COMPUTATIONAL METHODS FOR ONE-DIMENSIONAL SCATTERING IN NON-SMOOTH MEDIA

CHIEN-CHENG CHIU

A THESIS SUBMITTED TO

THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF ARTS

GRADUATE PROGRAM IN MATHEMATICS

YORK UNIVERSITY

TORONTO, ONTARIO

SEPTEMBER 2021

Abstract

CHIEN-CHENG CHIU

COMPUTATIONAL METHODS FOR ONE-DIMENSIONAL
SCATTERING IN NON-SMOOTH MEDIA

This thesis implements various numerical algorithms used for acoustic imaging of layered media: the scattering-based algorithm, the classical Born approximation, the refined impedance transform and the echoes-to-impedance transform. The last three are inverse scattering algorithms that numerically convert data in the time domain to an impedance in the spatial domain; however, the simplicity, speed and accuracy can differ greatly among them. In place of physical recordings, the scattering-based algorithm is used to generate accurate synthetic data. Numerical experiments and error analyses reveal significant differences among the three. The principal conclusion is that the method based on the most sophisticated mathematical ideas, namely the echoes-to-impedance transform, is far superior.

Dedication

To my professors, my friends and my family, especially my parents

Acknowledgements

I would like to acknowledge Dr. Peter Gibson, my supervisor and my professor in Partial Differential Equations. With his knowledge, resources and support, I finally achieved something that I believed I could not do. I would also like to acknowledge all my other Physics and Mathematics professors throughout my years of education. They provided me with all the knowledge I needed to complete this thesis, and much more in many other fields of Mathematics. Finally, I would also like to acknowledge my parents for providing me the financial support needed to reach this far and even further.

Contents

List of Tables

List of Figures

1 Introduction

1.1 One Problem, Many Solutions

There may be many numerical methods to solve a given mathematical problem when analytical methods are time-consuming or impossible; however, by comparing the accuracy and efficiency among the numerical methods, usually only one of them gives the best results. As a simple example, the bisection method, the fixed-point iteration and the Newton's method are some numerical schemes in a root-finding problem, but it is considered that the Newton's method is the most powerful as its accuracy and efficiency is greater than the other two methods. A similar situation appears when working with an important second-order hyperbolic differential equation applied to perform acoustic imaging of layered media, the one-dimensional wave equation,

$$u_{tt} - \frac{1}{\zeta}\left(\zeta u_x\right)_x = 0 \tag{1.1a}$$

$$u(x,0) = f(x), \qquad u_t(x,0) = g(x) \tag{1.1b}$$

where we assume

1. ζ is the acoustic impedance of the layered medium in terms of position x with the following restrictions:

- $\zeta > 0$ must be constant within the intervals, $(-\infty, x_l)$ and (x_u, ∞) such that $x_l < x_u$.

2. u is the particle velocity function in terms of position x and time t.

Suppose that $\zeta(x_l) = \zeta_l$ and $\zeta(x_u) = \zeta_u$ are the constants within the intervals, $(-\infty, x_l)$, and (x_u, ∞), respectively. One important aspect about (1.1) is the position x is in travel-time coordinates, and not in spatial coordinates.

1.2 The Mathematics behind Acoustic Imaging

1.2.1 From Three- to One-Dimensional Acoustic Wave Equation

In reality, the acoustic wave equation for a three-dimensional medium is

$$\frac{1}{c^2}u_{tt} - \rho\left[\nabla \cdot \left(\frac{1}{\rho}\nabla u\right)\right] = 0 \tag{1.2a}$$

$$u(\mathbf{x}, 0) = f(\mathbf{x}), \qquad u_t(\mathbf{x}, 0) = g(\mathbf{x}) \tag{1.2b}$$

where

1. $\rho : \mathbb{R}^3 \to \mathbb{R}^+$ is the density.
2. $c = \sqrt{\dfrac{K}{\rho}}$ is the wave speed and $K : \mathbb{R}^3 \to \mathbb{R}^+$ is the bulk modulus.
3. $\mathbf{x} \in \mathbb{R}^3$.

However, it is possible to reduce (1.2) to (1.1) with $\zeta = \sqrt{K\rho}$ for acoustic imaging for layered media because the symmetry of a physical structure with multiple laminated layers

allows the wave speed and the bulk modulus to only vary in the x-direction with respect to spatial coordinates as the planar acoustic wave travels through the layered medium.

An important example occurring in nature is that of sedimentary bedrock, which often has horizontally layered structure where each layer may correspond to a different geological formation. If the source sound wave is planar, and aligned with the layers in the medium, the problem reduces from three-dimensional to one by symmetry. Moreover, a spherically expanding three-dimensional wave becomes approximately planar at long distances. Thus, planar sound waves correspond to certain realistic physical situations.

1.2.2 Mathematical Framework

Mathematically, acoustic imaging consists of two parts:

1. computing synthetic measured data
2. computing acoustic impedance from measured data

The former entails solving the wave equation at a particular point x_0 corresponding to the location of the source/receiver. The scattering-based algorithm in [11, p.2], which differs from widely used numerical methods such as finite differences, can be used in the context of a distributional source wavelet such as the Dirac function, and/or with a discontinuous impedance. The second part involves inverse scattering algorithms such as

1. the Born approximation in [15, p.4], [23], [24].
2. the refined impedance transform in [15, p.3]
3. the echoes-to-impedance transform in [12, p.5]

The purpose of this thesis is to describe the above algorithms, implement them in MATLAB, and compare their efficacy on a range of test media and source wavelets.

1.2.3 Scattering Theory

In the context of acoustic imaging, the forward problem, or direct scattering, resembles the propagation of an acoustic source wave through a medium with an acoustic impedance as a function of a spatial variable as governed by the wave equation. This generates a wave field, or the many solutions to the wave equation, in which one of the solutions at a specific spatial location for a time duration is enough for acoustic imaging. The inverse problem, or inverse scattering, corresponds to determination of the acoustic impedance as a function of the spatial variable from recorded data, which consists of the solution to the wave equation at a specific spatial location for a finite time duration. Inverse scattering is almost the opposite of direct scattering except that it does not need all the information about the wave propagation happening through a medium to determine the acoustic impedance.

There are well-established numerical algorithms applicable to the forward and inverse scattering problems. However, the inverse scattering problem has until recently never been fully solved theoretically. Moreover, established algorithms to compute forward scattering break down in the presence of singularities; these may include singularity in the source wavelet such as when it is a Dirac distribution, or in the medium such as when the acoustic impedance function has discontinuities. A major contribution of this thesis is to implement algorithms that overcome these shortcomings.

1.2.4 Important Components of the Algorithms

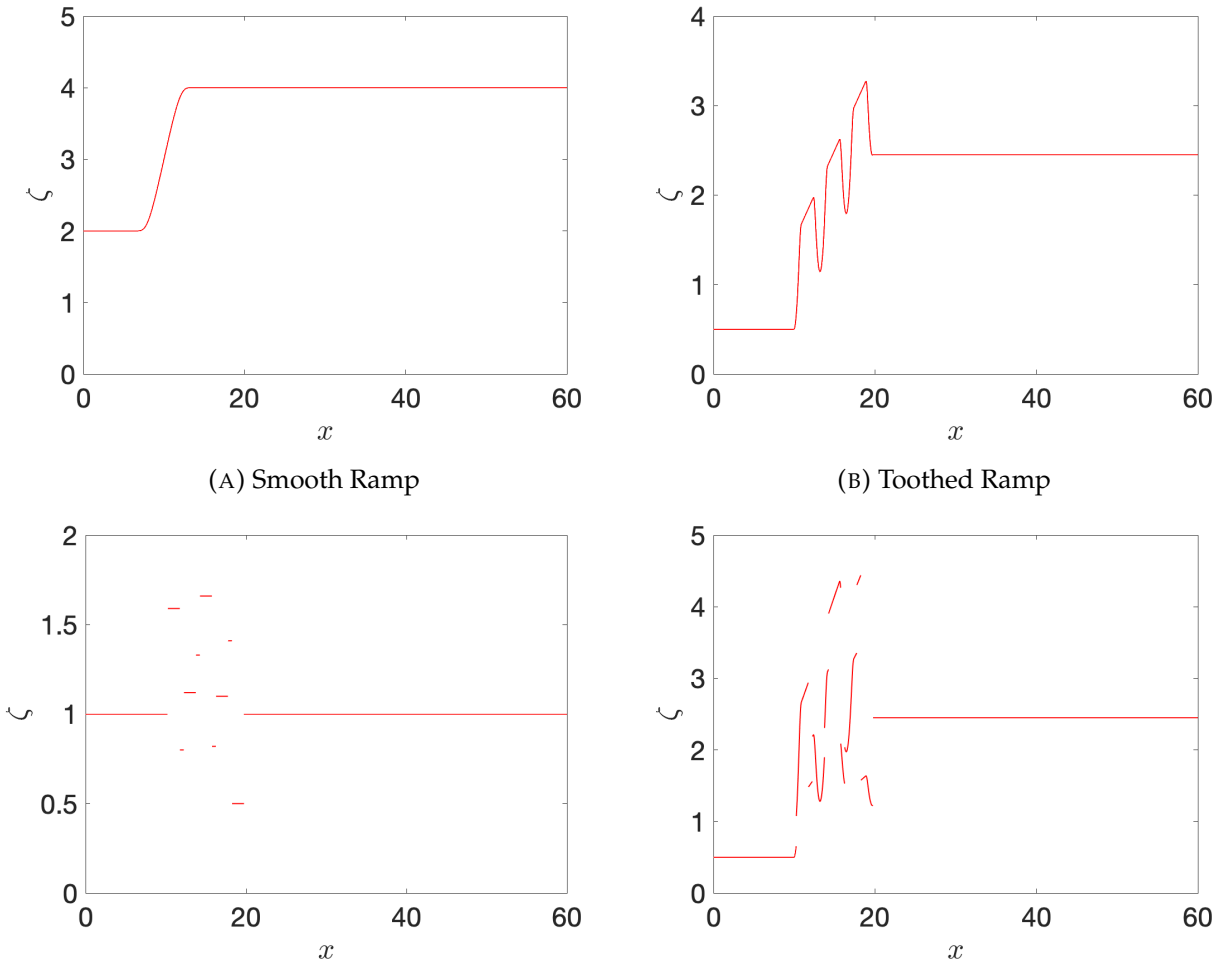
There are two key objects that affect the performance of the applied numerical schemes:

1. the impedance, ζ
2. the source wavelet

In addition to $\zeta > 0$, while ζ must behave constantly outside (x_l, x_u), it does not need to behave the same way inside (x_l, x_u) and so the complexity of ζ can vary hugely. ζ can be highly oscillating, contain multiple discontinuities, or a mixture of both in (x_l, x_u). As ζ becomes more complicated, it is necessary to modify the spatial grid to acquire more accurate results.

Some examples of ζ's are in Fig. 1.1. Fig. 1.1a has the same function as the smooth ramp function on [12, p.21], but translates x to the right by 5. Fig. 1.1b has the same function as the toothed ramp function on [12, p.21]. Fig. 1.1c has jump points at: $x = [10.25, 11.75, 12.25, 13.75, 14.25, 15.75, 16.25, 17.75, 18.25, 19.75]$ with the following values: $y = [1, 1.59, 0.8, 1.12, 1.33, 1.66, 0.82, 1.1, 1.41, 0.5, 1]$. Fig. 1.1d has the function that is the product of the toothed ramp function and the random steps function. These are ζ's used in the numerical experiments and error analyses.

Secondly, the source wavelet can be a regular or singular source. As for a singular source wavelet, it is crucial to modify the numerical schemes to achieve proper results. Thus, it is essential to be familiar with both ζ and the source wavelet both of which have an important bearing on numerical schemes.



(A) Smooth Ramp

(B) Toothed Ramp

(C) Random Steps

(D) Discontinuous Toothed Ramp

FIGURE 1.1: Examples of ζ

1.3 Methodology

1.3.1 The Scattering-Based Algorithm

The scattering-based algorithm applies the theoretical approach of approximating ζ as a step function and solving (1.1) exactly. Unlike some methods that use triangular matrices, the scattering-based algorithm works with a pentadiagonal matrix with $4n$ non-zero

elements where n is the number of spatial grid points. Its simplicity reduces the number of iterations, resulting in quicker computation on any spatial grid than past methods. Furthermore, Theorems 1, 2, and 3 in [11] prove that the algorithm can approximate the solution to (1.1) even when dealing with discontinuous ζ's or singular source wavelets. Though the algorithm can also show the interactions between a right- and left-moving source wavelet transmitting through ζ from opposite boundaries, the right-moving source wavelet is enough for the study of acoustic imaging.

Some important considerations in implementing the scattering-based algorithm are:

1. If ζ has rapid oscillations, then the spatial grid needs to be correspondingly dense.
2. If ζ contains discontinuities, then it is important to adjust the spatial grid such that each discontinuity lies at a midpoint between two consecutive spatial grid points.

1.3.2 The Classical Born Approximation

The classical Born approximation remains as a leading one-dimensional method in contemporary seismic imaging despite its serious flaws because its evaluation is quick in a simple manner and is fairly accurate for certain ζ's. One of its weaknesses is approximating the solution with only single scattering, which leads to inaccuracies. Moreover, the method requires pre-processing the data by deconvolving the source wavelet, which is highly problematic in practice in [15, p.2-3].

1.3.3 The Refined Impedance Transform

The refined impedance transform is similar to the Born approximation in how it utilizes the data albeit using a different formula. It overcomes one of the weaknesses of the single

scattering approximation, and works without the necessity of source wavelet deconvolution in [15, p.3]. It can operate on a wide scale of ζ's and recover an approximated impedance, $\tilde{\zeta}$, at the far side of an inhomogeneity in [15, p.3]. It matches or exceeds the Born approximation in terms of speed, accuracy and efficiency.

1.3.4 The Echoes-to-Impedance Transform

Unlike the Born approximation and the refined impedance transform published in 2018, the echoes-to-impedance transform operates on a completely different level that lets it achieve the most accurate results easily and quickly. As stated in [12, p.2], it can handle an arbitrary ζ and work with non-preprocessed data. Mysteriously, it seems to recover $\tilde{\zeta}$ at higher resolution than the width of the source wavelet. Finally, even though the measured data and parameters are real-valued in the acoustic context, the algorithm's ability to compute with complex values widens its range of potential applications to other physical contexts in [12, p.2].

1.4 Outline

Chapter 2 presents the physics behind acoustic imaging of layered media. It provides more details on the term impedance and shows that there are different types of impedance. It also informs how the speed of sound can vary in different media and pinpoints that there is a difference between two types of velocities that sound can have: particle velocity and acoustic velocity. There is also a description of how acoustic imaging of a general medium works and indicates how it can simplify for a layered medium. It discusses the wave equation in terms of the wave speed instead of impedance that is commonly seen

in physics and provides an analytical method that can find an approximate solution to the wave equation. Lastly, it gives some examples of acoustic imaging in different areas of research.

Chapter 3 contains information about wave propagation, which is important to understand before applying any of the computational methods. It gives more details on the behaviour of the source wavelet as it partially reflects and transmits through any ζ and generates a wave field. §3.3 explores why and what modifications to the numerical schemes are critical, especially to the scattering-based algorithm, for singular source wavelets. It elucidates a theorem concerning the scattering-based algorithm that explains some subtleties involved in accurately plotting the output.

Chapter 4 provides information about the three different inverse scattering algorithms. It discusses more on how each computes with the recorded data, which may contain noise. It expresses more about the efficiency and accuracy of each inverse scattering algorithm, which provides support for the comparison among the inverse scattering algorithms. §4.4 also introduces the idea of data downsampling to return $\tilde{\zeta}$.

Chapter 5 shows the numerical experiments and error analysis with the computational methods. Working with various ζ's, it provides illustrations on wave propagation at different times and positions when using the variations of the scattering-based algorithm. Operating with the same recorded data with or without noise generated by the scattering-based algorithm, each inverse scattering algorithm provides a plot of $\tilde{\zeta}$ to match ζ. In addition, it includes figures of $\tilde{\zeta}$ when data downsampling is applied to the echoes-to-impedance transform. With these figures and some calculations on relative errors, it yields evidence that the echoes-to-impedance transform is a powerful tool for acoustic

imaging in layered media.

Chapter 6 provides a summary of the thesis. It reviews the purpose of the thesis and talks about the evidence shown to achieve the goals. There is a discussion about the limitations of the computational methods for one-dimensional scattering in non-smooth media. Lastly, it also acknowledges how the methodologies can contribute to the present era of acoustic imaging of inhomogeneous layered media.

Appendices A.1 to D.1 contains MATLAB implementations of the computational methods used in the numerical experiments, which follow the order in §1.2.2. Furthermore, the versions of the code for the scattering-based algorithm can show animations of the source wavelet transmitting through ζ to display the behavioural changes that contributes to the wave field. Appendix E.1 shows a proof of a formula used to determine the amplitude of an impulse supported at a certain point. Appendix F.1 introduces a MATLAB `myplot` function to avoid the issue of displaying the vertical lines at the discontinuities in ζ when using `plot` in MATLAB. Finally, Appendix G.1 shows an estimated computational time equation in terms of the number of spatial grid points for the echoes-to-impendance transform.

2 Physical Background of Acoustic Imaging

2.1 Acoustic Waves

As the name implies, acoustic imaging uses acoustic, or simply known as sound, waves that can have varying speeds at different points in a given medium. In the case of layered media, the speed of sound only varies in the direction normal to the layers, say in the z-direction. The formula to compute the speed of sound in a layered medium is

$$c = \sqrt{\frac{K}{\rho}} \tag{2.1}$$

where c, K and ρ are the sound speed, the bulk modulus and the density of the material in [10]. However, it is possible to use a change of variable to transform the spatial coordinates into travel-time coordinates in which the wave speed becomes constant relative to the new coordinates in [11].

2.2 Acoustic Impedance

In terms of acoustic imaging, acoustic impedance is defined to be the square root of the product of the bulk modulus and density:

$$\zeta = \sqrt{K\rho}, \tag{2.2}$$

Thus, acoustic impedance is purely real because the bulk modulus and the density of a medium are always real. However, in other areas of research, impedance can take on complex values. An example is the impedance of a capacitor in a mechanical oscillator with damping. In such an electrical system, it is better to define the capacitor's impedance by a single complex value using complex exponentials to express the voltage and current in [10].

2.3 The Physical Derivation of (1.1a)

Some Physics textbooks describe the wave equation for the wave propagation in inhomogeneous layered media in terms of $c(z)$, or the varying wave speed due to the varying properties of the structure. In this model, the wave equation in spatial coordinates is

$$u_{tt} - c^2 u_{zz} = 0. \tag{2.3}$$

Then, setting

$$\zeta = \frac{1}{c}$$

and using the change of variable

$$x = \int_0^z \frac{1}{c(s)}\, ds, \tag{2.4}$$

(2.3) converts to (1.1a) in travel-time coordinates, x [11]. However, this is the incorrect physical derivation of (1.1a) because $\zeta \dfrac{1}{c} \neq \sqrt{K\rho}$.

The true physical derivation of (1.1a) comes from a coupled system in terms of ρ and K.

$$\begin{aligned} 0 &= \rho u_t + P_z \\ 0 &= \frac{1}{K} P_t + u_z \end{aligned}$$

where P is the acoustic pressure and u is the particle velocity. Then

$$\begin{aligned} P_t &= -K u_z \qquad \text{and} \qquad P_z = -\rho u_t \\ P_{zt} &= -\left(K u_z\right)_z \qquad \text{and} \qquad P_{zt} = -\rho u_{tt} \end{aligned}$$

Thus,

$$\begin{aligned} -\rho u_{tt} &= -\left(K u_z\right)_z \\ 0 &= \rho u_{tt} - \left(K u_z\right)_z \end{aligned}$$

By using the same change of variable,

$$x = \int_0^z \frac{1}{c(s)}\, ds,$$

the previous equation becomes

$$\begin{aligned} 0 &= u_{tt} - \frac{1}{\rho}\left(K u_z\right)_z \\ &= u_{tt} - \frac{1}{\rho}\sqrt{\frac{\rho}{K}}\left(K\sqrt{\frac{\rho}{K}}u_x\right)_x \\ &= u_{tt} - \frac{1}{\sqrt{K\rho}}\left(\sqrt{K\rho}u_x\right)_x \end{aligned}$$

Finally, by using the true definition of acoustic impedance (2.2),

$$0 = u_{tt} - \frac{1}{\zeta}\left(\zeta u_x\right)_x .$$

2.4 The Wentzel-Kramers-Brillouin Approximation

Though (1.1) is not commonly seen in Physics textbooks, there are some well-known methods to solve (2.3), one of which is the Wentzel-Kramers-Brillouin (WKB) approximation. It assumes the solution to (2.3) takes on the form

$$u(z,t) = A(z)\cos\left[S(z) - \omega t\right] \tag{2.5}$$

where A, ω are the wave's amplitude and frequency, respectively. By substituting (2.5) into (2.3), it is possible to find the formulas for $A(z)$ and $S(z)$ using the derived coupled differential equations

$$A_{zz} + \left[\left(\frac{\omega}{c}\right)^2 - S_z^2\right]A = 0 \tag{2.6a}$$

$$2S_z A_z + S_{zz}A = 0 \tag{2.6b}$$

The WKB approximation is

$$S(z) \approx \int_{z_0}^{z} \frac{\omega}{c(s)}\, ds \tag{2.7a}$$

$$A(z) = A(z_0)\sqrt{\frac{c(z)}{c(z_0)}} \tag{2.7b}$$

where z_0 is some reference position that the wave has some certain amplitude. Thus,

$$u(z,t) \approx A(z_0)\sqrt{\frac{c(z)}{c(z_0)}}\cos\left[\int_{z_0}^{z} \frac{\omega}{c(s)}\, ds - \omega t\right] \tag{2.8}$$

in [6]. It should be emphasized, however, that this is only an approximation.

2.5 Procedure

Acoustic imaging produces images of internal formation of an object that may be opaque to light by analyzing reflected acoustic echoes. By sending a sound wave through a medium, the sound wave partially reflects and transmits at each point of the medium. The acoustic impedance determines how much a wave partially reflects and transmits at each point, also known as reflectivities and transmittivities. Recording the reflections of the sound wave with a sensor collects information about the material's density and can help recover the acoustic impedance. It is essential for the analyst to record a period of time that is at least twice the depth of the object to be imaged in [12, p.5]. Acoustic imaging aims to determine hidden structure that is otherwise obscured from observation. One special case of acoustic imaging occurs when the medium has a layered structure, also known as acoustic imaging of layered media, which can simplify modelling the internal structures of the unique medium.

2.6 Applications

2.6.1 A Geophysical Example

The oil and gas industry uses acoustic imaging for the analysis of the complex structure of hydrocarbon reservoirs. This involves using seismic surveys to make economic decisions. Seismic waves, which have frequencies from 3 to 100 Hz, can help depict the hydrocarbon reservoirs beneath the earth surface in [30]. Other examples of acoustic imaging in geophysics can be found in [1], [3], [17], [27], [28].

