned} \tag{A.11}$$

The term $e^{i\omega\psi(x)}$ is eliminated since it is non-zero, and the equation is then divided by ω^2. Following these manipulations have been carried out and the real part of the resulting expression has been considered, the following equation is obtained:

$$\begin{aligned}
-\rho(x)A_i = \frac{1}{2}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}\Bigg[&\frac{1}{\omega^2}\frac{\partial c_{ijkl}(x)}{\partial x_j}\left(\frac{\partial A_k}{\partial x_l}+\frac{\partial A_l}{\partial x_k}\right)+\frac{c_{ijkl}(x)}{\omega^2}\left(\frac{\partial^2 A_k}{\partial x_j\partial x_l}+\frac{\partial^2 A_l}{\partial x_j\partial x_k}\right)\\
&-c_{ijkl}(x)\left(\frac{\partial\psi}{\partial x_l}\frac{\partial\psi}{\partial x_j}A_k+\frac{\partial\psi}{\partial x_k}\frac{\partial\psi}{\partial x_j}A_l\right)\Bigg].
\end{aligned} \tag{A.12}$$

In the limit as the frequency tends to infinity, the equation simplifies to:

$$-\rho(x)A_i = -\frac{1}{2}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}c_{ijkl}(x)\left(\frac{\partial\psi}{\partial x_l}\frac{\partial\psi}{\partial x_j}A_k+\frac{\partial\psi}{\partial x_k}\frac{\partial\psi}{\partial x_j}A_l\right). \tag{A.13}$$

The symmetry of the elasticity tensor c_{ijkl} with respect to k and l allows the equation to be simplified as follows:

$$\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}c_{ijkl}(x)\left(\frac{\partial\psi}{\partial x_l}\frac{\partial\psi}{\partial x_j}A_k\right)-\rho(x)A_i=0. \tag{A.14}$$

Rephrasing the equation as:

$$\sum_{k=1}^{3}\left(\sum_{j=1}^{3}\sum_{l=1}^{3}c_{ijkl}(x)\frac{\partial\psi}{\partial x_j}\frac{\partial\psi}{\partial x_l}-\rho(x)\delta_{ik}\right)A_k(x)=0, \tag{A.15}$$

The definition of the phase-slowness vector, $p_j := \dfrac{\partial\psi}{\partial x_j}$, can be utilized to derive the following result:

$$\sum_{k=1}^{3}\left(\sum_{j=1}^{3}\sum_{l=1}^{3}c_{ijkl}(x)p_jp_l-\rho(x)\delta_{ik}\right)A_k(x)=0. \tag{A.16}$$

This equation is known as Christoffel's equation. The characteristic equation associated with Christoffel's equations, where the determinant must be equal to zero to find the eigenvalues, represented as λ, of the system. The characteristic equation corresponding to Christoffel's equations is as follows:

$$det\left[\sum_{j=1}^{3}\sum_{l=1}^{3} c_{ijkl}(x)p_jp_l - \rho(x)\delta_{ik})\right] = 0. \tag{A.17}$$

The eigenvalues of Christoffel's equations are related to the velocity of the wavefront, and the eigenvectors are correspond to the directions of the displacement.

Indeed, $p^2 = 0$ would imply that the slowness of the wavefront is zero, resulting in an infinite velocity, which is a nonphysical situation [11]. Therefore, assuming $p^2 \neq 0$ ensures that the velocity of the wavefront remains finite and physically realistic, and we can factor out p^2 from the determinant as follows:

$$(p^2)^3 det\left[\sum_{j=1}^{3}\sum_{l=1}^{3} c_{ijkl}(x)\frac{p_jp_l}{p^2} - \frac{\rho(x)}{p^2}\delta_{ik})\right] = 0. \tag{A.18}$$

By factorizing the equation into three parts, we reveal three distinct eigenvalues v_1, v_2, v_3 corresponding to different wave velocities in the medium:

$$\left[p^2 - \frac{1}{v_1^2(\mathbf{x,p})}\right]\left[p^2 - \frac{1}{v_2^2(\mathbf{x,p})}\right]\left[p^2 - \frac{1}{v_3^2(\mathbf{x,p})}\right] = 0. \tag{A.19}$$

For all phases, we have the eikonal equation:

$$p^2 = \frac{1}{v^2(\mathbf{x,p})} \tag{A.20}$$

Replacing the phase-slowness vector with the eikonal function, we obtain:

$$[\nabla\psi(x)]^2 = \frac{1}{v^2(\mathbf{x,p})}. \tag{A.21}$$

This expression gives rise to the eikonal equation, which describes the magnitude of phase slowness based on the properties of an anisotropic inhomogeneous medium. The propagation of the wavefront through such a continuum is explained by this equation. Expressing the gradient $\nabla\psi$ in terms of q_i and p_i:

$$\nabla\psi = \left(\frac{\partial\psi}{\partial x_1}, \frac{\partial\psi}{\partial x_2}, \frac{\partial\psi}{\partial x_3}\right) = (p_1, p_2, p_3). \tag{A.22}$$

This gradient provides information about both the direction and the rate of change of the wave's phase.