2.6.2 An Engineering Example

Aeronautical engineering applies acoustic imaging as a nondestructive testing method to check on the materials, parts, components and assemblies of airplanes and rockets without damage. Ultrasonic C-scan can detect the conditions or defects in the fillet size, shear tie void, core to face holes, and braze puddlings in its structure. Sonic test can evaluate the gross core to face gaps due to severe titanium aluminium formation in an aircraft having aluminium-blazed titanium honeycomb structure. Finally, a combination of ultrasonic attenuation, eddy-current conductivity and thermoelectric can determine the titanium-hydride concentrations in brazed alpha alloys in [16]. Other examples of acoustic imaging in the field of engineering can be found in [2], [5], [22], [26].

2.6.3 A Medical Example

A common medical imaging method for breast cancer detection is x-ray mammography [21]; however, there still are less expensive, safer and/or more effective alternative imaging methods for patients. Though x-ray mammography provides high-quality images, it has some drawbacks:

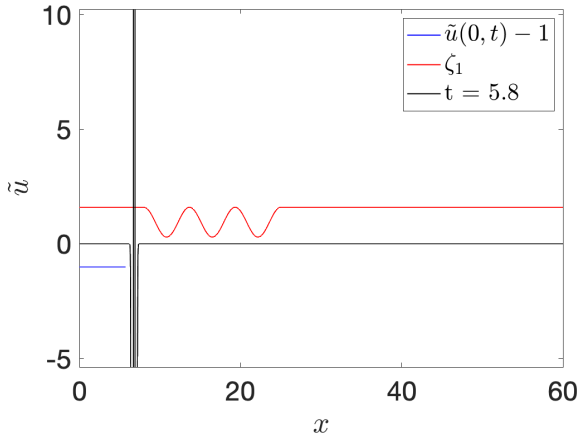
- It is harmful due to x-ray's ionizing radiation so it is necessary to deliver low-radiation doses specifically at the part of the human body being examined.
- It may not provide adequate diagnostic information, which can range only between 60% to 80% accuracy, to make a distinction between malignant and benign tumours.
- It cannot create three-dimensional images in one trial so radiologists need to take multiple images from different angles.

Microwave imaging and ultrasonic imaging avoids these disadvantages where the latter employs acoustic imaging to identify the presence of tumours in [8], [29]. Other examples of acoustic imaging in the field of medicine can be found in [4], [7], [31].

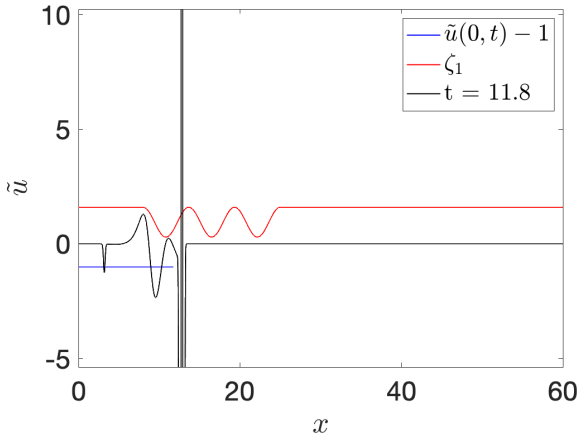
3 Wave Propagation

3.1 Wavelets

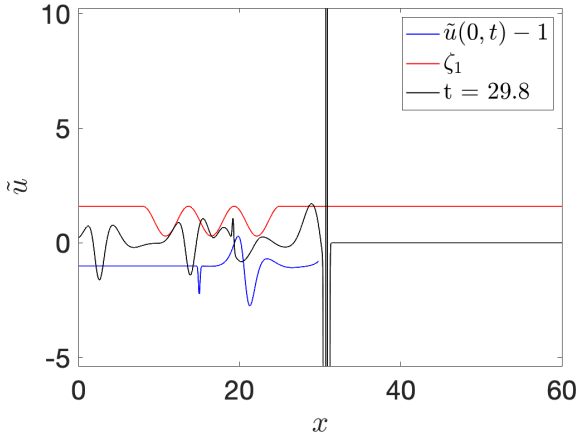
The term wavelets for source wavelets should not be confused with the Daubechies wavelet. Source wavelets generate echoes as they transmit into a given medium. Recorded echoes comprise the measured data. Typically, wavelets have the shape of a concentrated pulse. Their waveforms can be singular, compactly supported or nearly compactly supported, such as the Dirac delta function, the bump function, and the Gaussian, respectively. Fig. 3.1 illustrates the transmission of a wavelet into a medium at three specific moments.



(A) Before the Medium



(B) In the Medium



(C) After the Medium

FIGURE 3.1: Transmission of Wavelet

3.2 Background

One theoretical method on wave propagation for (1.1) is to approximate an impedance, ζ, for a given medium as a piecewise constant function,

$$\zeta_P(x) = \begin{cases} \zeta(x_l) & x < x_1^* \\ \zeta(x_j) & x_{j-1}^* \leq x < x_j^* \qquad (2 \leq j \leq n) \\ \zeta(x_u) & x_n^* \leq x \end{cases} \tag{3.1}$$

where x_j^* for $1 \leq j \leq n$ are equally spaced jump points, and also the midpoints of x_j and x_{j+1}. Since ζ_P is a constant in the intervals $(-\infty, x_1^*)$, (x_{j-1}^*, x_j^*) or (x_n^*, ∞), this simplifies (1.1) to

$$u_{tt} - u_{x_j x_j} = 0 \tag{3.2a}$$

$$u(x_j, 0) = f(x_j), \qquad u_t(x_j, 0) = g(x_j) \tag{3.2b}$$

and shows that a d'Alembert solution exists in each interval. A moving wavelet gets partially reflected and transmitted at each jump point. The wave field is a sum of left- and right-moving components at each point . As the intervals get narrower, the exact solution to (3.2) in each interval generates an approximated wave field that get closer to the exact solution to (1.1).

From this theoretical background, wave propagation in a medium for a given impedance involves a continuum limit of reflections and transmissions at jump points. These reflections and transmissions can differ in intensities depending on the impedance contrast between two consecutive points. The corresponding intensities are the reflectivities, r_j, and the transmittivities, $1 + r_j$ for $j \leq n \in \mathbb{Z}^+$, respectively. The derivation of the reflectivity

The scattering algorithm

Input $\zeta, \alpha, \beta, a, b, n, T$

Spatial grid $\Delta = (b-a)/n, \quad x_j = a + (j-1)\Delta \qquad (1 \le j \le n+1)$

Temporal grid $m = \lfloor T/\Delta \rfloor, \quad t_k = k\Delta \qquad (0 \le k \le m)$

Weights $r_j = (\zeta(x_j) - \zeta(x_{j+1}))/(\zeta(x_j) + \zeta(x_{j+1})) \qquad (1 \le j \le n)$

Variables $v = (v_i^k) \in \mathbb{R}^{(m+1)\times(2n+2)}, \quad \tilde{u} = (\tilde{u}_j^k) \in \mathbb{R}^{(m+1)\times(n+1)}$

Initialization $v_{2j-1}^0 = \alpha(x_j), \quad v_{2j}^0 = \beta(x_j), \quad \tilde{u}_j^0 = v_{2j-1}^0 + v_{2j}^0 \qquad (1 \le j \le n+1)$

$v_1^k = \alpha(x_1 - t_k), \quad v_{2n+2}^k = \beta(x_{n+1} + t_k) \qquad (1 \le k \le m)$

Iterative step Given v^k $(0 \le k \le m-1)$, set

$v_{2j-1}^{k+1} = (1 + r_{j-1})v_{2j-3}^k - r_{j-1}v_{2j}^k \qquad (2 \le j \le n+1)$

$v_{2j}^{k+1} = r_j v_{2j-1}^k + (1 - r_j)v_{2j+2}^k \qquad (1 \le j \le n)$

$\tilde{u}_j^{k+1} = v_{2j-1}^{k+1} + v_{2j}^{k+1} \qquad (1 \le j \le n+1)$

Output $\tilde{u} = (\tilde{u}_j^k) \in \mathbb{R}^{(m+1)\times(n+1)}$

FIGURE 3.2: The Scattering-Based Algorithm

formula involves the Fourier transform and disk automorphism.

In Fig. 3.2, the scattering-based algorithm works with the following inputs:

1. α and β are the initial right- and left-moving source wavelets, respectively,
    - $\beta = 0$ in the context of acoustic imaging
2. a and b are the boundaries of ζ for the spatial grid,
3. n is the number of spatial grid points for the spatial grid, and

4. T is the time to record the measured data for the temporal grid,
    - $T \geq 2(b-a)$ in the context of acoustic imaging

to compute

1. r_j using the reflectivity formula where $j \leq n$,

$$r_j = \frac{\zeta(x_j) - \zeta(x_{j+1})}{\zeta(x_j) + \zeta(x_{j+1})} \tag{3.3}$$

2. v_n^k are the right- and left-moving components when n is odd and even, respectively, and

3. $\tilde{u}$ is the approximated wave field.

in the theoretical approach of wave propagation.

(3.3) points out that $|r_j| < 1$ and, if $r_j = 0$, then the right-moving source wavelet makes a complete transmission with no change in its direction or amplitude.

With the scattering-based algorithm, it is possible to model the measured data. In the context of acoustic imaging of layered media, the source wavelet requires (nearly) compact support waveforms. There are two basic types:

1. regular source

2. singular source

Moreover, the discontinuities of ζ are important to know especially for the case of a singular source wavelet.

3.3 Regular Source

The regular source wavelets are continuously differentiable functions with (nearly) compact support, or also known as regular functions in [11]. The most basic regular source wavelet is the Gaussian

$$W_g(x) = \frac{1}{\varepsilon\sqrt{2\pi}} e^{-\frac{x^2}{2\varepsilon^2}} \tag{3.4}$$

However, any of its derivatives are also regular functions such as the second derivative known as the Ricker

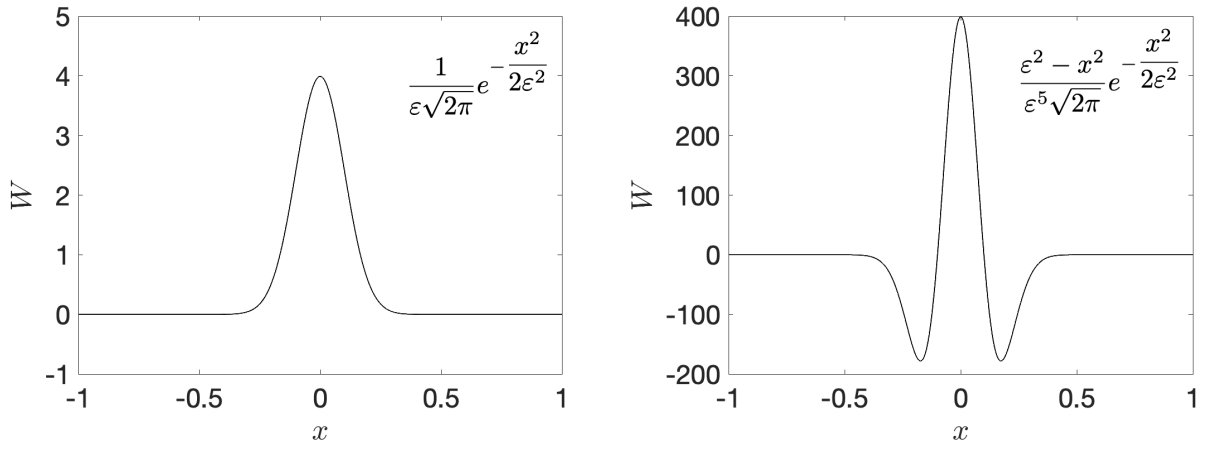
$$W_r(x) = \frac{\varepsilon^2 - x^2}{\varepsilon^5\sqrt{2\pi}} e^{-\frac{x^2}{2\varepsilon^2}} \tag{3.5}$$

Though the Gaussian and its derivatives are symmetric and exponential functions, other regular functions do not need to be the same, such as the asymmetric function

$$W_a(x) = \frac{d^4}{dx^4} \frac{W_g(x)\,\cos\left(\frac{2\pi}{\varepsilon} - \frac{\pi}{12}\right)}{-0.13072} \tag{3.6}$$

in [12]. The scattering-based algorithm in [11] is the original numerical scheme for regular functions. When the regular source wavelet travels through an arbitrary impedance, the wave field is a continuously differentiable function. Fig. 3.3 shows some examples of regular source along with the functions.

3.4 Singular Source



(A) Gaussian (B) Ricker

FIGURE 3.3: Examples of Regular Sources

3.4.1 The Dirac Delta Function

The singular source wavelet applied is a right-moving Dirac delta function, $\delta(x - t - a)$, supported at $x = t - a$. Despite its given name, it is a distribution that is a linear functional that maps functions, known as test functions φ, to their values at 0. It can be defined as

$$\delta[\varphi] = \varphi(0) \tag{3.7}$$

Furthermore, the Dirac delta function is considered broadband because the Fourier transform of the Dirac delta function is a constant function of 1, which shows that the Dirac delta function consists of all frequencies [9]. For simplicity, the Dirac impulse refers to the Dirac delta function, or its rescaling and translations, for the rest of the thesis.

3.4.2 Change in Behaviour

The Dirac impulse may seem simple as it propagates like a regular source wavelet; however, the wave field contains a regular and singular part. Whenever the Dirac impulse hits a discontinuity, two new impulses moving in opposite directions emerge, and may have different amplitudes based on the reflectivity.

1. If ζ has no discontinuity, then only one Dirac impulse with changing amplitude transmits completely without any reproduction of impulses due to no discontinuity, and a smooth function appears on the left side of the impulse.

2. Suppose that ζ has at least one discontinuity. Then at least one Dirac impulse appears in the model of the measured data because there is at least one left-moving Dirac impulse, which emerges when a moving Dirac impulse hits a discontinuity, recorded at x_0.

3.4.3 The Amplitude of an Impulse

Remarkably, there are formula(s) to calculate the exact amplitude(s) of Dirac impulse(s), or the singular part of the wave field, u_{sing}, with respect to ζ. However, when ζ is discontinuous, the number of formulas expands too rapidly as the number of discontinuities increases. Thus the thesis only presents the following formula to calculate u_{sing} when a Dirac impulse travels through a continuous ζ:

$$u_{\text{sing}} = \sqrt{\frac{\zeta(a)}{\zeta(a+t)}} \tag{3.8}$$

where u_{sing} is the amplitude of the Dirac impulse supported at $t = x - a$[1]. The proof of (3.8) is in Appendix E.1. Note that (3.8) is surprisingly similar to (2.7b). If there is a change in variable using (2.4), ζ is the reciprocal of c and the amplitude of the Dirac source wavelet at the reference position $x_0 = a$ is 1, then (2.7b) transforms to (3.8).

3.4.4 Issues with the Dirac Impulse

When the original scattering-based algorithm in [11] is implemented with a Dirac source wavelet, two issues arise:

1. two separate curves appear for the regular part of the wave field, as shown in Fig. 3.4, and
2. amplitudes for the regular part of the wave field vary with different $\Delta's$, as shown in Fig. 3.5

and can be explained by the Theorem of Disjoint Components:

Theorem 1 (The Theorem of Disjoint Components)**.** *Suppose that you have a right-moving Dirac source supported at a single point,*

$$\alpha(x) = c\,\delta(x - t - a) \qquad \textit{and} \qquad \beta(x) = 0$$

Then, when you implement the original scattering-based algorithm in [11] with a Dirac source wavelet, the following statements are true for $j, k \in \mathbb{Z}$.

1. *If k and j are of the same parity, then our right-moving components are $v^k_{2j-1} = 0$.*

[1] $a = 0$ for the numerical experiments and error analysis in Chapter 4.

2. *If k and j are of the opposite parity, then our left-moving components are $v^k_{2j} = 0$.*

Proof. We will represent $k = 2l - 1, j = 2p - 1$ to be odd integers and $k = 2l, j = 2p$ to be even integers where $l, p \in \mathbb{Z}$.

Base Step: $k = 0$

The right-moving Dirac source supported at x_1 tells us that $v^0_1 = c$ and $v^0_j = 0$ for $j \neq 1$. Thus, $v^0_{2j} = 0$ for all $j \in \mathbb{Z}$. In particular, $v^0_{2(2p-1)} = 0$ for all p, which proves (2). Also, $v^0_{2j-1} = 0$ if $j \neq 1$. So, $v^0_{2(2p)-1} = 0$ for all p, which proves (1). Therefore, the theorem is true for $k = 0$ for all $j \in \mathbb{Z}$.

Induction Hypothesis: Suppose that the theorem is true for $0 \leq k \leq s$. We want to prove that the theorem is true for $k = s + 1$.

Induction Step: $k = s + 1$

Firstly, we will prove that our boundary condition satisfies the theorem.

The right-moving Dirac source tells us that

$$v^{s+1}_1 = \begin{cases} c & \text{if } x_1 - t_{s+1} = a \\ 0 & \text{otherwise,} \end{cases}$$

which in the scattering algorithm becomes

$$v^{s+1}_1 = \begin{cases} c & \text{if } -(s+1)\Delta = 0 \\ 0 & \text{otherwise} \end{cases}$$

Since $\Delta > 0$ and $s \geq 0$, $v_1^{s+1} \neq c$. Therefore, $v_1^{s+1} = 0$, which proves (1). In addition, $v_{2j}^{s+1} = 0$ for all $j \in \mathbb{Z}$. In particular, $v_{2n+2}^{s+1} = v_{2(n+1)}^{s+1} = 0$, which proves (2).

Secondly, we will prove that if k and j are of the same parity and $j > 1$, then $v_{2j-1}^{k} = 0$.

Suppose that $s = 2l$ and $j = 2p - 1$. Then

$$\begin{aligned}
v_{2j-1}^{s+1} &= \left(1 + r_{j-1}\right) v_{2j-3}^{s} - r_{j-1} v_{2j}^{s} \\
v_{2j-1}^{2l+1} &= \left(1 + r_{j-1}\right) v_{2(2p-1)-3}^{2l} - r_{j-1} v_{2(2p-1)}^{2l} \\
&= \left(1 + r_{j-1}\right) v_{2(2p-2)-1}^{2l} - r_{j-1} v_{2(2p-1)}^{2l} \\
&= 0
\end{aligned}$$

because the induction hypothesis says:

1. when $j = 2p - 2$ and $s = 2l$, which are of the same parity, we have $v_{2j-1}^{s} = 0$.
2. when $j = 2p - 1$ and $s = 2l$, which are of the opposite parity, we have $v_{2j}^{s} = 0$.

Next, suppose that $s = 2l - 1$ and $j = 2p$. Then

$$\begin{aligned}
v_{2j-1}^{s+1} &= \left(1 + r_{j-1}\right) v_{2j-3}^{s} - r_{j-1} v_{2j}^{s} \\
v_{2j-1}^{2l-1+1} &= \left(1 + r_{j-1}\right) v_{2(2p)-3}^{2l-1} - r_{j-1} v_{2(2p)}^{2l-1} \\
v_{2j-1}^{2l} &= \left(1 + r_{j-1}\right) v_{2(2p-1)-1}^{2l-1} - r_{j-1} v_{2(2p)}^{2l-1} \\
&= 0
\end{aligned}$$

because the induction hypothesis says:

1. when $j = 2p - 1$ and $s = 2l - 1$, which are of the same parity, we have $v_{2j-1}^{s} = 0$.
2. when $j = 2p$ and $s = 2l - 1$, which are of the opposite parity, we have $v_{2j}^{s} = 0$.

Therefore, we can conclude that if k and j are of same parity, then $v^k_{2j-1} = 0$.

Now, we will prove that if k and j are of the opposite parity and $j \geq 1$, then $v^k_{2j} = 0$.

Suppose that $s = 2l$ and $j = 2p$. Then

$$\begin{aligned} v^{s+1}_{2j} &= r_j v^s_{2j-1} + (1 - r_j) v^s_{2j+2} \\ v^{2l+1}_{2j} &= r_j v^{2l}_{2(2p)-1} + (1 - r_j) v^{2l}_{2(2p)+2} \\ &= r_j v^{2l}_{2(2p)-1} + (1 - r_j) v^{2l}_{2(2p+1)} \\ &= 0 \end{aligned}$$

because the induction hypothesis says:

1. when $j = 2p$ and $s = 2l$, which are of same parity, we have $v^s_{2j-1} = 0$.
2. when $j = 2p + 1$ and $s = 2l$, which are of opposite parity, we have $v^s_{2j} = 0$.

Next, suppose that $s = 2l - 1$ and $j = 2p - 1$. Then

$$\begin{aligned} v^{s+1}_{2j} &= r_j v^s_{2j-1} + (1 - r_j) v^s_{2j+2} \\ v^{2l-1+1}_{2j} &= r_j v^{2l-1}_{2(2p-1)-1} + (1 - r_j) v^{2l-1}_{2(2p-1)+2} \\ v^{2l}_{2j} &= r_j v^{2l-1}_{2(2p-1)-1} + (1 - r_j) v^{2l-1}_{2(2p)} \\ &= 0 \end{aligned}$$

because the induction hypothesis says:

1. when $j = 2p - 1$ and $s = 2l - 1$, which are of same parity, we have $v^s_{2j-1} = 0$.
2. when $j = 2p$ and $s = 2l - 1$, which are of opposite parity, we have $v^s_{2j} = 0$.

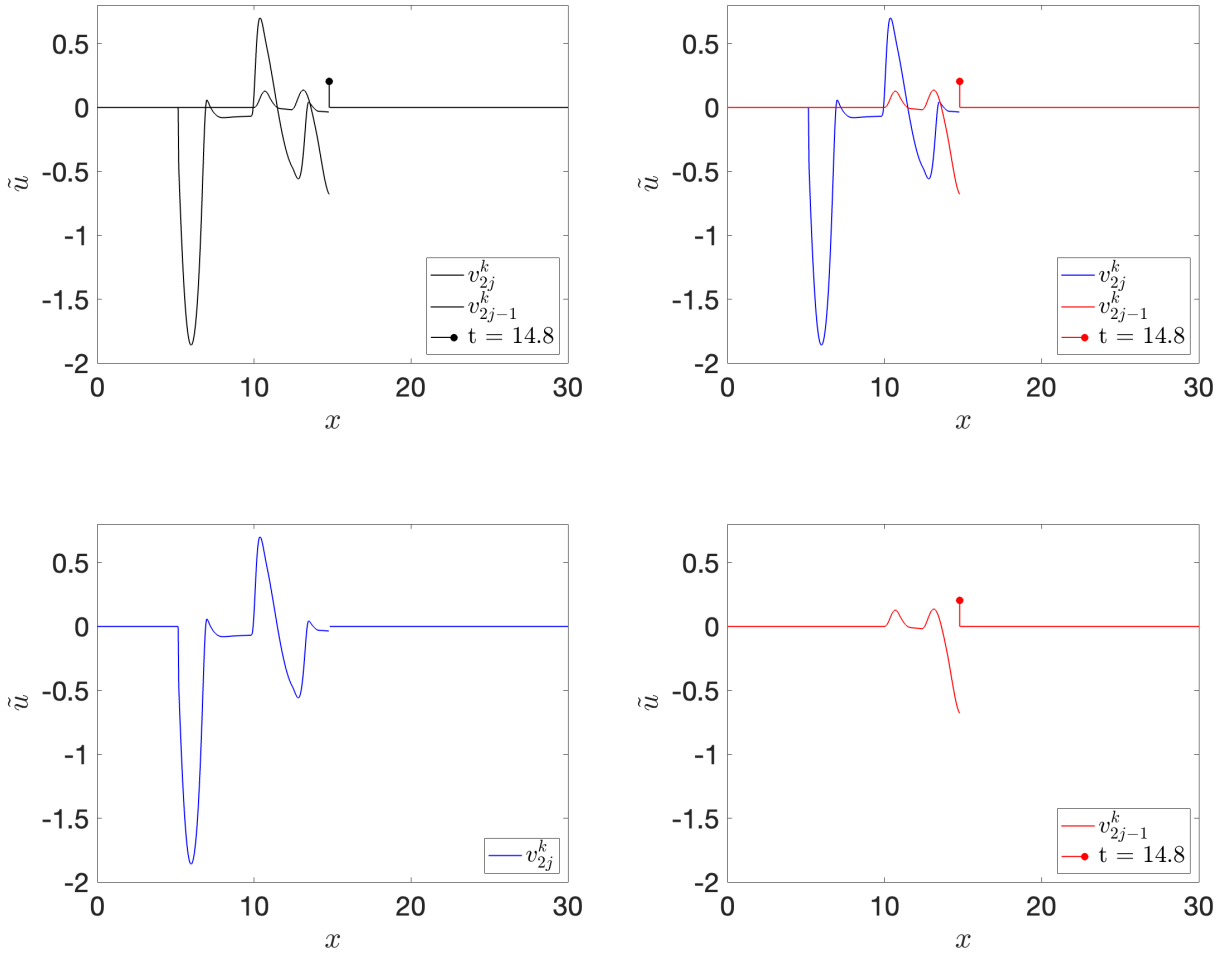
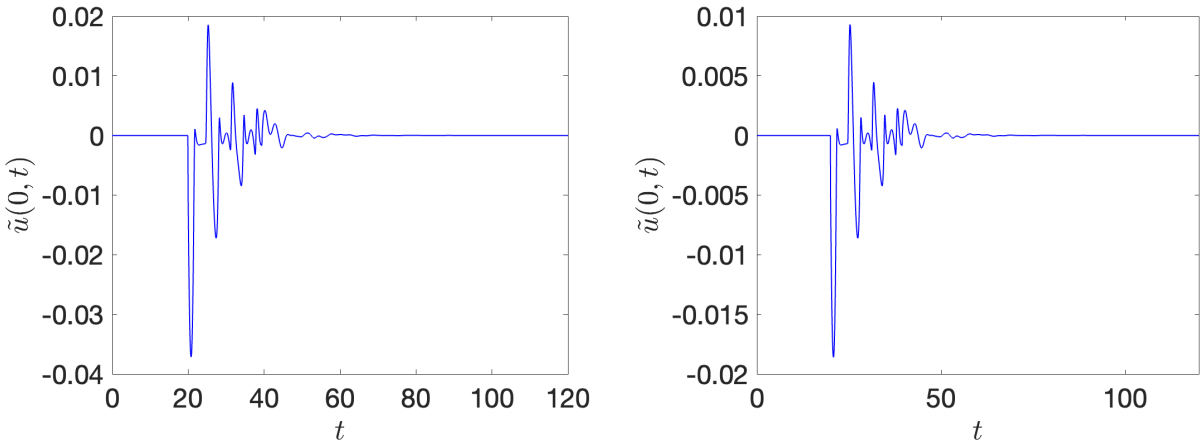


FIGURE 3.4: Top left: A model of the wave field paused at $t = 14.8$. Top right: Distinction between the left- and right-moving components using blue and red, respectively. Bottom left: A model of the left-moving components. Bottom right: A model of the right-moving components.

Therefore, we can conclude that if k and j are of the opposite parity and $j \geq 1$, then $v_{2j}^k = 0$.

By mathematical induction, we have proven that the theorem is true for all $j, k \in \mathbb{Z}$. □

Theorem 1, Theorems 2 and 3 in [11] indicate how to modify the original scattering-based



(A) $\Delta = 0.04$ (B) $\Delta = 0.02$

FIGURE 3.5: Note that when Δ was divided by 2, the amplitudes of the regular part was also divided by 2.

algorithm to accommodate a Dirac source and better model the measured data by combining the disjoint components and renormalizing the amplitudes to exploit a separation of scales phenomenon.

3.5 The Modified Scattering-Based Algorithm for a Dirac Impulse

In this thesis, there are two versions of the modified scattering-based algorithm for a Dirac impulse depending on whether ζ is continuous or not. The modifications to fix the issues for the regular part of the wave field in §3.4.4 apply to both versions; however, each version computes the singular part of the wave field differently:

1. For a continuous ζ, the first version of the modified scattering-based algorithm uses (3.8) in §3.4.3 to compute the exact values for the singular part. The MATLAB code

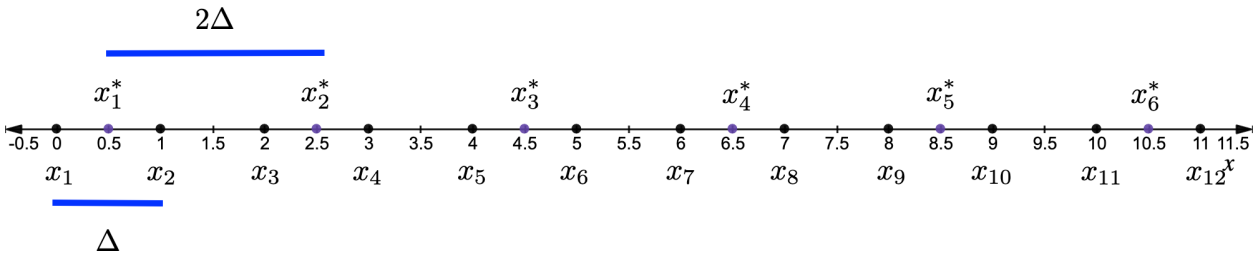


FIGURE 3.6: Adjustments to Spatial Grid

in Appendix A.1.2 implements the first version of the modified scattering-based algorithm.

2. For a discontinuous ζ, the second version of the modified scattering-based algorithm searches for the approximated values for the singular part because the original scattering-based algorithm already computes the approximated values for the singular part correctly. The MATLAB code in Appendix A.1.3 implements the second version of the modified scattering-based algorithm.

With the correct computations of the regular and singular parts of the wave field, we can finally model the wave field properly.

3.5.1 Computation for the Regular Part

For the regular part of the wave field, $\tilde{u}_{\text{reg}}$, the modified scattering-based algorithm works with an additional spatial grid shown in Fig. 3.6 such that x_j^* are the averages of every other pair of consecutive spatial grid points from the original spatial grid and the new locations for $\tilde{u}_{\text{reg}}$. With this new spatial grid, we correctly compute $\tilde{u}_{\text{reg}}$ by taking the

sum of every other pair of consecutive values of $\tilde{u}$ to combine the disjoint components and dividing it by 2Δ[2] to renormalize the amplitudes.

3.5.2 Mapping the Singular Part for Discontinuous ζ

As stated on the list in §1.3.1, the locations of each discontinuity must be at exact midpoints of x_j and x_{j+1}; otherwise, the impulses do not appear in the computed wave field as they are not on any spatial grid point at a certain time.

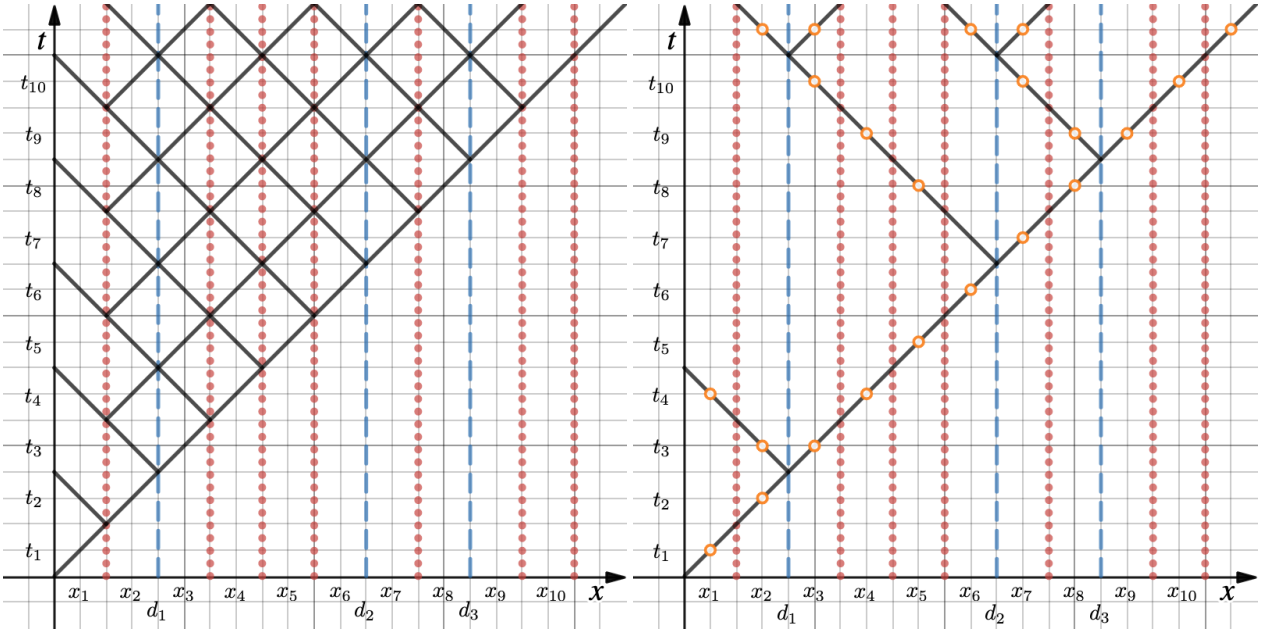
With this in mind, Fig. 3.7 displays two types of tree diagram to illustrate what happens when a right-moving impulse hits equally spaced jump points x_j^*, which are the midpoints of x_j and x_{j+1}:

1. The whole tree diagram - The diagram shows that there are always two new impulses emerging and moving in opposite directions when there is a discontinuity at every jump point.

2. The partial tree diagram - The diagram shows that there are only two new impulses emerging and moving in opposite directions when there is a discontinuity at a distinct jump point.

Using the tree diagrams, the second version of the modified scattering-based algorithm searches for the partial tree in the whole tree to locate the approximated values of the singular part on the spatial grid at a certain time for ζ.

[2] Though Theorems 2 and 3 in [11] shows the rescaling factor is Δ, the rescaling factor is 2Δ in the codes since the distance between the new spatial grid points differs by 2Δ instead of Δ.



(A) Whole Tree Diagram (B) Partial Tree Diagram

FIGURE 3.7: These figures both show the left- and right-moving impulses in the presence of discontinuities. The vertical dotted lines represents the non-existing discontinuities and the vertical dashed lines represents the existing discontinuities. The solid lines represents the reflections and transmissions from the discontinuities.

Suppose that ζ_s and ζ have the same discontinuities $\mathbf{d} = [d_1, d_2, \ldots, d_m]$ where ζ_s represents a simple step function. As an example, Fig. 1.1c and Fig. 1.1d are ζ_s and ζ, respectively. Then the second version of the modified scattering-based algorithm computes two different wave fields, $\tilde{u}_S$ and $\tilde{u}$, for ζ_S and ζ, respectively. The extra step in computing $\tilde{u}_S$ is to help search for the singular part of $\tilde{u}$ because $\tilde{u}_S$ is a matrix having only non-zero entries located at the same spots of the approximated values of the singular part in the matrix $\tilde{u}$. With the matching indices of the singular part between $\tilde{u}_S$ and $\tilde{u}$, it is possible to extract only the singular part of $\tilde{u}$, $\tilde{u}_{\text{sing}}$. Finally, with both $\tilde{u}_{\text{sing}}$ and $\tilde{u}_{\text{reg}}$ from §3.5.1, the modified scattering-based algorithm models $\tilde{u}$ correctly.

Though the code in Appendix A.1.3 only inputs ζ to help with the computation of $\tilde{u}$, it can still compute $\tilde{u}_S$ without needing to input ζ_S using our knowledge of what and where the discontinuities in ζ exist. The code:

1. computes the indices of the discontinuities,
2. finds the reflectivities computed by ζ located at the computed indices
3. creates a zero vector of reflectivities, $\tilde{r}_S$,
4. locates the zeroes in $\tilde{r}_S$ at the computed indices, and
5. replaces the located zeroes with the located reflectivities

to represent the reflectivities for ζ_s, considering the reflectivities for ζ_S are all zeros except at where the discontinuities exist. After the code computes $\tilde{r}_S$, it repeats the Initialization and Iterative Steps in the scattering-based algorithm (refer back to Fig. 3.2) to compute $\tilde{u}_S$. Finally, the code uses the matching indices of the singular part between $\tilde{u}_S$ and $\tilde{u}$ to extract $\tilde{u}_{\text{sing}}$.

It is important to note that $\tilde{u}_{\text{sing}}$ is not the exact wave field, u_S, for ζ_S because $\tilde{r}_S$ is using the approximated values from the reflectivies of ζ. In fact, for a discontinuous ζ, the approximated values of $\tilde{u}_{\text{sing}}$ approach the exact values of the singular part of u_S as Δ decreases. The reason behind this is the theoretical approach of wave propagation used in the scattering-based algorithm. By approximating ζ as a piecewise constant function, $\zeta \cong \zeta_S$ at the discontinuities with a denser grid. Evidence can be shown by computing the relative error of the measured data recorded at $x = 0$ between $\tilde{u}_{\text{sing}}$ and u_S.

4 Inverse Scattering

4.1 Impedance Profile and Source Wavelet

In Chapter 2, it is discussed how ζ and the source wavelet affect the wave propagation, which involves the scattering-based algorithm. However, as stated in §1.2.4, they also play a role in the inverse scattering algorithms. §4.1.1 dives a little deeper in source wavelets. Though all four algorithms begin by interpreting ζ as a step function, the scattering-based algorithm and the inverse scattering algorithms differ slightly in the role played by the source wavelets. As reviewed in §3.5, a singular source wavelet in the scattering-based algorithm requires major changes to the numerical scheme, but there is only a minor change in the inverse scattering algorithms.

4.1.1 The Source Wavelet

In all the inverse scattering algorithms, $W(x)$ is the source wavelet , and it is important to know its first non-zero moment. The formula

$$\mu_j = \int_{-\infty}^{\infty} \frac{x^j}{j!} W(x)\, dx \tag{4.1}$$

Source Wavelet	Formula	k	w
Gaussian	$\frac{1}{\varepsilon\sqrt{2\pi}}e^{-\frac{x^2}{2\varepsilon^2}}$	0	1
Ricker	$\frac{\varepsilon^2-x^2}{\varepsilon^5\sqrt{2\pi}}e^{-\frac{x^2}{2\varepsilon^2}}$	2	-1
Dirac	$\delta(x-a)$	0	1

TABLE 4.1: Source Wavelets and their Non-Zero Moments

defines the j-th moment of W where $j \geq 0$. The first non-zero moment of W is μ_k where $k = j$ is the minimum integer such that $\mu_k \neq 0$. For all three inverse scattering algorithms, k and $\mu_k = w$ are inputs. Table 4.1 shows the experimental source wavelets and their non-zero moments.

Furthermore, the source wavelet can be categorized in having the original or a virtual source waveform depending on k.

1. If $\mu_k \neq 0$ when $k = 0$, then the source wavelet takes on the original source waveform as W.

2. If $\mu_k \neq 0$ when $k > 0$, the the source wavelet takes on a virtual source waveform as

$$V = (-1)^k \int_{-\infty}^{x} \frac{(x-s)^{k-1}}{(k-1)!} W(s)\, ds \tag{4.2}$$

4.1.2 Reflected Data

By defining the reflection Green's function G_ζ in the case of a unit Dirac source as

$$G_\zeta(t) = \begin{cases} 0 & t \leq 0 \\ u(0,t) & t > 0 \end{cases} \tag{4.3}$$

the left-moving part of the solution to (1.1) in terms of G_ζ can be represented as

$$\mathcal{D}^{(-k)}(t) = \begin{cases} \tilde{W} * G_\zeta(t) & k = 0 \\ \tilde{V} * G_\zeta(t) & k \neq 0 \end{cases} \tag{4.4}$$

where

1. $\mathcal{D}^{(-k)}$ is the k-fold antiderivative of $\mathcal{D}$ and has the structure of the measured data corresponding to W if $\mu_0 \neq 0$, or V if $\mu_k \neq 0$ for some $k > 0$,
2. $\tilde{W} = W(-x)$, and
3. $\tilde{V} = V(-x)$.

See [15]. Some inverse scattering algorithms rely on finding an inverse filter $F(t)$ for the wavelet $\tilde{W}$ such that $\tilde{W} * F(t) = \delta(t)$ and use $G_\zeta(t)$ to compute ζ.

$$\begin{aligned} D^{(-k)}(t) &= G_\zeta(t) * \tilde{W} \\ D^{(-k)}(t) * F(t) &= G_\zeta(t) * \tilde{W} * F(t) \\ G_\zeta(t) &= D^{(-k)}(t) * F(t) \end{aligned} \tag{4.5}$$

However, it is not possible to find $F(t)$ such that $F(t) * \tilde{W} = \delta(t)$ in practice due to the

band-limited nature of a wavelet so deconvolution can still lead to huge errors [23]. Fortunately, the thesis presents one inverse scattering algorithm that avoids deconvolution and returns amazing results.

Finally, it is important to note that the modified scattering-based algorithms compute the regular and singular parts of the wave field correctly so it can show the accurate model of the measured data unlike the graphs in Fig. 3.4. However, a singular source wavelet in the original scattering-based algorithm still computes the disjoint left- and right-moving components in the correct sense, but only misses the rescaling factor of Δ as there is no change in the original spatial grid.

4.2 Born Approximation

A quick derivation of the Born approximation

$$\tilde{\zeta}(x) = \zeta_l \, e^{-2 \int_{-\infty}^{2x} G_\zeta(t)\, dt} \tag{4.6}$$

where

$$G_\zeta(t) = \begin{cases} \dfrac{\tilde{u}(0,t)}{w} & \text{for regular source wavelet} \\ \dfrac{\tilde{u}(0,t)}{\Delta} & \text{for singular source wavelet} \end{cases} \tag{4.7}$$

is in [15, p.2-3]. As mentioned in §1.3.2, the Born approximation is not very accurate because it works with single scattering based on the approximation $G_\zeta \cong R_\zeta$, which are the corresponding reflection Green's function and the reflectivity function in [15]. As the algorithm is dealing with discrete temporal grid points, it uses cumulative sums to

compute the integral of $\int_{-\infty}^{2x} G_\zeta(t)\,dt$. The various codes of the Born approximation are in Appendix B.1.

4.3 Refined Impedance Transform

The refined impedance transform has a similar algorithm as the Born approximation but it avoids some of the flaws in Born approximation. It avoids single scattering, and can recover ζ exactly at the far side of an inhomogeneity. The codes for the refined impedance transform follows the same steps as the codes for the Born approximation until it reaches its inverse scattering formula:

$$\tilde{\zeta}(x) = \zeta_l \frac{w - \int_{-\infty}^{2x} g(t)\,dt}{w + \int_{-\infty}^{2x} g(t)\,dt} \tag{4.8}$$

where

$$g(t) = \begin{cases} \tilde{u}(0,t) & \text{for regular source wavelet} \\ \dfrac{\tilde{u}(0,t)}{\Delta} & \text{for singular source wavelet} \end{cases} \tag{4.9}$$

Similarly to the Born approximation, cumulative sums compute the integral of $\int_{-\infty}^{2x} g(t)\,dt$. The various codes for the refined impedance transform are in Appendix C.1.

4.3.1 The Process

The definition of the refined impedance transform for a general integrable distribution $g \in \mathcal{S}'(\mathbb{R})$ is the formula

$$\mathcal{F}_{w,c}\, g(x) = c\, \frac{w - \int_{-\infty}^{2x} g(t)\, dt}{w + \int_{-\infty}^{2x} g(t)\, dt} \tag{4.10}$$

where $c > 0 \in \mathbb{R}$. Suppose that $g(t) = \mathcal{D}$ is the reflection data measured at $x = 0$ for (1.1) and $\mu_0 \neq 0$. Then Theorem 1 in [15] shows that

$$\mathcal{F}_{w,\zeta_l}\, \mathcal{D}(x) \to \zeta_u \qquad \text{as} \qquad x \to \infty \tag{4.11}$$

which means (4.10) can recover ζ_u in the interval $[x_l, x_u]$ exactly without deconvolution for any source wavelet given that the recording time for $\mathcal{D}$ at $x = 0$ is at least $2x_u$. Furthermore, there is a convergence where $\mathcal{F}_{w,\zeta_l}\, \mathcal{D}(x_u) \cong \zeta_u$ that can extend to find

$$\zeta(x) \cong \mathcal{F}_{w,\zeta_l}\, \mathcal{D}(x) \tag{4.12}$$

by replacing x_u with $x < x_u$. Theorem 2 in [15] explains a similar idea when $\mu_k \neq 0$ for some $k > 0$ where

$$\mathcal{F}_{w,\zeta_l} \mathcal{D}^{(-k)}(x) \to \zeta_u \qquad \text{as} \qquad x \to \infty \tag{4.13}$$

and

$$\zeta(x) \cong \mathcal{F}_{w,\zeta_l} \mathcal{D}^{(-k)}(x) \tag{4.14}$$

4.3.2 The Derivation

Suppose that $\int_{-\infty}^{\infty} W = \mu_0 = w \neq 0$. By defining

$$r_j = \frac{\zeta(x_j) - \zeta(x_{j+1})}{\zeta(x_j) + \zeta(x_{j+1})} \qquad \text{and} \qquad \tau_j = 2\left(x_j - x_{j-1}\right) \qquad (1 \leq j \leq n \in \mathbb{Z}), \tag{4.15a}$$

$$r = (r_1, \ldots, r_n) \qquad \text{and} \qquad \tau = (\tau_1, \ldots, \tau_n), \tag{4.15b}$$

analysis on the nonlinear measurement operator $\zeta \mapsto \mathcal{D}$ shows that we can express ζ in terms of (τ, r). We can rewrite (4.4) in the form

$$\mathcal{D} = {}_{\chi[0,T]} W * G^{(\tau,r)}, \tag{4.16}$$

as in [13, p. 89], where

$$\begin{aligned} G^{(\tau,r)}(t) &= G_\zeta(t) \\ &= \sum_{k \in 1 \times \mathbb{Z}^{n-1}} a(r,k)\, \delta\left(t - \langle \tau, k \rangle\right), \end{aligned} \tag{4.17}$$

is the reflection Green's function, and $a(r,k)$, are the polynomials in terms of r from Theorem 3.4 in [14, p. 933]. Furthermore, Theorem 1.2 in [13, p. 90] ensures that

$$\begin{aligned} \sum_{k \in 1 \times \mathbb{Z}^{n-1}} a(r,k) &= \tanh\left(\sum_{j=1}^{n} \tanh^{-1} r_j\right) \\ &= \tanh\left[\frac{1}{2}\left(\log \zeta(x_j) - \log \zeta(x_{j+1})\right)\right] \end{aligned} \tag{4.18}$$

converges absolutely because $\left|r_j\right| < 1$. Then,

$$\begin{aligned}\int_{-\infty}^{\infty} G_\zeta &= \sum_{i=1}^{\infty} a_i \\ &= \tanh\left[\frac{1}{2}\left(\log\zeta_l - \log\zeta_u\right)\right] \\ &= \frac{\zeta_l - \zeta_u}{\zeta_l + \zeta_u}\end{aligned} \tag{4.19}$$

where $a_i \neq 0$ are the amplitudes of the reflection Green's function, and

$$\begin{aligned}\zeta_u &= \zeta_l \frac{1 - \int_{-\infty}^{\infty} G_\zeta}{1 + \int_{-\infty}^{\infty} G_\zeta} \\ &= \zeta_l \frac{\int_{-\infty}^{\infty} W - \int_{-\infty}^{\infty} W \int_{-\infty}^{\infty} G_\zeta}{\int_{-\infty}^{\infty} W + \int_{-\infty}^{\infty} W \int_{-\infty}^{\infty} G_\zeta} \\ &= \zeta_l \frac{w - \int_{-\infty}^{\infty} \mathcal{D}}{w + \int_{-\infty}^{\infty} \mathcal{D}} \\ &= \lim_{x\to\infty} \mathcal{F}_{w,\zeta_l}\, \mathcal{D}\end{aligned} \tag{4.20}$$

after rearranging the terms in (4.19). Finally, the extension of this idea with replacing x_u with $x < x_u$, we conclude

$$\begin{aligned} \zeta(x) &\cong \mathcal{F}_{w,\zeta_l}\mathcal{D}(x) \\ \tilde{\zeta}(x) &= \zeta_l \frac{w - \int_{-\infty}^{\infty} \mathcal{D}(t)\,dt}{w + \int_{-\infty}^{\infty} \mathcal{D}(t)\,dt} \\ &= \zeta_l \frac{w - \int_{-\infty}^{2x} g(t)\,dt}{w + \int_{-\infty}^{2x} g(t)\,dt} \end{aligned} \tag{4.21}$$

In the case where $\int_{-\infty}^{\infty} W = 0$, the proof of Theorem 2 in [15] follows in the same manner where $\mathcal{D}^{(-k)}(t) = \tilde{V} * G_{\zeta}(t)$ instead of $\mathcal{D}(t) = \tilde{W} * G_{\zeta}(t)$.

4.4 Echoes-to-Impedance Transform

There are some aspects of the previous two algorithms in §4.2 and §4.3 that the echoes-to-impedance transform avoids:

1. deconvolution in the Born approximation,
2. approximation of the reflection Green's function using a simplified model.

The echoes-to-impedance transform avoids these issues by using the data itself as an input with no need for deconvolution and exploiting a newly discovered connection between orthogonal polynomials on the unit circle (OPUC) and the reflection Green's function. These remarkable aspects are what sets the echoes-to-impedance transform apart from the other inverse scattering algorithms in accuracy and efficiency, making it the best

of the three to recover $\tilde{\zeta}$ from the measured data. A more detailed discussion on the distinguishing features of the echoes-to-impedance transform is in [12, p.4].

Based on the performance of the echoes-to-impedance transform, the numerical experiments in the thesis also consider data downsampling with the algorithm. From a set of temporal grid points, the echoes-to-impedance transform can still recover $\tilde{\zeta}$ quite accurately even though it takes only a subset of equally spaced temporal grid points. In the case of the input,

$$D(t) = \begin{cases} \tilde{u}(0,t) & \text{for regular source wavelet} \\ \dfrac{\tilde{u}(0,t)}{\Delta} & \text{for singular source wavelet} \end{cases} \tag{4.22}$$

The codes for the echoes-to-impedance transform for various conditions are in Appendix D.1.

Another unique property of the echoes-to-impedance transform is computing $\tilde{\zeta}$, which is a function of x, directly from the time domain unlike other direct methods such as the MUSIC algorithm, linear sampling and factorization method in [18, Ch.6], [20], [19], which use time harmonic data to reproduce $\tilde{\zeta}$.

4.4.1 The Process

A deep theoretical fact underlying the echoes-to-impedance transform is that if ζ_n is a step function with n equally spaced jump points, then the associated reflection Green's function G_n may be expressed in terms of orthogonal polynomials on the unit circle (OPUC) relative to some measure μ_n on the circle determined by G_n. (Note that the measure μ_n is not discrete; on the contrary, it is absolutely continuous with respect to Lebesgue

measure.) Remarkably, the recurrence coefficients for the orthogonal polynomials determined by μ_n are precisely the reflectivities associated with ζ_n. The latter fact is proved in [12, §5], and is beyond the scope of the present thesis. The basic idea behind the echoes-to-impedance transform is described in [12, §4.1]:

> The echoes-to-impedance transform combines the idea of approximating arbitrary ζ using step functions with the connection to OPUC as follows. The sampling rate of a digital recording imposes a grid (in time); from the perspective of the scattering algorithm in [11], the given temporal grid has an associated spatial grid that corresponds in turn to a step function approximation ζ_n of ζ. Thus sampled data may be interpreted as containing information about the Green's function G_n associated to ζ_n. OPUC then allows one to compute reflectivities (associated to jumps in ζ_n) directly from the measured data, and hence to recover ζ_n exactly. Since ζ_n agrees with ζ on a grid X, one thereby obtains values of ζ on this spatial grid. The process imposes no restrictions on ζ; it can have discontinuities or be smooth etc.

The Green's function G_n corresponding to ζ_n has the form of a delta train,

$$G_n(t) = \sum_{j=1}^{\infty} \alpha_j \, \delta\left(t - t_j\right) \qquad t_j = \frac{\Delta}{2} + (j-1)\,\Delta \qquad (1 \leq j < \infty)\,. \tag{4.23}$$

Denote by D the measured scattering data, $D = \left\{\left(t_j, a_j\right)\right\}_{j=1}^{n}$ consisting of amplitudes $a_j = \mathcal{D}\left(t_j\right)$. Then, the echoes-to-impedance transform can be divided into four steps:

1. Preliminary step: $D \mapsto \alpha$ where $\alpha = (\alpha_1, \ldots, \alpha_n)$ is a vector with the first n coefficients of G_n as entries,

2. $\alpha \mapsto m$ where $(m_1, \ldots, m_n)$ is the vector of the first n moments of μ_n,

3. $m \mapsto r$ where $r = (r_1, \ldots, r_n)$ is the reflectivity vector, and

4. $r \mapsto \zeta_n$

Note that the moments computed in step (2) allow one to generate the corresponding sequence of orthogonal polynomials (i.e., one can carry out the Gram-Schmidt procedure starting with monomials to yield the sequence of orthogonal polynomials), and this in turn allows one to compute the recurrence coefficients arising in the standard three-term-recurrence relation satisfied by orthogonal polynomials on the unit circle. This is what underlies step (3).

4.4.2 The Derivation

According to [12], the preliminary step in the echoes-to-impedance transform is based on three assumptions:

(a) G_n approximates G weakly (i.e., for any smooth test function φ, $\int G_n \varphi \approx \int G\varphi$)

(b) G_n varies slowly at the scale of ε in the sense that $\alpha_j \cong \alpha_i$ if $|t_j - t_i| < \varepsilon$

(c) if $t_j > \varepsilon$ then $\int W \cong \sum_{i=1}^{\infty} W(t_i - t_j)\Delta$

Suppose that $\int_{-\infty}^{\infty} W = \mu_0 = w \neq 0$. The preliminary step $D \mapsto \alpha$ applies the three assumptions listed above as follows.

$$\begin{aligned}
a_j &= \mathcal{D}\left(t_j\right) \\
&\cong \mathcal{D}_n\left(t_j\right) && \text{(since } G_n \cong G \text{ weakly)} \\
&\cong \sum_{i=1}^{\infty} \alpha_i W\left(t_i - t_j\right) \\
&\cong \alpha_j \sum_{i=1}^{\infty} W\left(t_i - t_j\right) && \text{(since } G_n \text{ varies slowly)} \\
&\cong \alpha_j \frac{w}{\Delta} && \left(\text{since } \int W \cong \sum_{i=1}^{\infty} W\left(t_i - t_j\right)\Delta\right) \\
\alpha_j &\cong \frac{a_j \Delta}{w} && (1 \leq j \leq n)
\end{aligned}$$

Next, by taking the Fourier transform of (4.23),

$$\hat{G}_n(\sigma) = e^{-\frac{i\Delta}{2}} \sum_{j=1}^{\infty} \alpha_j \left(e^{i\Delta\sigma}\right)^j \tag{4.24}$$

we can define

$$R(z) = \sum_{j=1}^{\infty} \alpha_j z^j \tag{4.25}$$

where $R\left(e^{-\Delta\sigma}\right) = e^{\frac{i\Delta}{2}} \hat{G}_n(\sigma)$. By setting

$$r_j = \frac{\zeta\left(x_{j-1}\right) - \zeta\left(x_j\right)}{\zeta\left(x_{j-1}\right) + \zeta\left(x_j\right)} \qquad (1 \leq j \leq n) \tag{4.26}$$

and defining

$$\Psi_z^{r_j}(v) = z\frac{r_j + v}{1 + r_j v} \tag{4.27}$$

we can express R as a composition of maps on the unit disk $\mathbb{D}$

$$R(z) = \Psi_z^{r_1} \circ \cdots \circ \Psi_z^{r_n}(0). \tag{4.28}$$

It turns out (see [12]) that

$$F = \frac{1+R}{1-R} \tag{4.29}$$

defines a mapping $F : \mathbb{D} \to \mathbb{C}_+$ that is holomorphic from the disk to the right half-plane with $F(0) = 1$. The Herglotz Representation Theorem then asserts that we can represent F in terms of a probability distribution $d\mu_n$ on the unit circle S^1:

$$\begin{aligned} F(z) &= \int_{S^1} \frac{\xi + z}{\xi - z}\, d\mu_n(\xi) \\ &= 1 + 2\sum_{j=1}^{\infty} m_j z^j \qquad \text{where} \qquad m_j = \int_{S^1} \bar{\xi}^j\, d\mu_n(\xi) \end{aligned}$$

Thus, up to a factor of 2, the Taylor coefficients of F are the (conjugate) moments of the measure μ_n.

The probability measure μ_n on the unit circle determines a sequence of monic, orthogonal polynomials (in the variable $z \in \mathbb{C}$) whose recurrence coefficients coincide with the reflectivities associated to ζ_n. One can use the given moments m_j to iteratively generate the corresponding sequence of OPUC, and, at the same time, compute the associated recurrence coefficients r_j.

Finally, after finding r_j in terms of m_j, we can obtain $\tilde{\zeta}(x)$ by referring back to (4.26):

$$\begin{aligned} r_j &= \frac{\zeta(x_{j-1}) - \zeta(x_j)}{\zeta(x_{j-1}) + \zeta(x_j)} \\ \tanh^{-1} r_j &= \tanh^{-1} \frac{\zeta(x_{j-1}) - \zeta(x_j)}{\zeta(x_{j-1}) + \zeta(x_j)} \\ &= \ln \sqrt{\frac{\zeta(x_{j-1})}{\zeta(x_j)}} \\ \zeta(x_j) &= \zeta(x_0)\, e^{-2\sum_{i=1}^{j} \tanh^{-1} r_j} \end{aligned}$$

In the case where $\int_{-\infty}^{\infty} W = 0$, only the preliminary step $D \mapsto \alpha$ in the derivation of the echoes-to-impedance transform changes, the essential idea being to use a virtual wavelet in place of W as was done for the refined impedance transform.

4.5 Comparing the Three Inverse Scattering Algorithms

All these inverse scattering algorithms have to do the same job: find $\tilde{\zeta} \cong \zeta$; however, their performances differ greatly. There are two ways to check how well each inverse scattering algorithm finds $\tilde{\zeta}$:

1. inversion plots - plots of $\tilde{\zeta}$ and ζ on the same grid (least accurate)
2. l_2 relative error (most accurate)

Inversion plots only give an idea on which inverse scattering algorithm works the best; however, sometimes it is hard to tell depending on the important components in §1.2.4.

The l_2 relative error is most natural.

$$E = \frac{\|\zeta - \tilde{\zeta}\|}{\|\zeta\|} \tag{4.30}$$

Numerical experiments in Chapter 4 do not only show the modified scattering-based algorithm provides the best recorded data at x_0 for the inverse scattering algorithms, but also proves the echoes-to-impedance transform is the best out of the three inverse scattering algorithms, which is the purpose of the thesis.

4.6 Noise In Data

In reality, data involves noise, which changes the relative errors with each inverse scattering algorithm. However, even with noise, the ranking of the inverse scattering algorithms are still the same. By adding a certain proportion of noise to the data, then the inverse scattering algorithms may not work or may cause slight disfigurations in $\tilde{\zeta}$. As noise increases, the relative error for each inverse scattering algorithm should also be increasing; however, noise can also decrease the relative error due to randomness. As the Dirac source wavelet provides the best measured data for the inverse scattering algorithms, the thesis only shows inverse scattering with noisy data from a Dirac source wavelet. One important note about using noisy data for inverse scattering is the level of sensitivity to noise for each inverse scattering algorithm in regards to how complicated ζ gets. As ζ gets more complicated, the level of sensitivity to noisy data may increase as well.

5 Numerical Experiments and Error Analysis

5.1 Introduction

For the numerical experiments and error analyses using the impedances and source wavelets from §1.2.4, 3.3, and 3.4, the thesis provides a reference in §5.2 to the figures and tables in §5.3. Nevertheless, the numerical experiments and error analysis along with their figures[1] and tables use the same specifications.

5.1.1 Source Wavelets

For the regular source wavelets, they use the following codes:

1. the scattering-based algorithm in Appendix A.1.1
2. the Born approximation in Appendix B.1.1
3. the refined impedance transform in Appendix C.1.1
4. the echoes-to-impedance transform in Appendix D.1.1

[1] The figures in the thesis are the figures from the code with minor adjustments in the figure properties in MATLAB.

For the singular source wavelets, they use the following codes:

1. the scattering-based algorithm in Appendix A.1.2 and A.1.3, except Appendix A.1.2
2. the Born approximation in Appendix B.1.2
3. the refined impedance transform in Appendix C.1.2
4. the echoes-to-impedance transform in Appendix D.1.2

5.1.2 Wave Propagation

The wave propagation uses the scattering-based algorithm with the following settings:

1. $n = 3000$ unless otherwise specified
2. $c = 0$
3. $d = 60$
4. For the regular source wavelets:
   - $\alpha(x) = \dfrac{10}{\sqrt{2\pi}} e^{-50(x-1)^2}$ for the Gaussian source wavelet
   - $\alpha(x) = \dfrac{1000}{\sqrt{2\pi}} \left[0.01 - (x-1)^2\right] e^{-50(x-1)^2}$ for the Ricker source wavelet

and the figures of wave propagation at different times have the following settings:

1. the blue line for the reflected wave recorded at $x_0 = 0$
2. the black line for the source wavelet propagating through ζ
3. the red line for ζ

In the codes, `scale` is a scale factor to magnify the amplitude of the source wavelet to have a clearer view on how the wavelet propagates through ζ. If there is no specified scale factor for a source wavelet, then the scale factor is 1. However, the measured data are at its actual amplitude for inverse scattering to perform properly.

5.1.3 Inverse Scattering

For any plot on inverse scattering, the Born approximation , refined impedance transform use a thinner dashed line than the echoes-to-impedance transform to show the very accurate $\tilde{\zeta}$ obtained from the latter algorithm. Unfortunately, the MATLAB's `myplot` function in Appendix F.1 may not display the best images as it may show the vertical dashed lines at discontinuities for $\tilde{\zeta}$. For some of the figures and tables, the names are abbreviated:

1. BA for Born approximation
2. RIT for refined impedance transform
3. E2IT for echoes-to-impedance transform

For the comparison of one impedance, a highlighted row or column in a table of relative errors shows which inverse scattering algorithm gives the lowest relative error and a bolded text in a table cell reveals the lowest relative error of them all. When there is noisy data, the Dirac inverse scattering plots separate graphs in a same ascending order of the noise percentage where the black line is the actual ζ. These graphs provide evidence on the level of sensitivity to noise for each inverse scattering algorithm. Finally, (4.30) calculates the relative errors for inverse scattering with or without noise.

5.1.4 Data Downsampling with the Echoes-to-Impedance Transform

Based on its high accuracy in returning $\tilde{\zeta}$, the echoes-to-impedance transform also performs data downsampling. For consistency, every continuous and discontinuous ζ's uses $n = 15000$ as the initial number of spatial grid points due to the restriction between discontinuities and spatial grid points for the Dirac source wavelet. Surprisingly though, the discontinuous ζ's cannot use the Dirac source wavelet for data downsampling as the problem of complex values appears right away. There is a pattern that as the number of temporal grid points decreases, the model of $\tilde{\zeta}$ distorts more. There is definitely a minimum number of temporal grid points for the echoes-to-impedance transform to work; however, there should not be any problem regenerating $\tilde{\zeta}$ between the minimum and maximum number of temporal grid points that is a subset of the original number of temporal grid points.

5.1.5 Running Time of the Algorithms

There are also tables to compare the running times for each inverse scattering algorithm as n doubles starting from $n = 1000$ for continuous ζ's, or only at $n = 3000$ and $n = 15000$ for discontinuous ζ's since the discontinuities must be at midpoints of spatial grid points to correctly implement the scattering-based algorithm. These tables show how fast each algorithm runs, and, though one algorithm may run faster than the others, it is not always true that the fastest algorithm brings back the best results.

It is essential to understand that these are estimated times because the running time, say R, varies slightly for several trials for one value of n or can vary differently depending on the computer used. With enough data on the running times for different initial values

of n, it is possible to find a linear relationship between $\ln R$ and $\ln n$ to approximate the running time of a code for an algorithm for a particular device, which is useful when analysts work with huge sets of data and need to know how long they may need to wait for their device to run the algorithms. The numerical experiments in this thesis can only provide sufficient data on the running time for continuous ζ's, but not for discontinuous ζ's because there are two reasons:

1. The computer used for the numerical analyses only works up to $n = 16000$ initial spatial grid points, and

2. The restrictions of the discontinuous being at midpoints of spatial grid points correctly implement the scattering-based algorithm can only provide up to two sets of measured data on running times for the inverse scattering algorithms.

An example on finding a linear relationship between $\ln R$ and $\ln n$ is given in Appendix G.1 for the echoes-to-impedance transform.

5.2 References to Figures and Tables from Numerical Experiments and Error Analysis

5.2.1 Smooth Ramp

1. Fig. 5.1 shows the wave propagation for the Gaussian source wavelet in a smooth ramp at different times. An animation of the wave propagation is available: Gaussian Smooth Ramp.

2. Fig. 5.2 shows the Gaussian scattering, inverse scattering plots, and Table 5.1 shows their corresponding tables of relative errors.

3. Fig. 5.3 shows data downsampling using Gaussian measured data.

4. Fig. 5.4 shows the wave propagation for the Ricker source wavelet in a smooth ramp at different times. An animation of the wave propagation is available: Ricker Smooth Ramp.

5. Fig. 5.5 shows the Ricker scattering, inverse scattering plots, and Table 5.2 show their corresponding tables of relative errors.

6. Fig. 5.6 shows data downsampling using Ricker measured data.

7. Fig. 5.7 shows the wave propagation for the Dirac source wavelet in a smooth ramp at different times. An animation of the wave propagation is available: Dirac Smooth Ramp.

8. Fig. 5.8 shows the Dirac scattering, inverse scattering plots, and Table 5.3 their corresponding tables of relative errors.

9. Fig. 5.9 shows data downsampling using Dirac measured data.

10. Fig. 5.10 shows the Dirac inverse scattering when there is Gaussian noisy data for the classical Born approximation.

11. Fig. 5.11 shows the Dirac inverse scattering when there is uniform noisy data for the classical Born approximation.

12. Fig. 5.12 shows the Dirac inverse scattering when there is Gaussian noisy data for the refined impedance transform.

13. Fig. 5.13 shows the Dirac inverse scattering when there is uniform noisy data for the refined impedance transform.

14. Fig. 5.14 shows the Dirac inverse scattering when there is Gaussian noisy data for the echoes-to-impedance transform.

15. Fig. 5.15 shows the Dirac inverse scattering when there is uniform noisy data for the echoes-to-impedance transform.

16. Table 5.13a gives a table of the l_2 relative errors for each inverse scattering algorithm as n increases.

17. Table 5.13b gives a table of approximated computational times for each inverse scattering algorithm as n increases.

5.2.2 Toothed Ramp

1. Fig. 5.16 shows the wave propagation for the Gaussian source wavelet in a toothed ramp at different times. An animation of the wave propagation is available: Gaussian Toothed Ramp.

2. Fig. 5.17 shows the Gaussian scattering, inverse scattering plots, and Table 5.4 shows their corresponding tables of relative errors.

3. Fig. 5.18 shows data downsampling using Gaussian measured data.

4. Fig. 5.19 shows the wave propagation for the Ricker source wavelet in a toothed ramp at different times. An animation of the wave propagation is available: Ricker Toothed Ramp.

5. Fig. 5.20 shows the Ricker scattering, inverse scattering plots, and Table 5.5 shows their corresponding tables of relative errors.

6. Fig. 5.21 shows data downsampling using Ricker measured data.

7. Fig. 5.22 shows the wave propagation for the Dirac source wavelet in a toothed ramp at different times. An animation of the wave propagation is available: Dirac Toothed Ramp.

8. Fig. 5.23 shows the Dirac scattering, inverse scattering plots, and Table 5.6 shows their corresponding tables of relative errors.

9. Fig. 5.24 shows data downsampling using Dirac measured data.

10. Fig. 5.25 shows the Dirac inverse scattering when there is Gaussian noisy data for the classical Born approximation.

11. Fig. 5.26 shows the Dirac inverse scattering when there is uniform noisy data for the classical Born approximation.

12. Fig. 5.27 shows the Dirac inverse scattering when there is Gaussian noisy data for the refined impedance transform.

13. Fig. 5.28 shows the Dirac inverse scattering when there is uniform noisy data for the refined impedance transform.

14. Fig. 5.29 shows the Dirac inverse scattering when there is Gaussian noisy data for the echoes-to-impedance transform.

15. Fig. 5.30 shows the Dirac inverse scattering when there is uniform noisy data for the echoes-to-impedance transform.

16. Table 5.14a gives a table of the l_2 relative errors for each inverse scattering algorithm as n increases.

17. Table 5.14b gives a table of approximated computational times for each inverse scattering algorithm as n increases.

5.2.3 Random Steps

1. Fig. 5.31 shows the wave propagation for the Gaussian source wavelet in random steps at different times. An animation of the wave propagation is available: Gaussian Random Steps.

2. Fig. 5.32 shows the Gaussian scattering, inverse scattering plots, and Table 5.7 shows their corresponding tables of relative errors.

3. Fig. 5.33 shows data downsampling using Gaussian measured data.

4. Fig. 5.34 shows the wave propagation for the Ricker source wavelet in random steps at different times. An animation of the wave propagation is available: Ricker Random Steps.

5. Fig. 5.35 shows the Ricker scattering, inverse scattering plots, and Table 5.8 shows their corresponding tables of relative errors.

6. Fig. 5.36 shows data downsampling using Ricker measured data.

7. Fig. 5.37 shows the wave propagation for the Dirac source wavelet in random steps at different times. An animation of the wave propagation is available: Dirac Random Steps.

8. Fig. 5.38 shows the Dirac scattering, inverse scattering plots, and Table 5.9 shows their corresponding tables of relative errors.

9. Fig. 5.39 shows the Dirac inverse scattering when there is Gaussian noisy data for the classical Born approximation.

10. Fig. 5.40 shows the Dirac inverse scattering when there is uniform noisy data for the classical Born approximation.

11. Fig. 5.41 shows the Dirac inverse scattering when there is Gaussian noisy data for the refined impedance transform.

12. Fig. 5.42 shows the Dirac inverse scattering when there is uniform noisy data for the refined impedance transform.

13. Fig. 5.43 shows the Dirac inverse scattering when there is Gaussian noisy data for the echoes-to-impedance transform.

14. Fig. 5.44 shows the Dirac inverse scattering when there is uniform noisy data for the echoes-to-impedance transform.

15. Table 5.15a gives a table of the l_2 relative errors for each inverse scattering algorithm as n increases.

16. Table 5.15b gives a table of approximated computational times for each inverse scattering algorithm as n increases.

5.2.4 Discontinuous Toothed Ramp

1. Fig. 5.45 shows the wave propagation for the Gaussian source wavelet in a discontinuous toothed ramp at different times. An animation of the wave propagation is available: Gaussian Discontinuous Toothed Ramp.

2. Fig. 5.46 shows the Gaussian scattering, inverse scattering plots, and Table 5.10 shows their corresponding tables of relative errors.

3. Fig. 5.47 shows data downsampling using Gaussian measured data.

4. Fig. 5.48 shows the wave propagation for the Ricker source wavelet in a discontinuous toothed ramp at different times. An animation of the wave propagation is available: Ricker Discontinuous Toothed Ramp.

5. Fig. 5.49 shows the Ricker scattering, inverse scattering plots, and Table 5.11 shows their corresponding tables of relative errors.

6. Fig. 5.50 shows data downsampling using Ricker measured data.

7. Fig. 5.51 shows the wave propagation for the Dirac source wavelet in a discontinuous toothed ramp at different times. An animation of the wave propagation is available: Dirac Discontinuous Toothed Ramp.

8. Fig. 5.52 shows the Dirac scattering, inverse scattering plots, and Table 5.12 shows their corresponding tables of relative errors.

9. Fig. 5.53 shows the Dirac inverse scattering when there is Gaussian noisy data for the classical Born approximation.

10. Fig. 5.54 shows the Dirac inverse scattering when there is uniform noisy data for the classical Born approximation.

11. Fig. 5.55 shows the Dirac inverse scattering when there is Gaussian noisy data for the refined impedance transform.

12. Fig. 5.56 shows the Dirac inverse scattering when there is uniform noisy data for the refined impedance transform.

13. Fig. 5.57 shows the Dirac inverse scattering when there is Gaussian noisy data for the echoes-to-impedance transform.

14. Fig. 5.58 shows the Dirac inverse scattering when there is uniform noisy data for the echoes-to-impedance transform.

15. Table 5.16a gives a table of the l_2 relative errors for each inverse scattering algorithm as n increases.

16. Table 5.16b gives a table of approximated computational times for each inverse scattering algorithm as n increases.

5.3 Figures and Tables

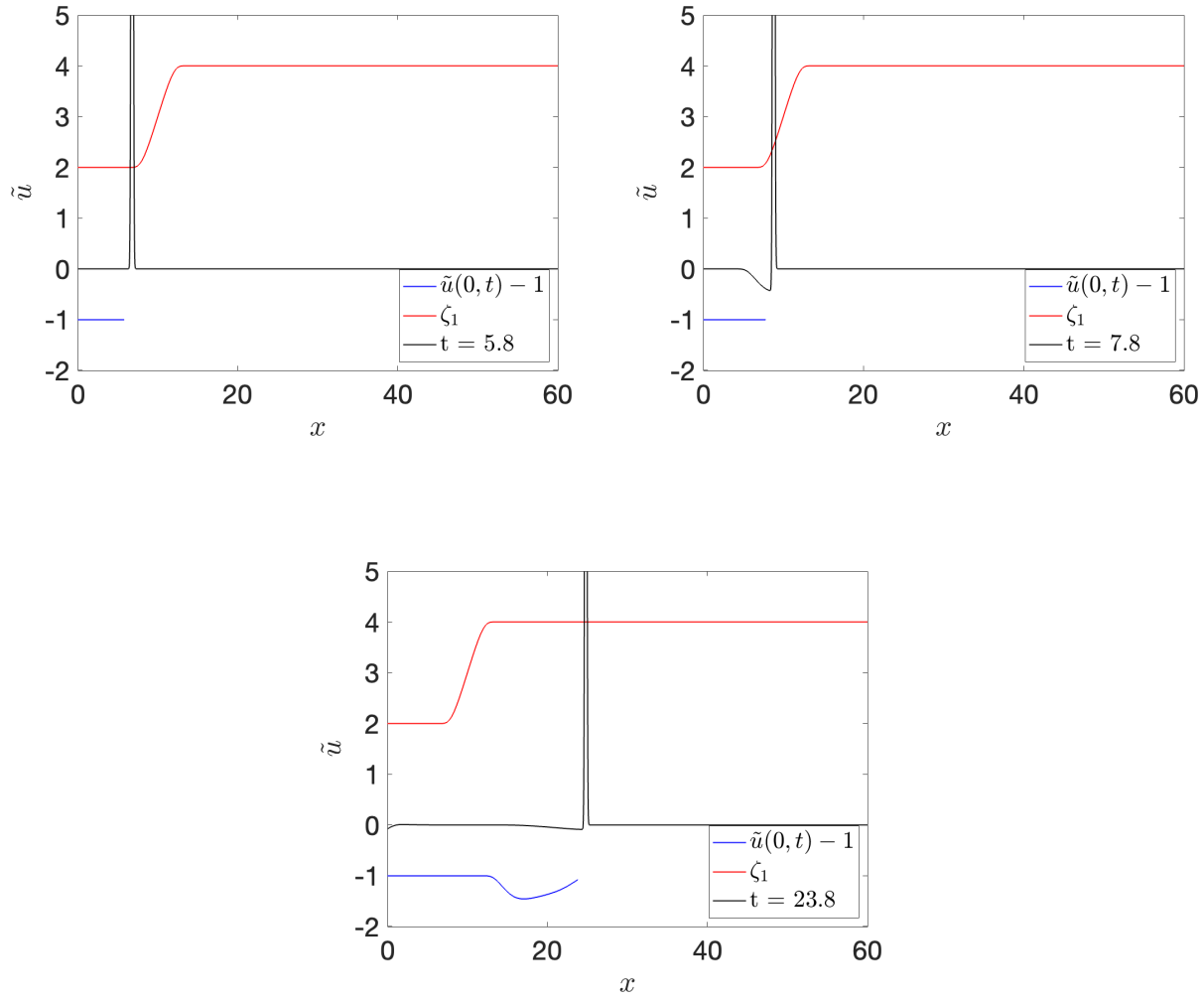
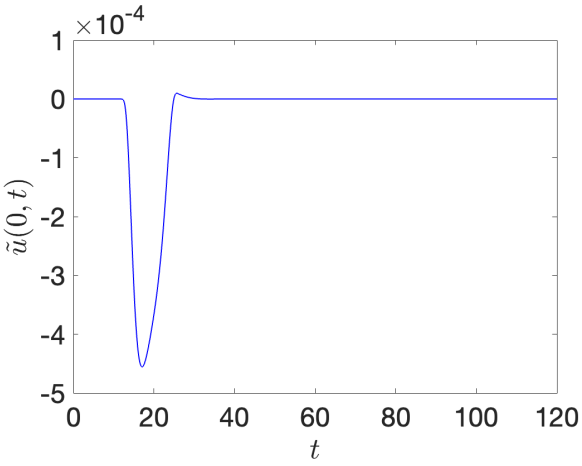
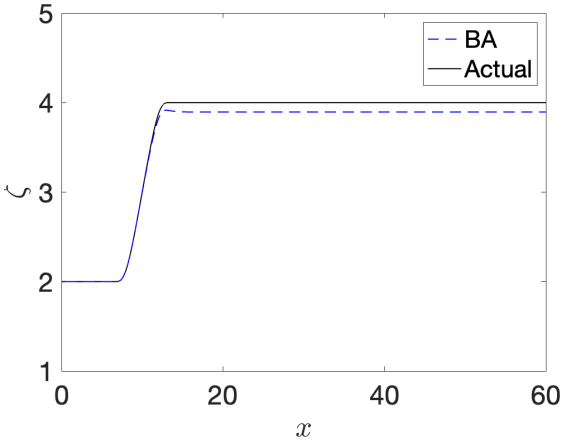


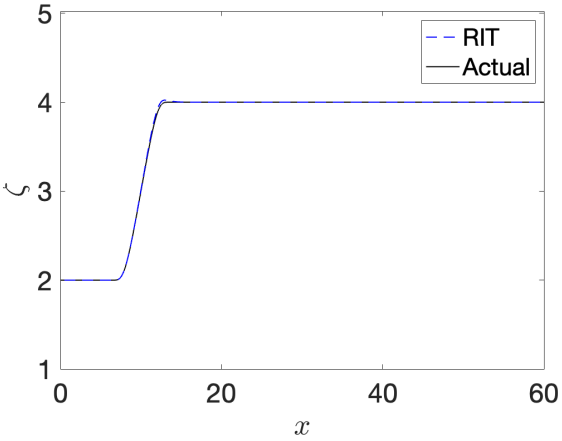
FIGURE 5.1: Scale Factor: 1000; Top left: The Gaussian source wavelet before passing through varying reflectivities. Top right: The Gaussian source wavelet passing through varying reflectivities. Bottom: The Gaussian source wavelet after passing through varying reflectivies.



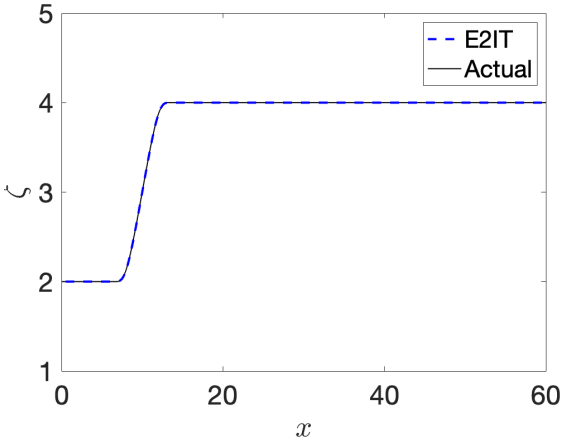
(A) Gaussian measured data



(B) Born approximation



(C) Refined impedance transform

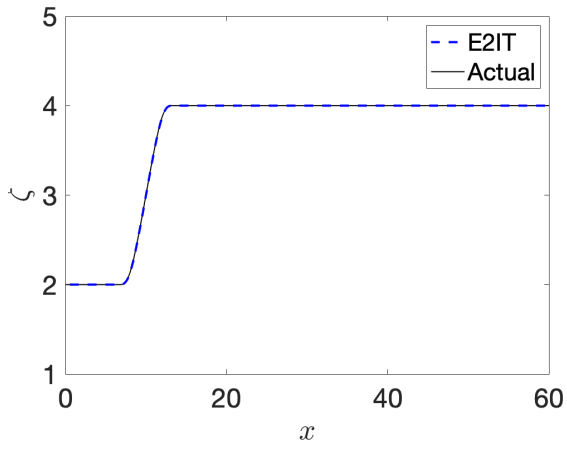


(D) Echoes-to-impedance transform

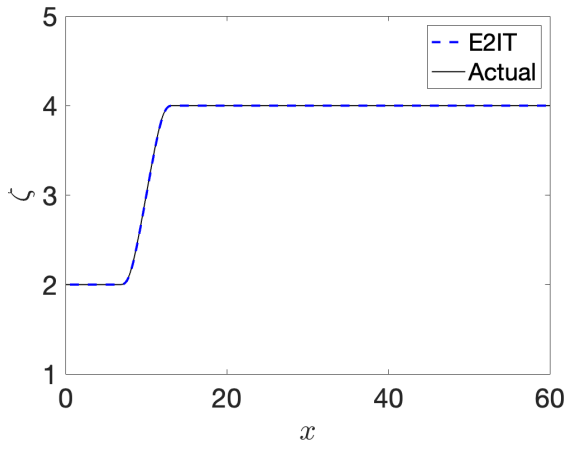
FIGURE 5.2: Scattering and Inverse Scattering of Gaussian in Smooth Ramp

	l_2 Errors
BA	0.0287
RIT	0.0170
E2IT	0.0154

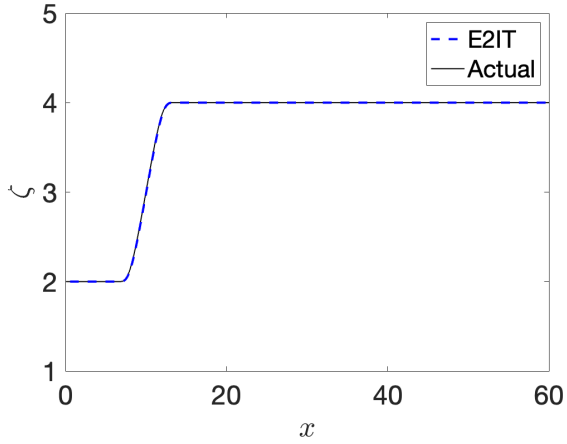
TABLE 5.1: l_2 Relative Errors for Gaussian in Smooth Ramp



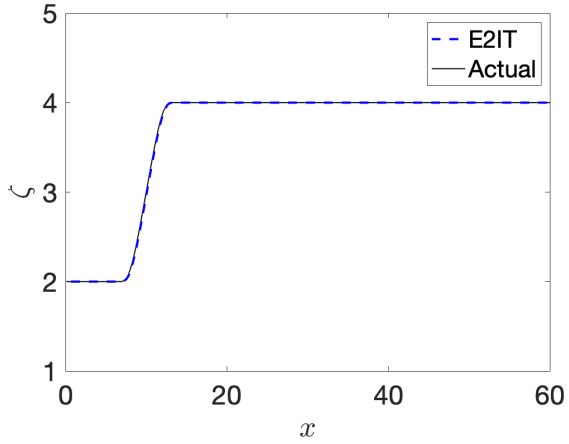
(A) Every 2nd point



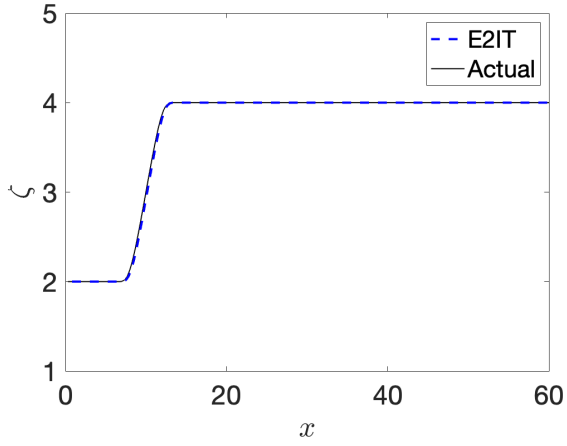
(B) Every 4th point



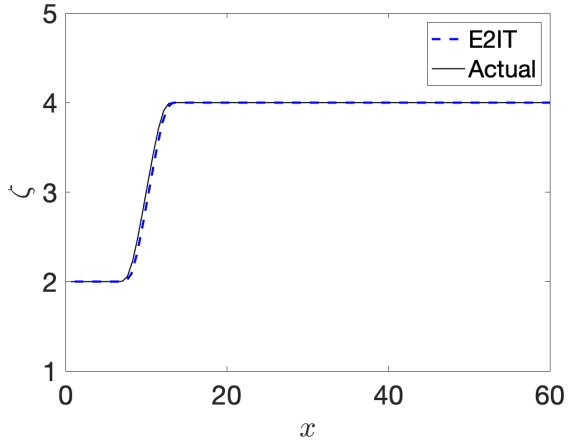
(C) Every 8th point



(D) Every 16th point



(E) Every 32th points



(F) Every 64th points

FIGURE 5.3: Data Downsampling on Gaussian Inverse Scattering for Smooth Ramp. Limit: Every 6825th point

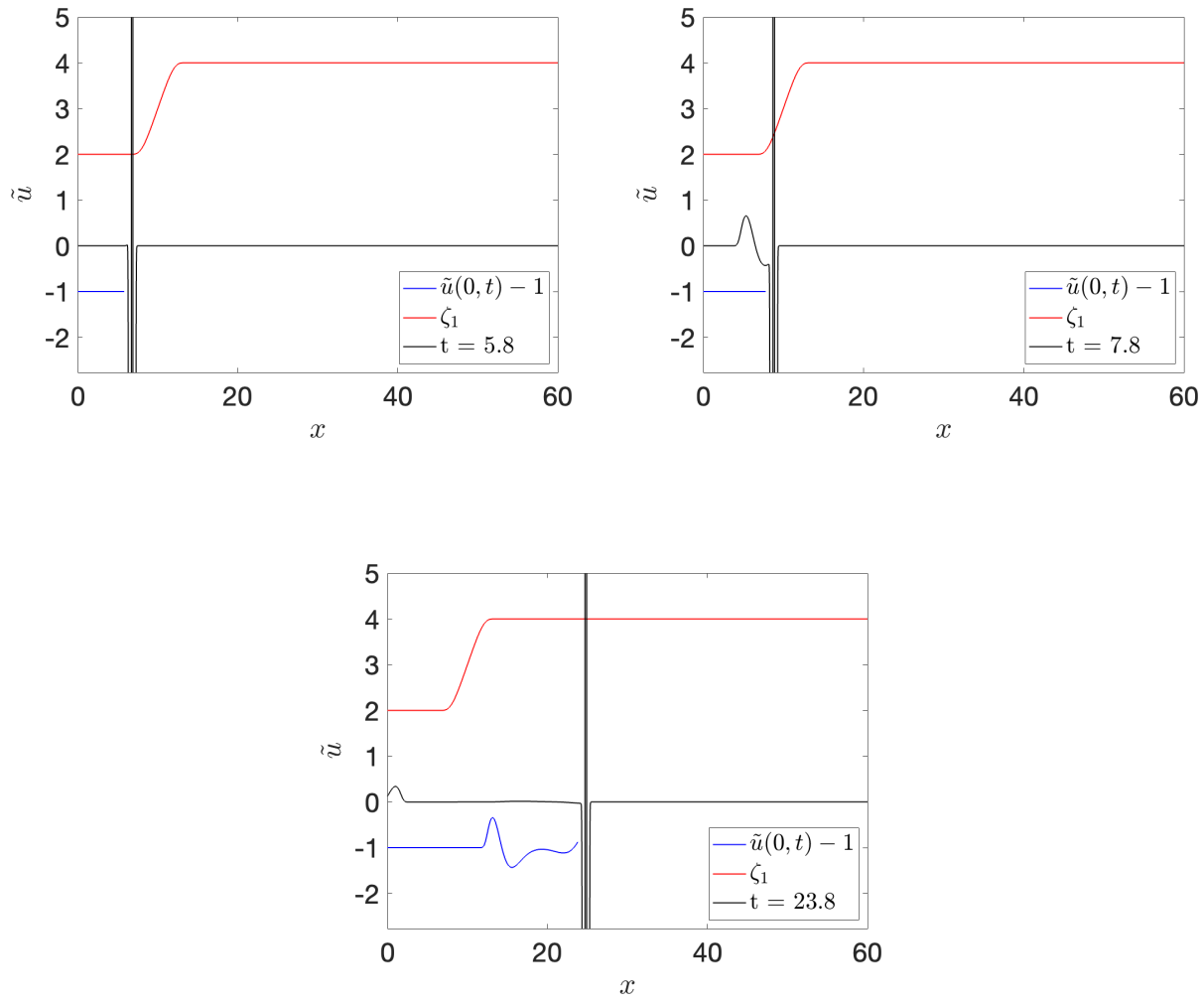
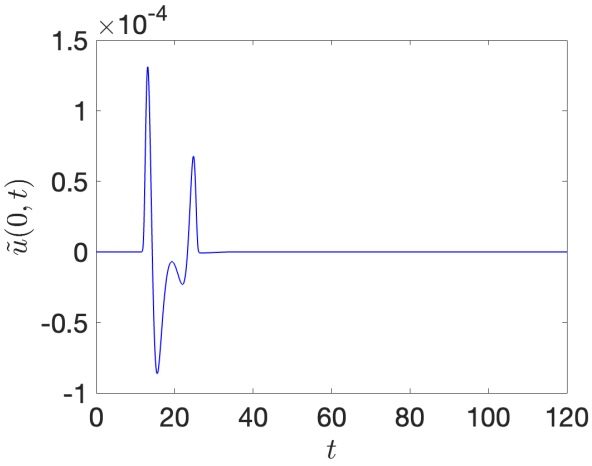
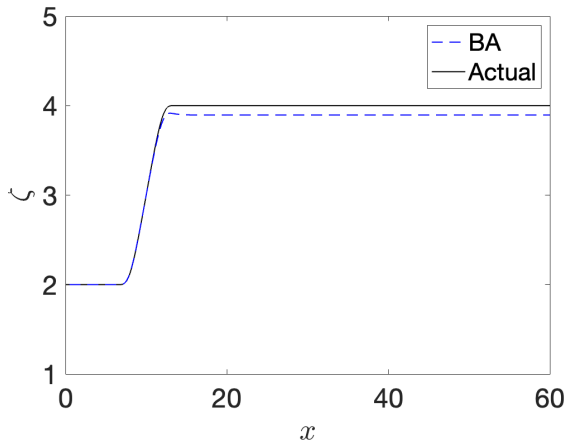


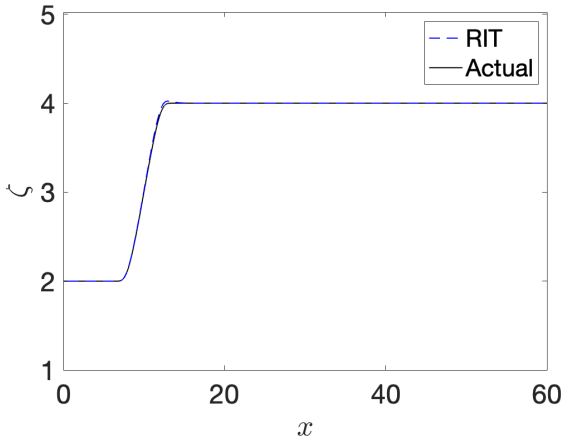
FIGURE 5.4: Scale Factor: 5000; Top left: The Ricker source wavelet before passing through varying reflectivities. Top right: The Ricker source wavelet passing through varying reflectivities. Bottom: The Ricker source wavelet after passing through varying reflectivies.



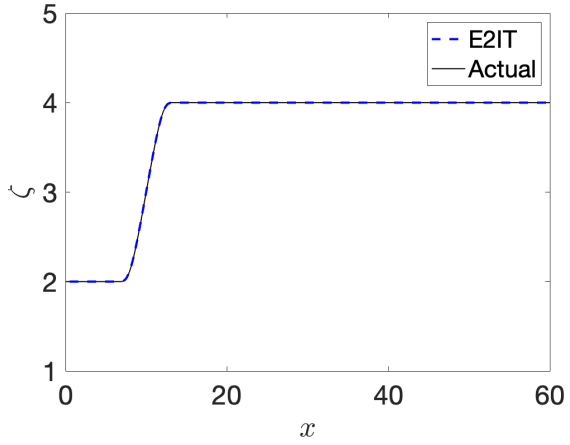
(A) Ricker measured data



(B) Born approximation



(C) Refined impedance transform

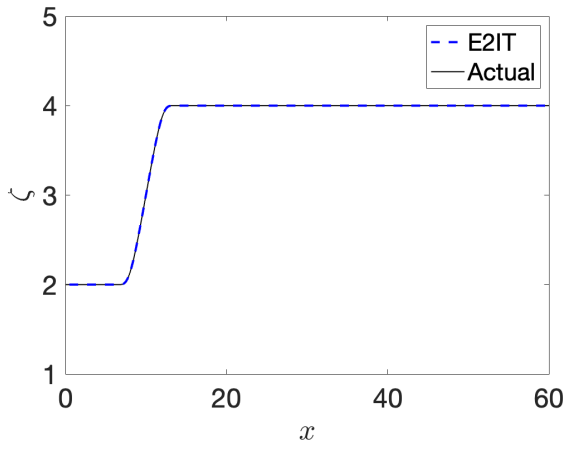


(D) Echoes-to-impedance transform

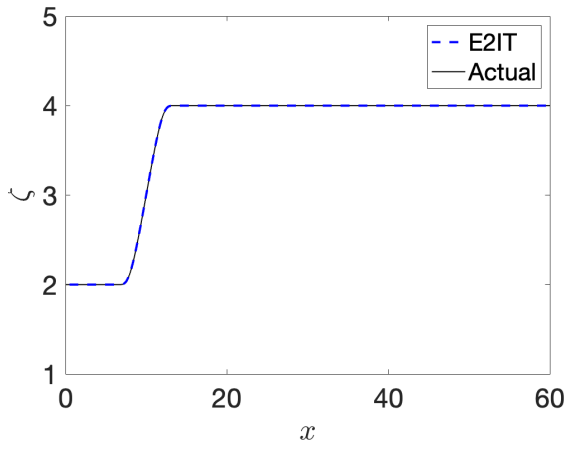
FIGURE 5.5: Scattering and Inverse Scattering of Ricker in Smooth Ramp

	l_2 Errors
BA	0.0288
RIT	0.0173
E2IT	0.0157

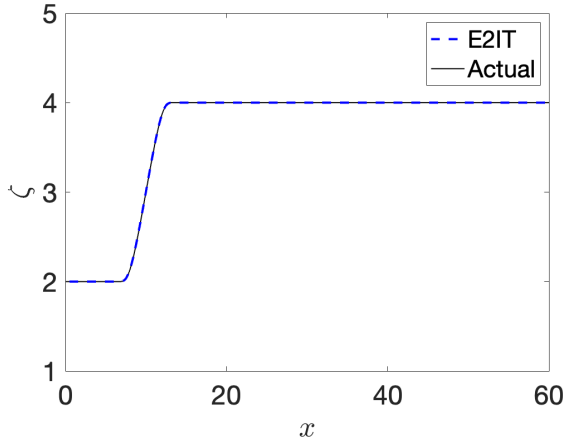
TABLE 5.2: l_2 Relative Errors of Ricker in Smooth Ramp



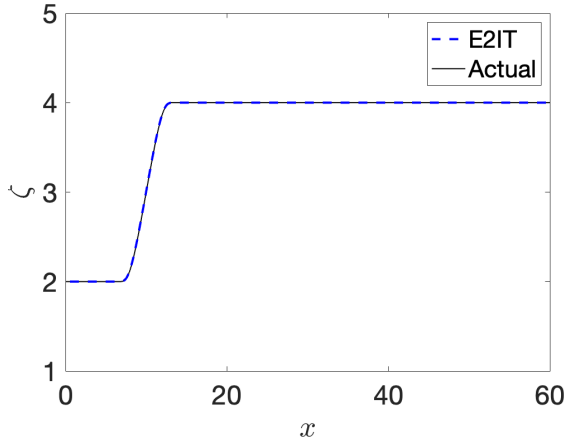
(A) Every 2nd point



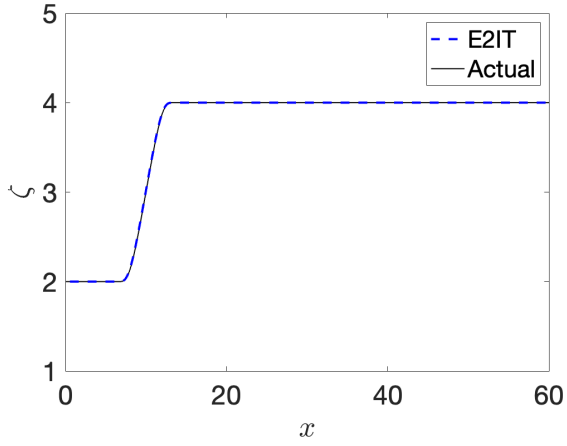
(B) Every 4th point



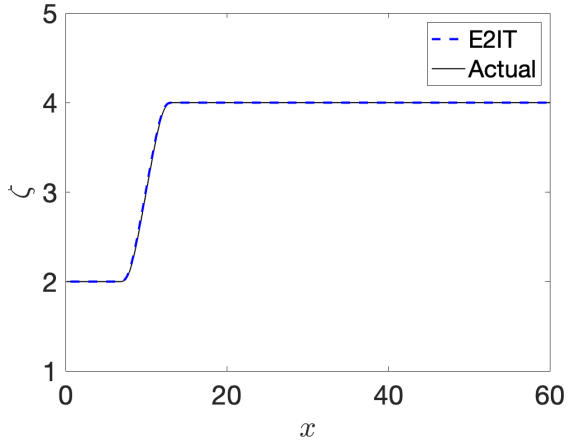
(C) Every 8th point



(D) Every 16th point



(E) Every 32th points



(F) Every 64th points

FIGURE 5.6: Data Downsampling on Ricker Inverse Scattering for Smooth Ramp. Limit: Every 256th point

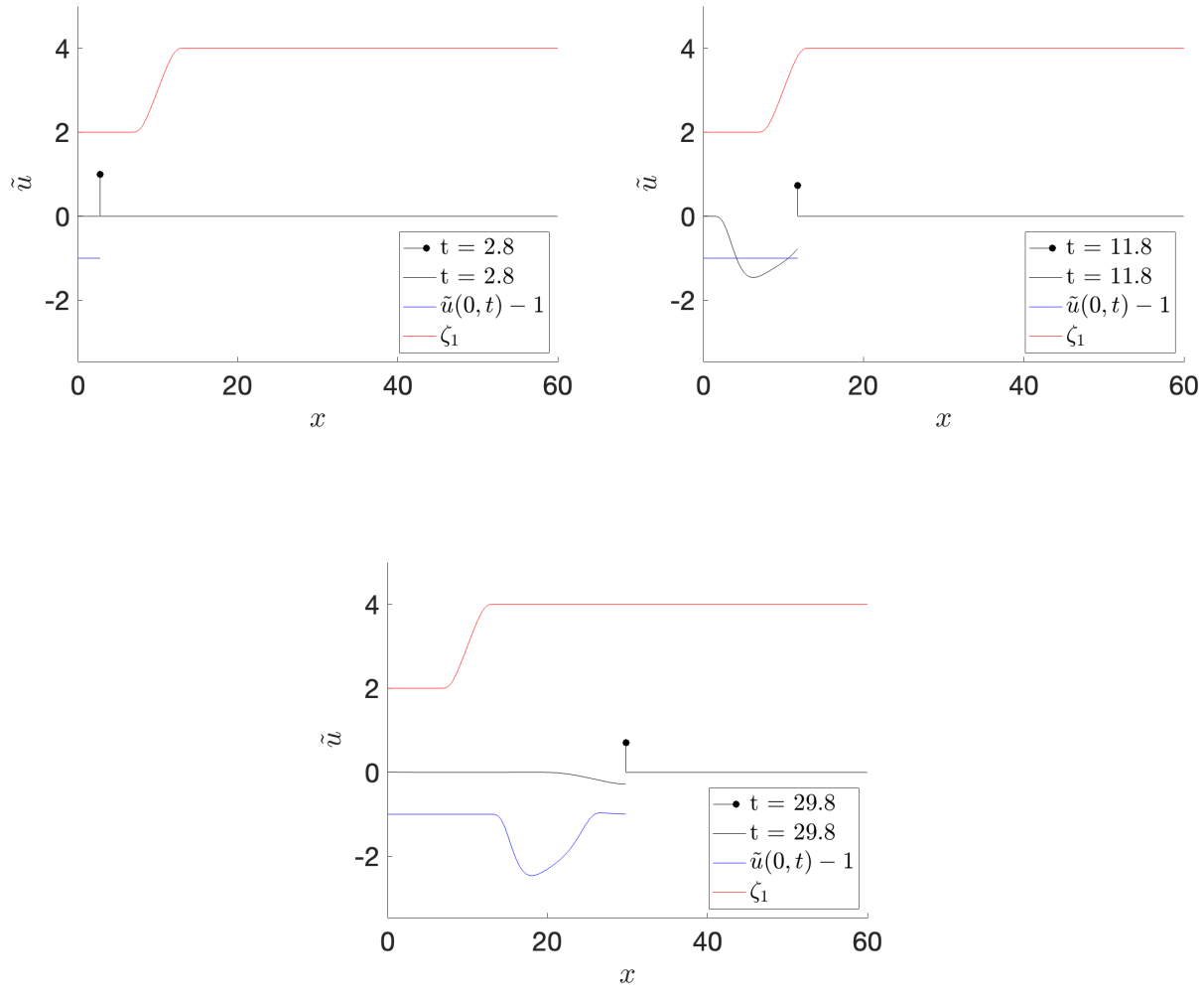
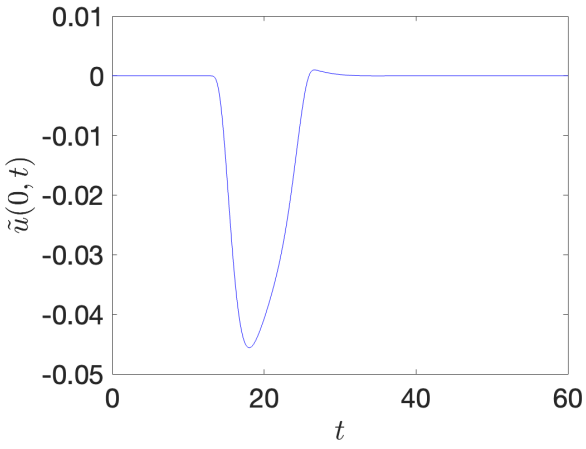
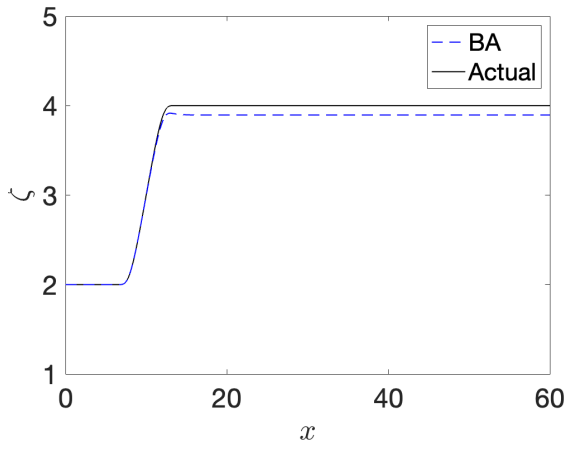


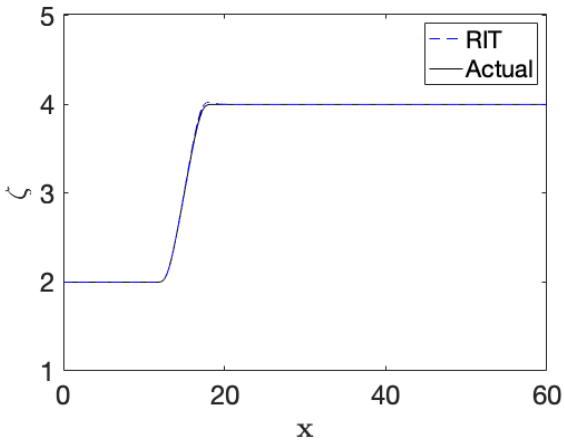
FIGURE 5.7: Scale Factor: 32; Top left: The Dirac source wavelet before passing through varying reflectivities. Top right: The Dirac source wavelet passing through varying reflectivities. Bottom: The Dirac source wavelet after passing through varying reflectivities.



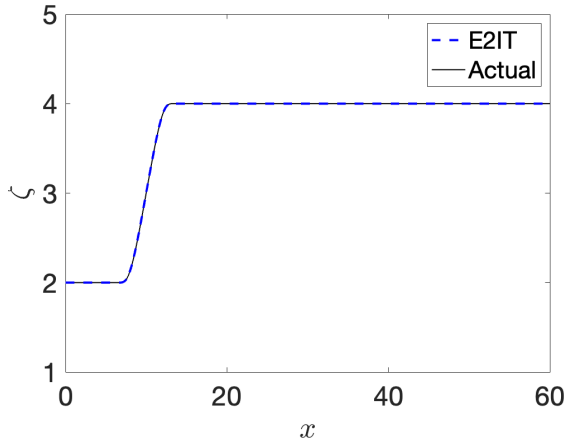
(A) Dirac measured data



(B) Born approximation



(C) Refined impedance transform

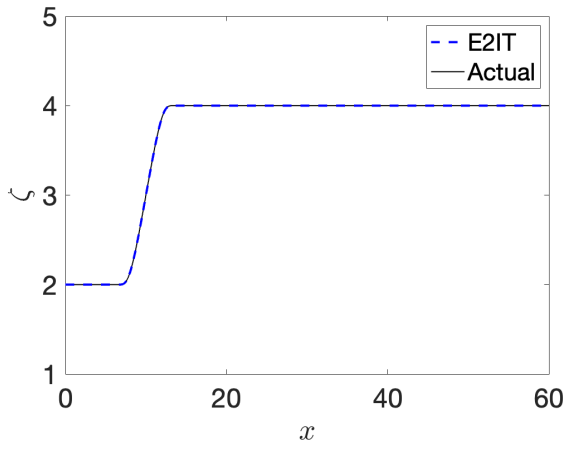


(D) Echoes-to-impedance transform

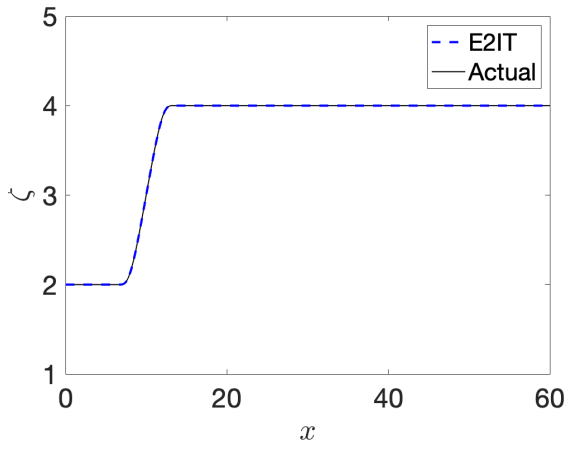
FIGURE 5.8: Scattering and Inverse Scattering of Dirac in Smooth Ramp

	l_2 Errors
BA	0.0248
RIT	0.0021
E2IT	**2.2118E-4**

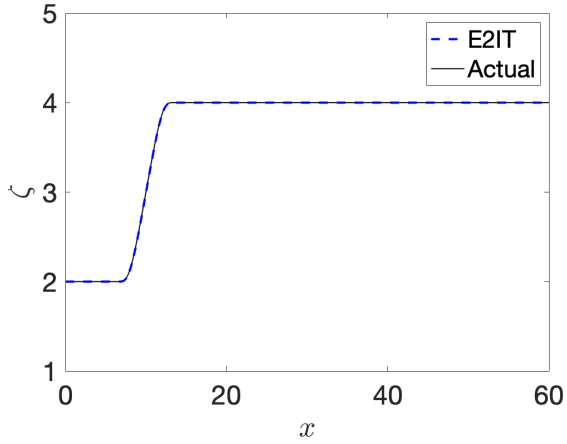
TABLE 5.3: l_2 Relative Errors for Dirac in Smooth Ramp



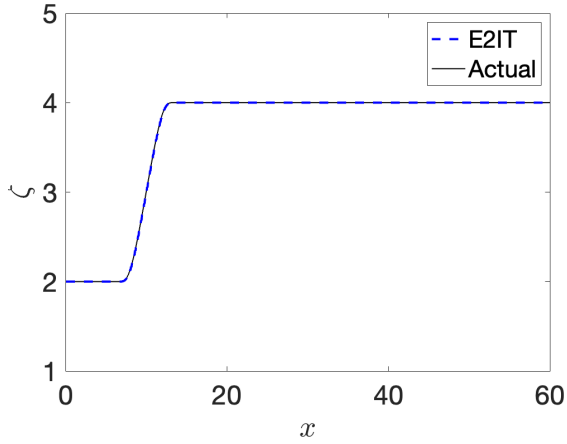
(A) Every 2nd point



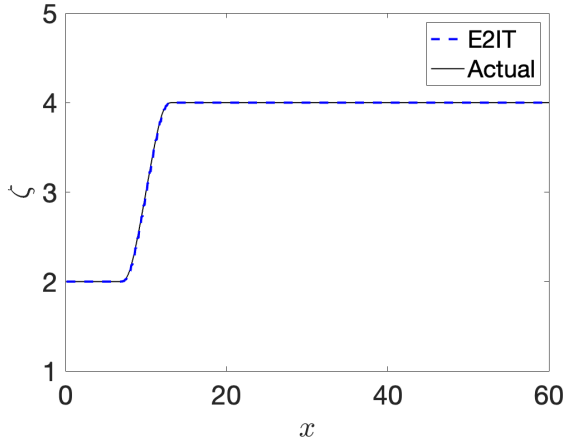
(B) Every 4th point



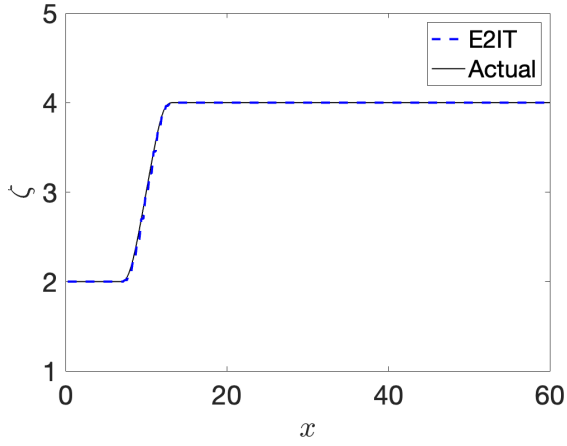
(C) Every 8th point



(D) Every 16th point

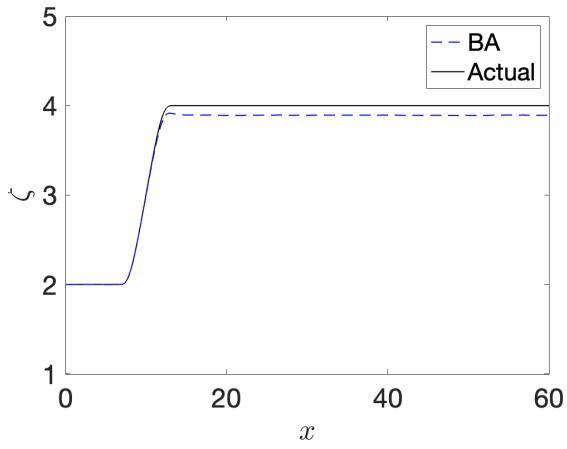


(E) Every 32th points

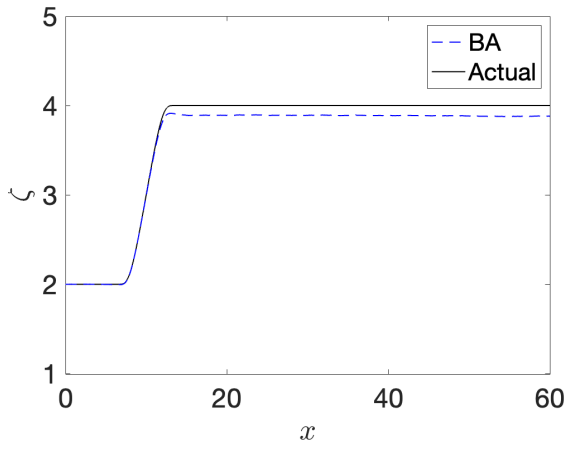


(F) Every 64th points

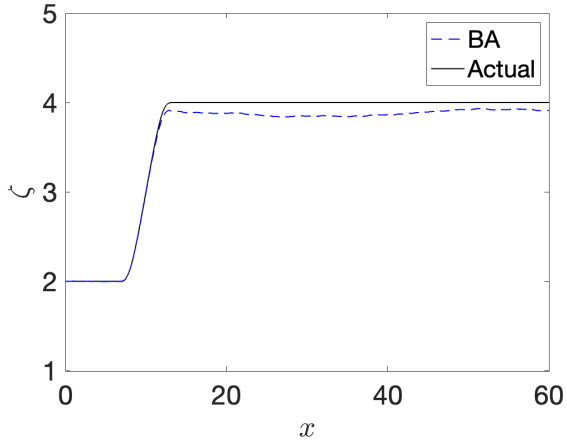
FIGURE 5.9: Data Downsampling on Dirac Inverse Scattering for Smooth Ramp. Limit: NaN. The plot of ζ and $\tilde{\zeta}$ eventually disappear.



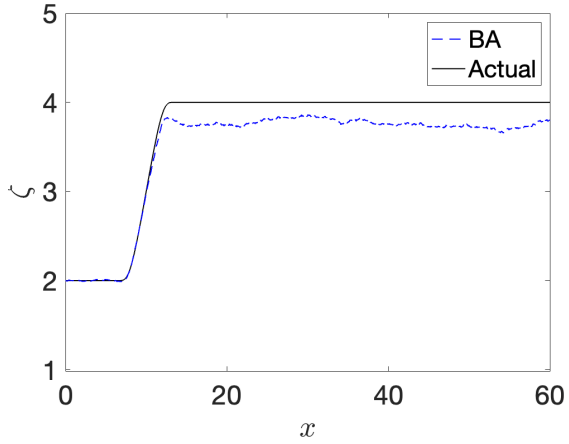
5% Gaussian Noise



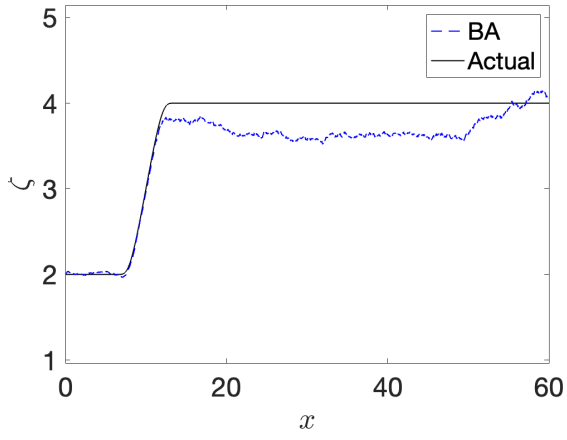
10% Gaussian Noise



25% Gaussian Noise



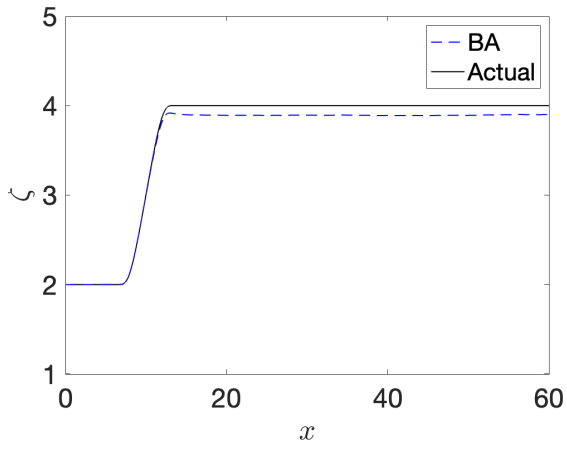
100% Gaussian Noise



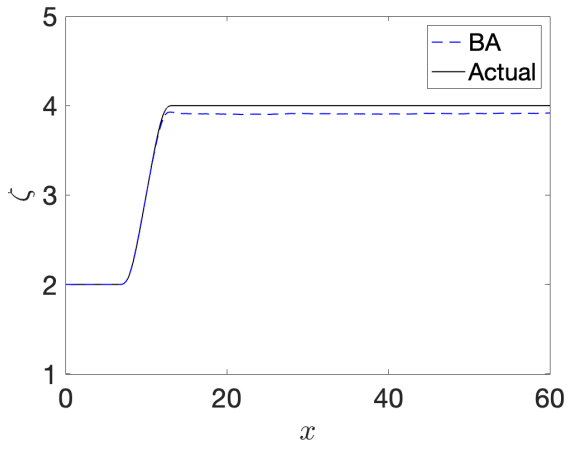
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0244
10	0.0267
25	0.0289
100	0.0575
200	0.0772

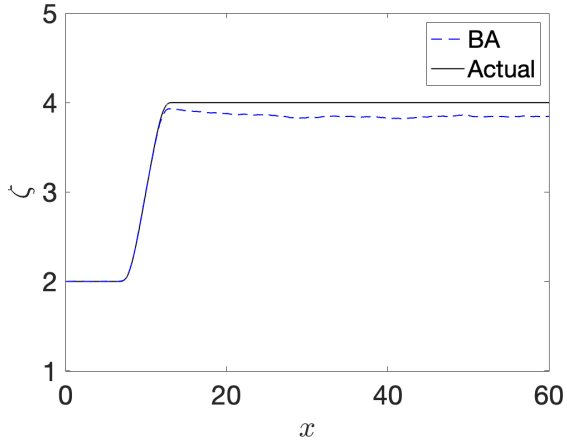
FIGURE 5.10: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors



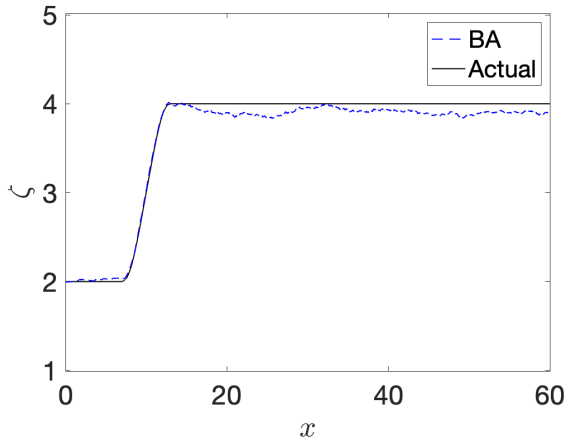
5% Uniform Noise



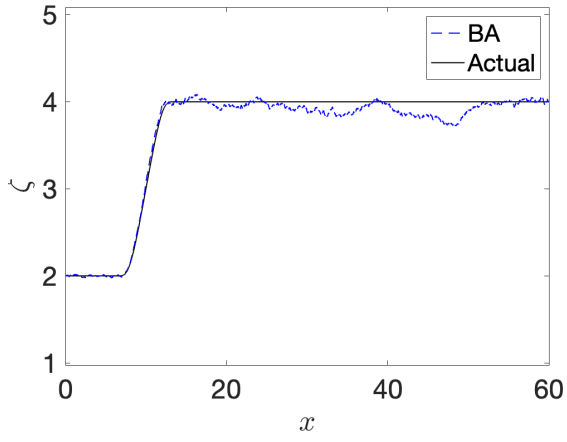
10% Uniform Noise



25% Uniform Noise



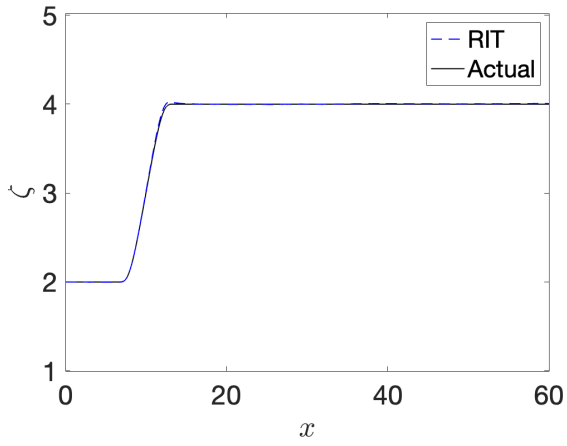
100% Uniform Noise



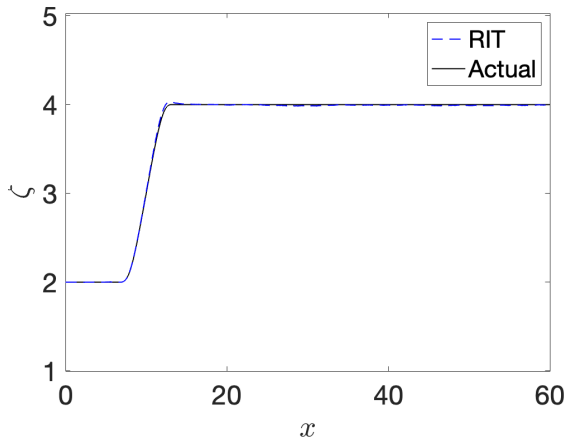
200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0253
10	0.0217
25	0.0354
100	0.0230
200	0.0247

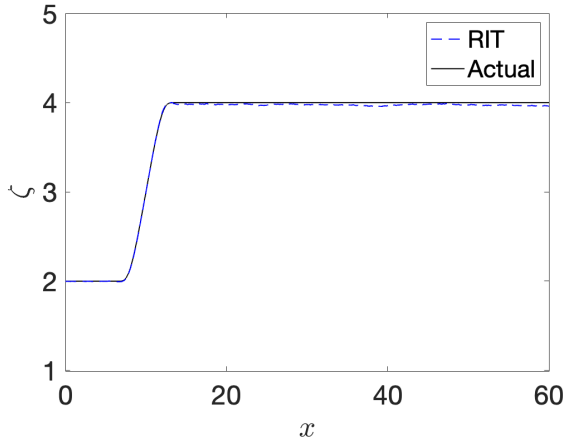
FIGURE 5.11: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



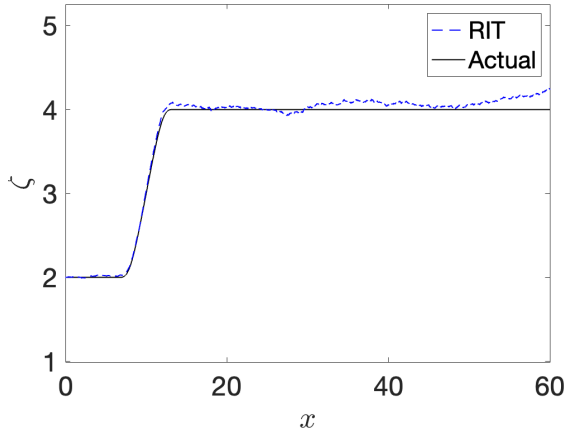
5% Gaussian Noise



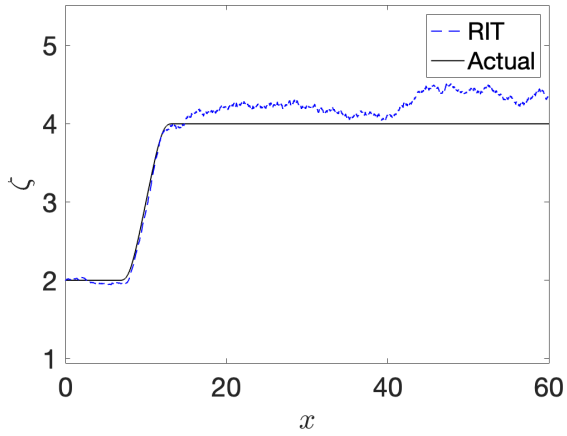
10% Gaussian Noise



25% Gaussian Noise



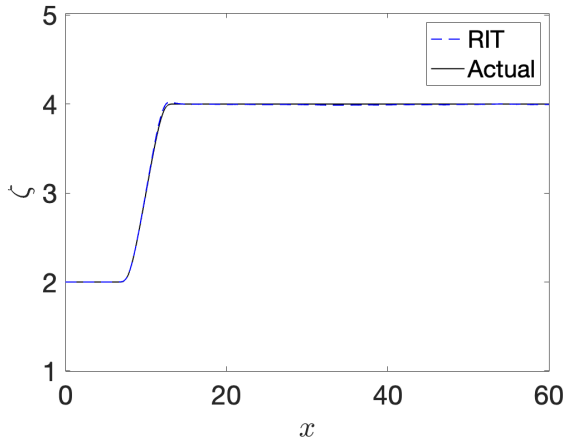
100% Gaussian Noise



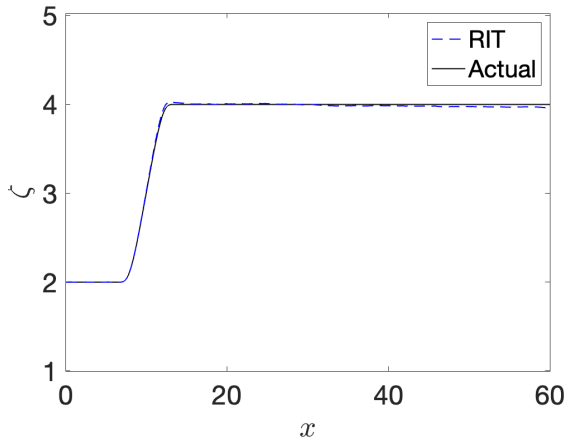
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0024
10	0.0267
25	0.0289
100	0.0575
200	0.0772

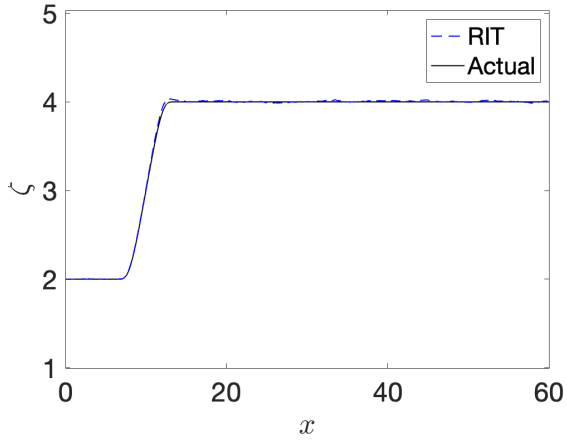
FIGURE 5.12: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



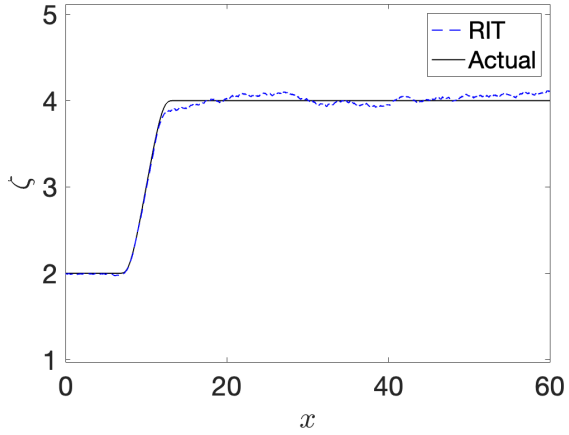
5% Uniform Noise



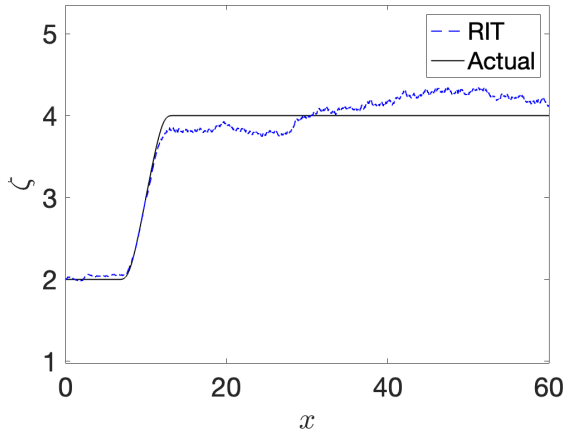
10% Uniform Noise



25% Uniform Noise



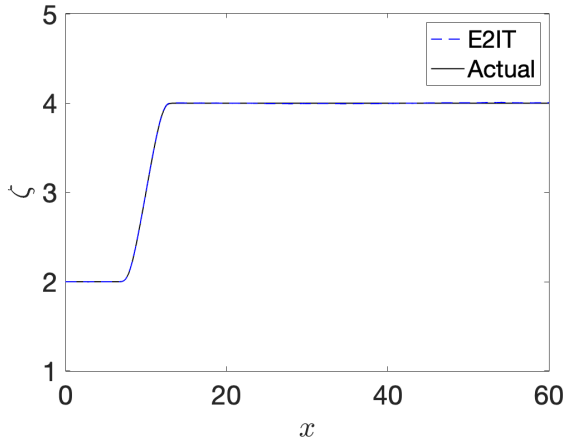
100% Uniform Noise



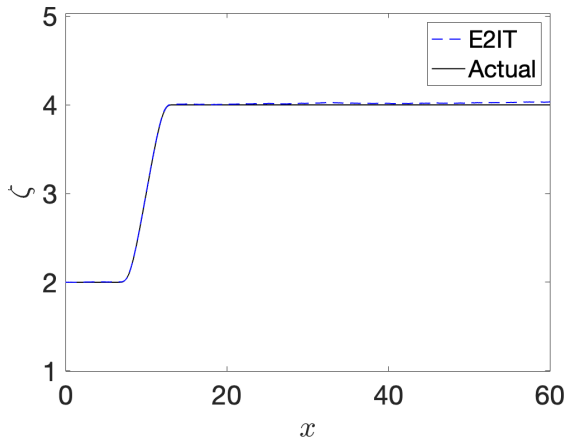
200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0026
10	0.0049
25	0.0037
100	0.0139
200	0.0484

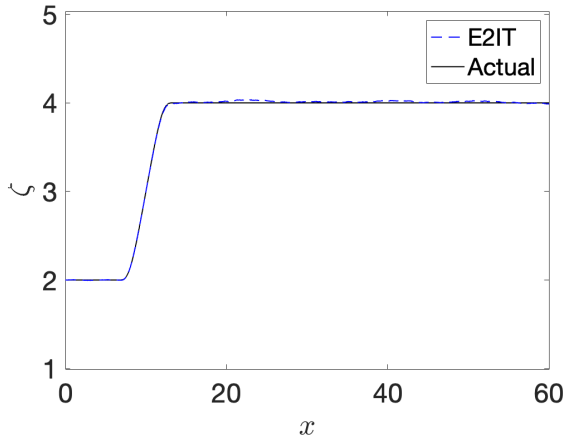
FIGURE 5.13: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



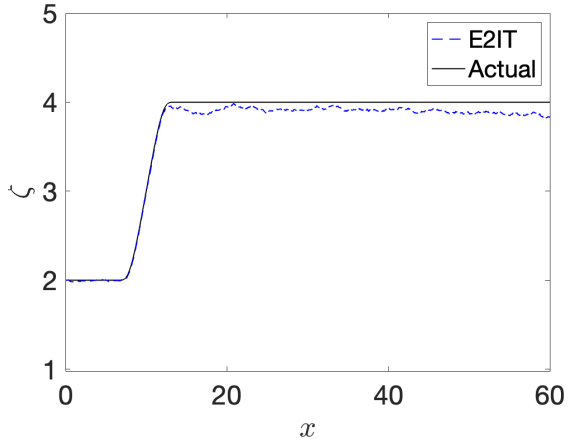
5% Gaussian Noise



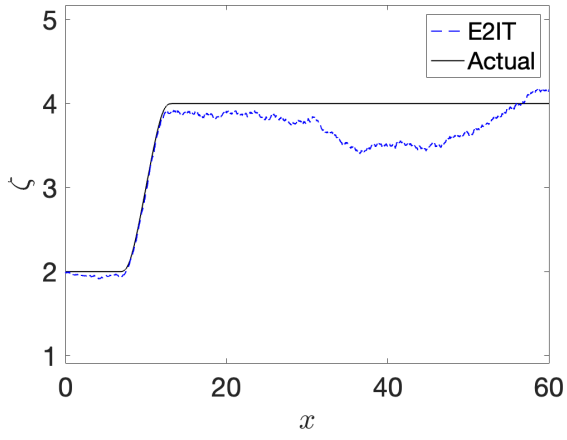
10% Gaussian Noise



25% Gaussian Noise



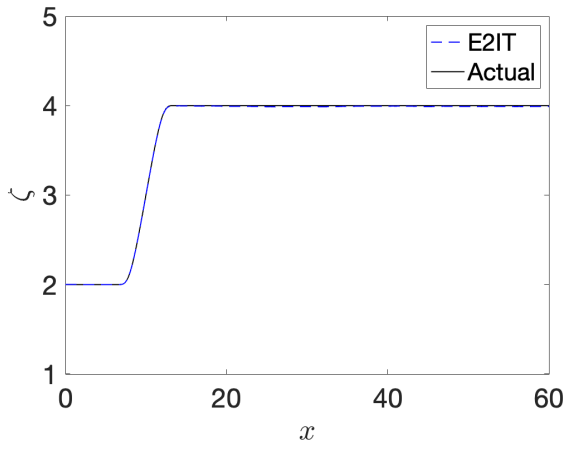
100% Gaussian Noise



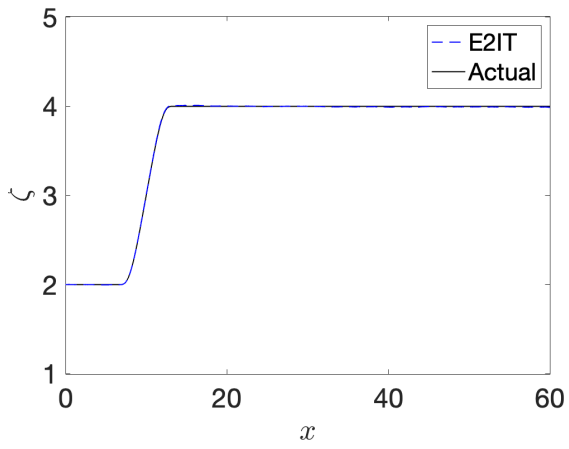
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0010
10	0.0044
25	0.0036
100	0.0239
200	0.0767

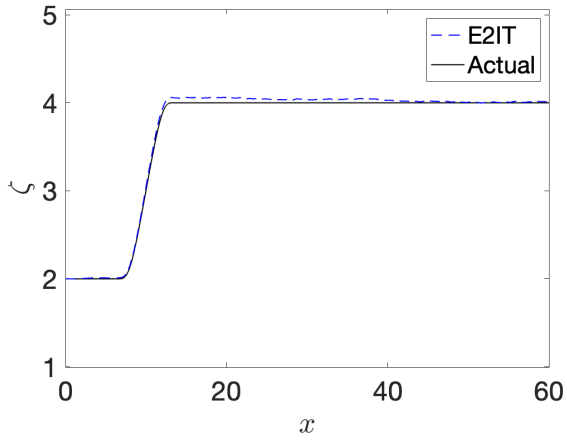
FIGURE 5.14: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



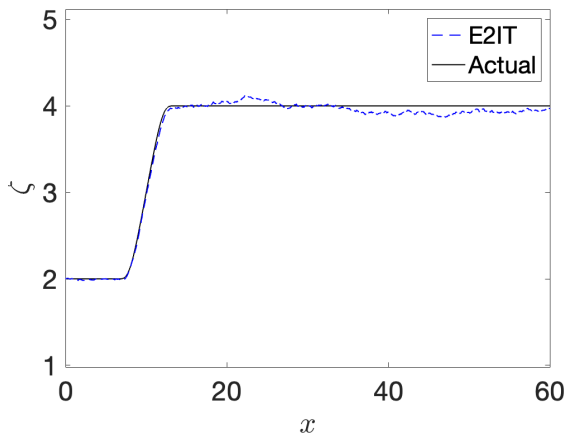
5% Uniform Noise



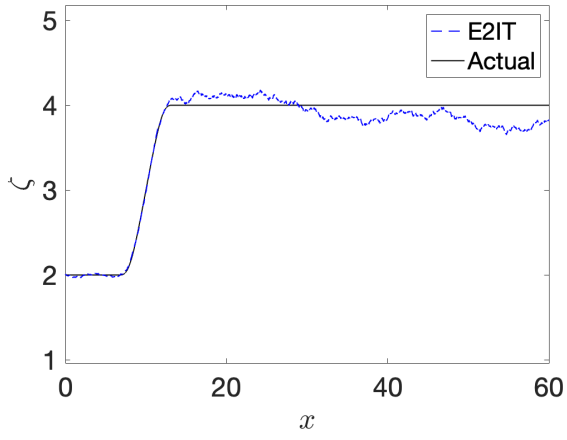
10% Uniform Noise



25% Uniform Noise



100% Uniform Noise



200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0021
10	0.0015
25	0.0094
100	0.0156
200	0.0379

FIGURE 5.15: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.

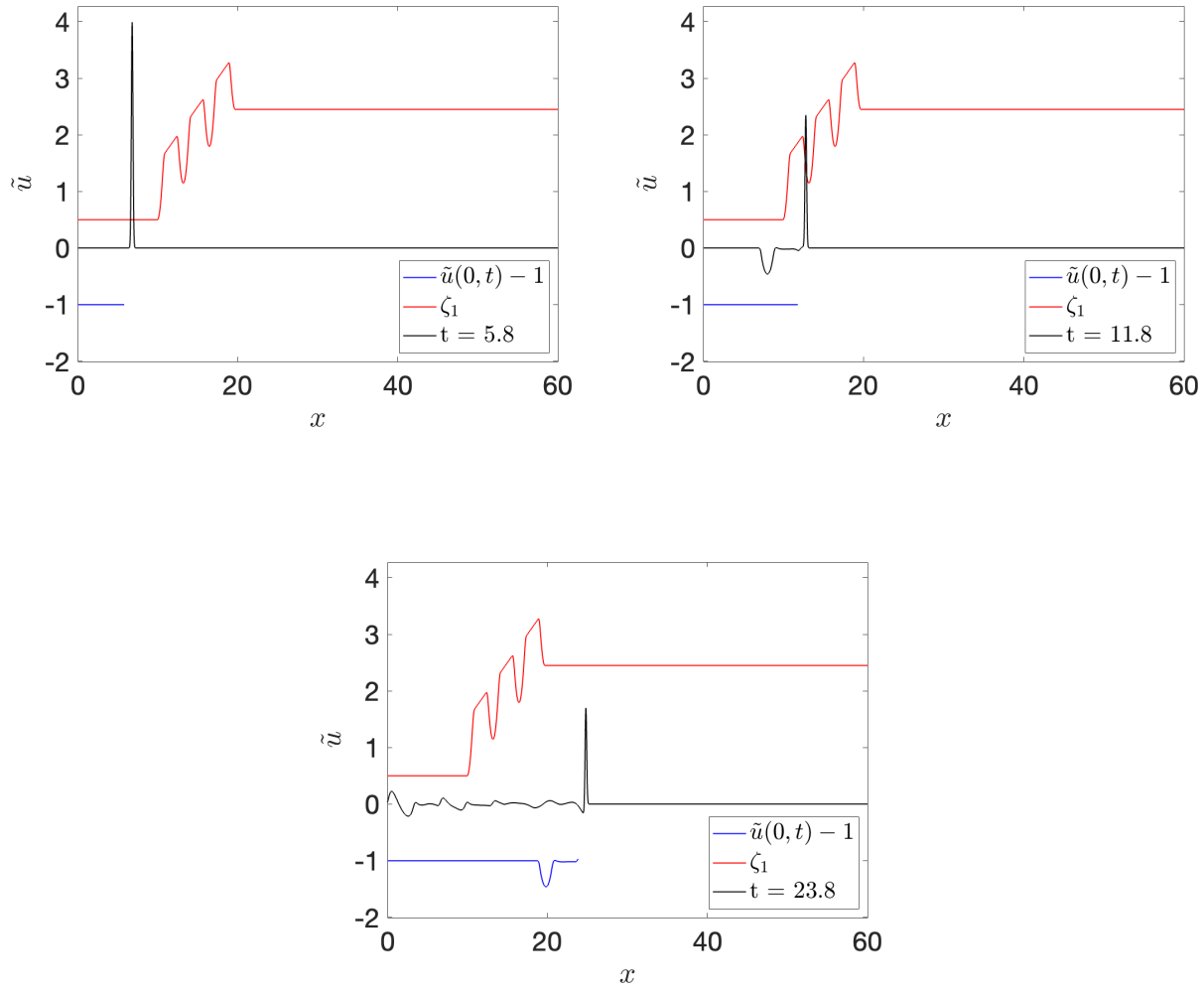
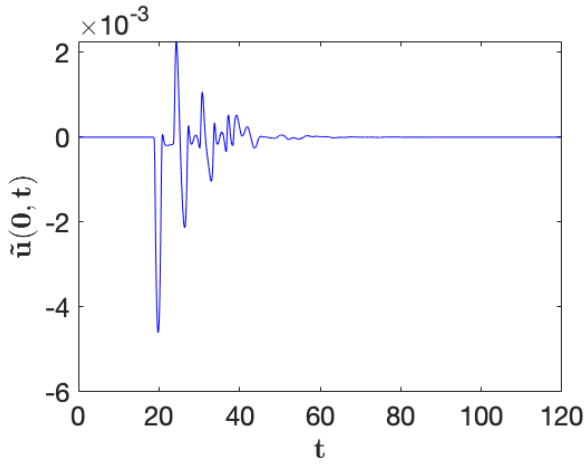
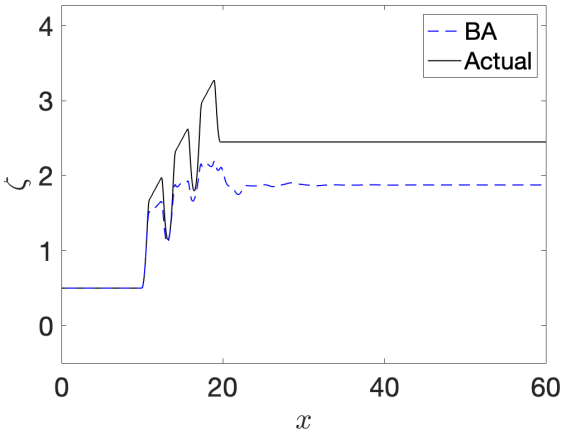


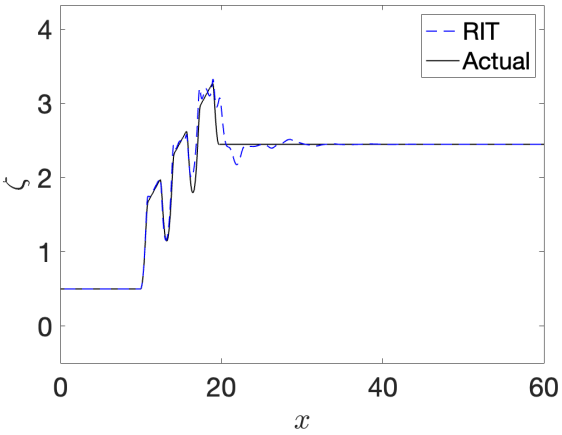
FIGURE 5.16: Scale Factor: 100; Top left: The Gaussian source wavelet before passing through varying reflectivities. Top right: The Gaussian source wavelet passing through varying reflectivities. Bottom: The Gaussian source wavelet after passing through varying reflectivities.



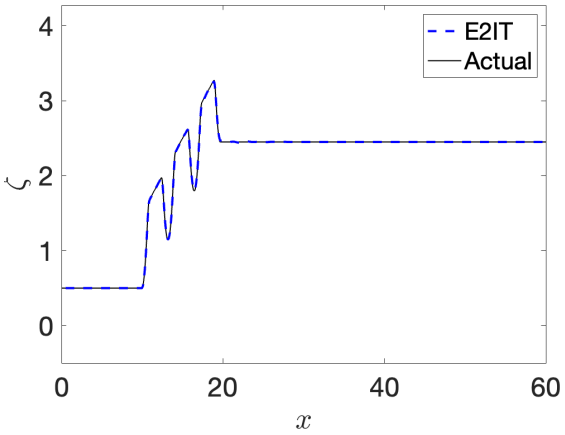
(A) Gaussian measured data



(B) Born approximation



(C) Refined impedance transform

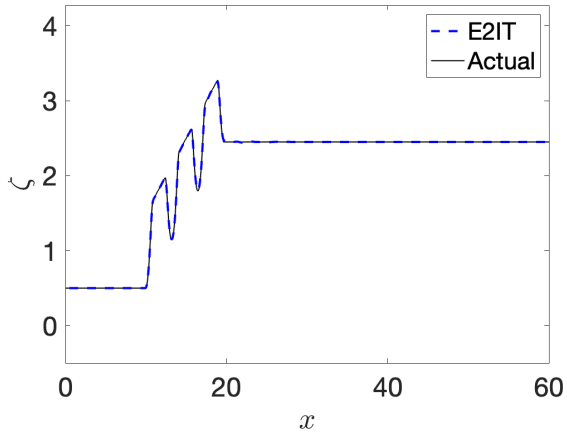


(D) Echoes-to-impedance transform

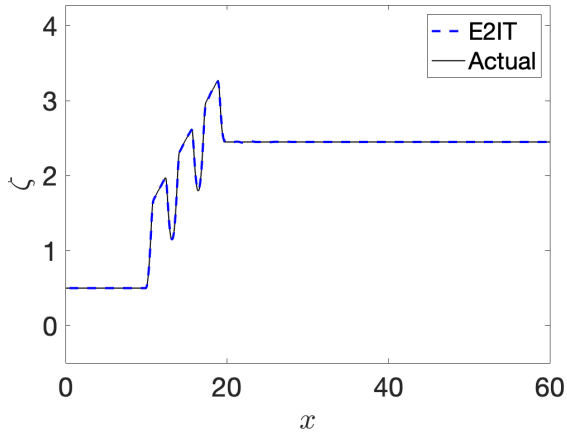
FIGURE 5.17: Scattering and Inverse Scattering of Gaussian in Toothed Ramp

	l_2 Errors
BA	0.2422
RIT	0.0901
E2IT	0.0767

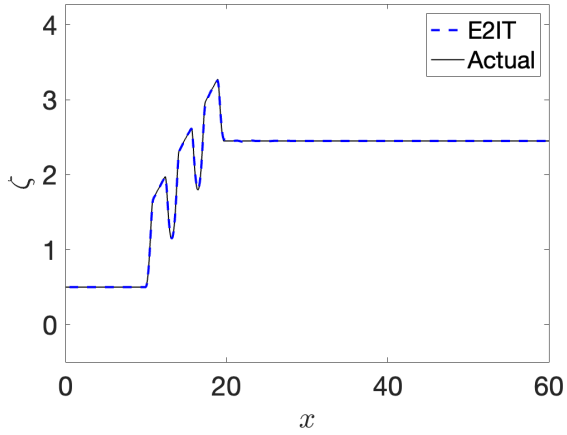
TABLE 5.4: l_2 Relative Errors for Gaussian in Toothed Ramp



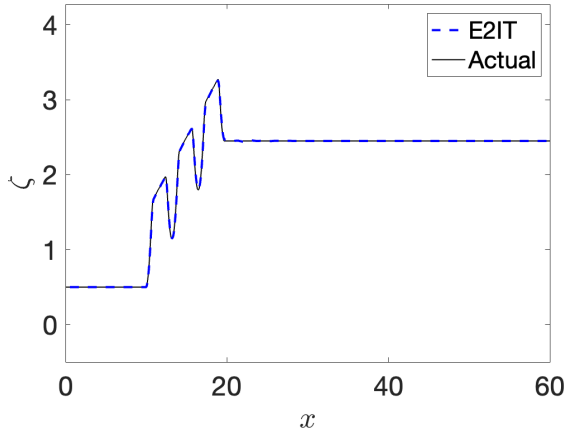
(A) Every 2nd point



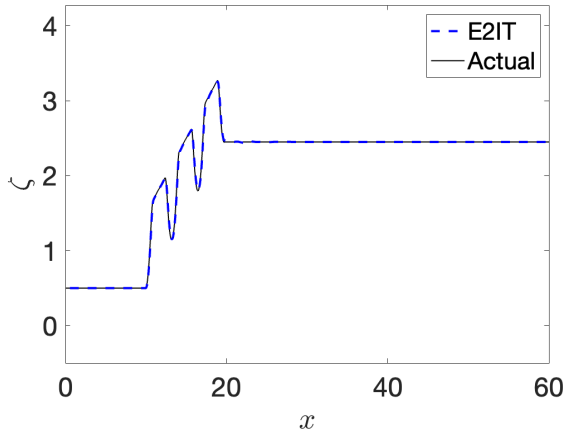
(B) Every 4th point



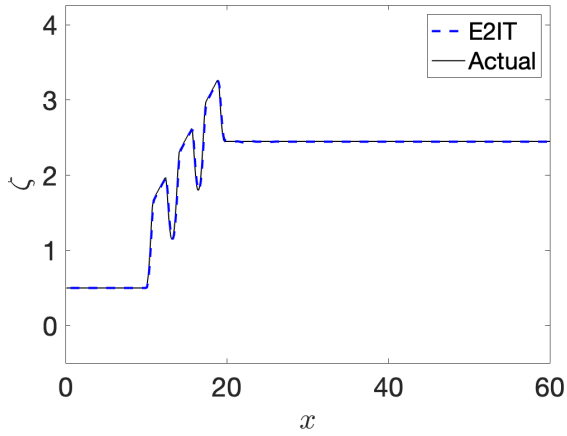
(C) Every 8th point



(D) Every 16th point



(E) Every 32th points



(F) Every 64th points

FIGURE 5.18: Data Downsampling on Gaussian Inverse Scattering for Toothed Ramp. Limit: Every 280th point

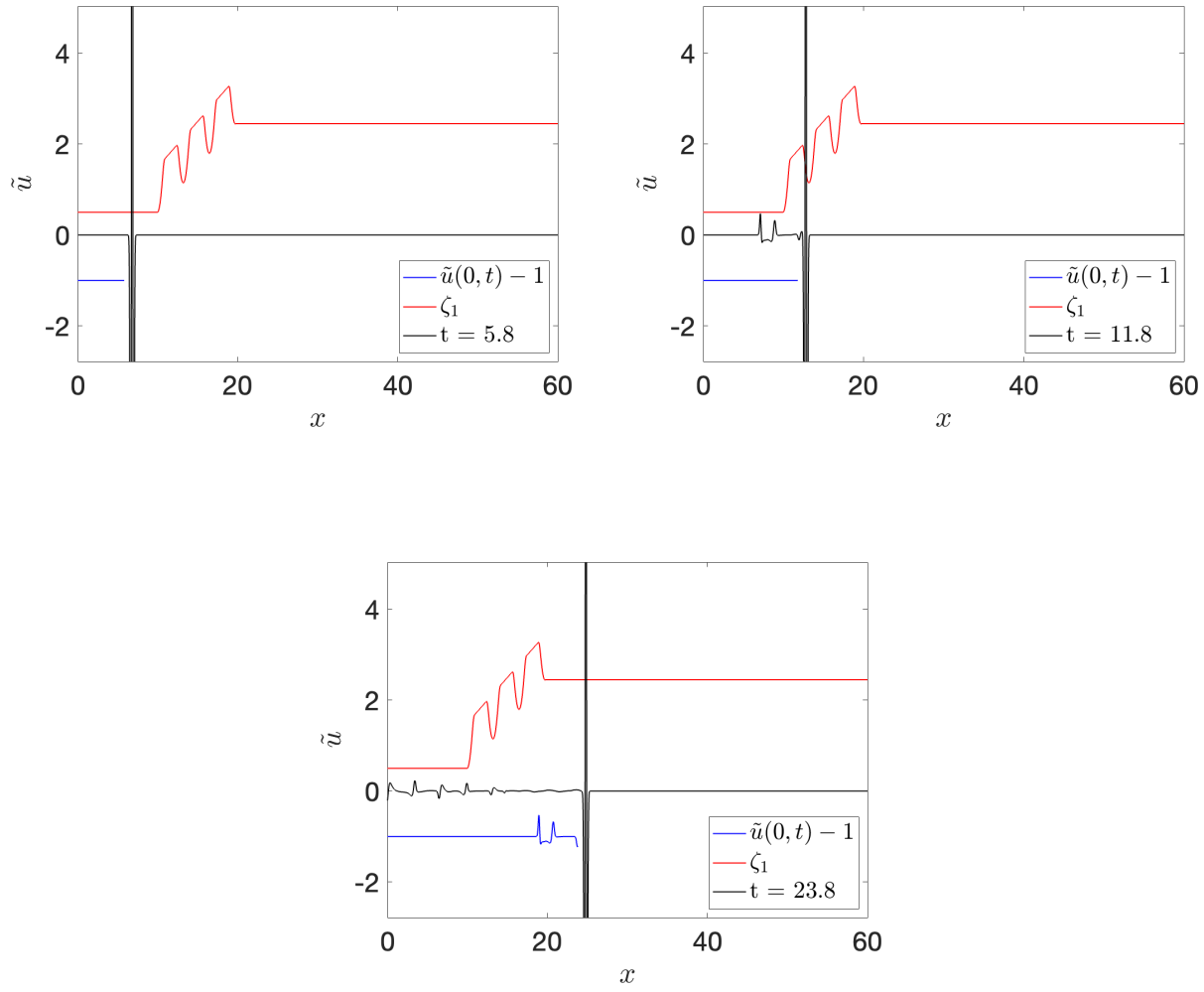
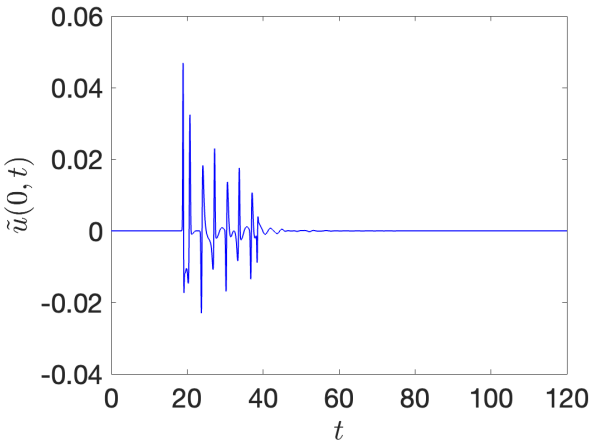
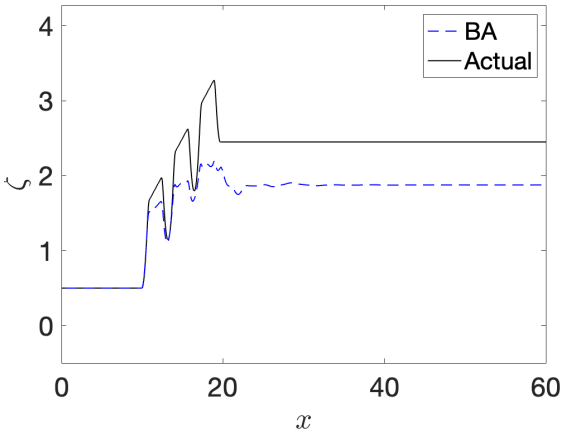


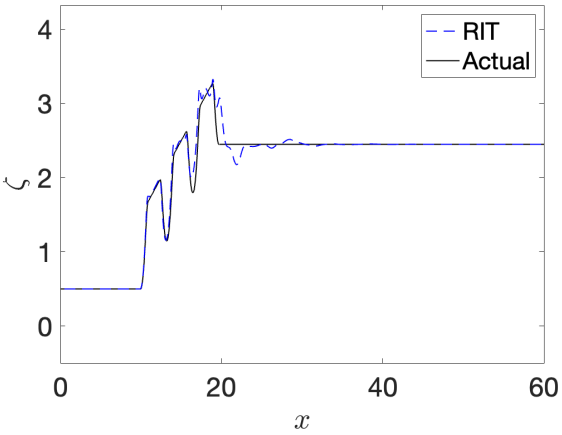
FIGURE 5.19: Scale Factor: 100; Top left: The Ricker source wavelet before passing through varying reflectivities. Top right: The Ricker source wavelet passing through varying reflectivities. Bottom: The Ricker source wavelet after passing through varying reflectivities.



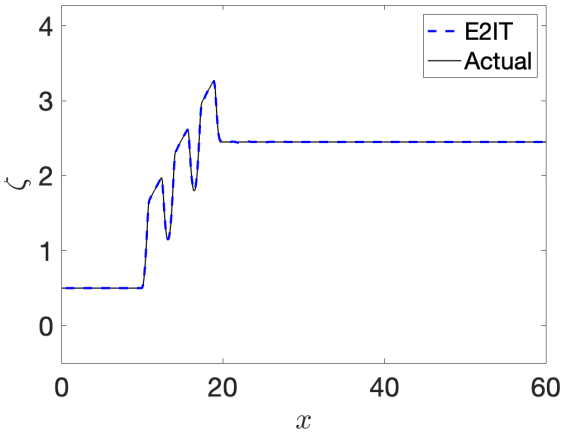
(A) Ricker measured data



(B) Born approximation



(C) Refined impedance transform

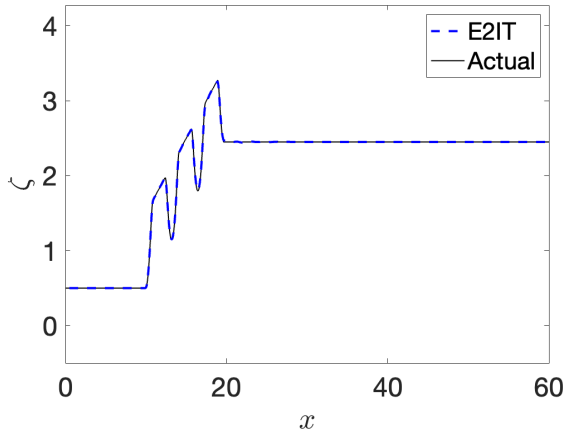


(D) Echoes-to-impedance transform

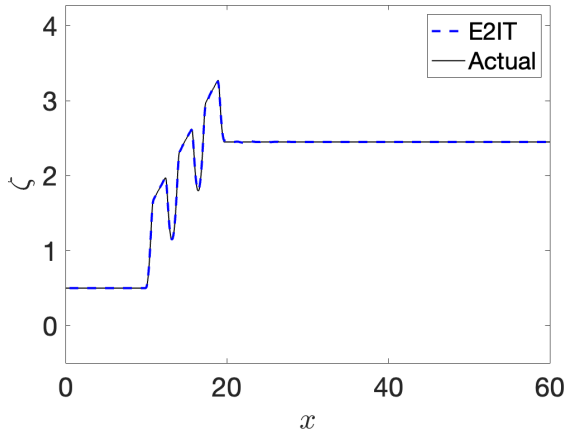
FIGURE 5.20: Scattering and Inverse Scattering of Ricker in Toothed Ramp

	l_2 Errors
BA	0.2423
RIT	0.0911
E2IT	0.0780

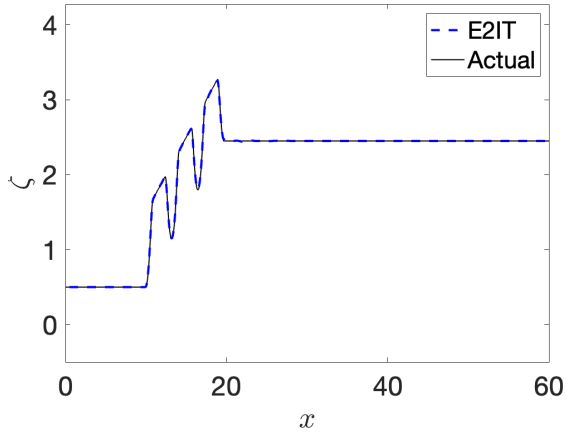
TABLE 5.5: l_2 Relative Errors for Ricker in Toothed Ramp



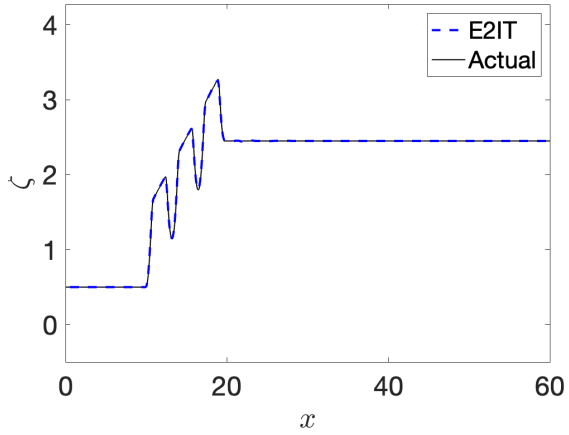
(A) Every 2nd point



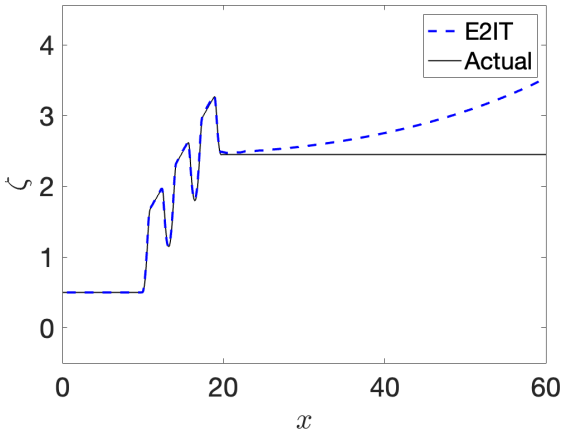
(B) Every 4th point



(C) Every 8th point



(D) Every 16th point



(E) Every 32th points

FIGURE 5.21: Data Downsampling on Ricker Inverse Scattering for Toothed Ramp. Limit: Every 37th point

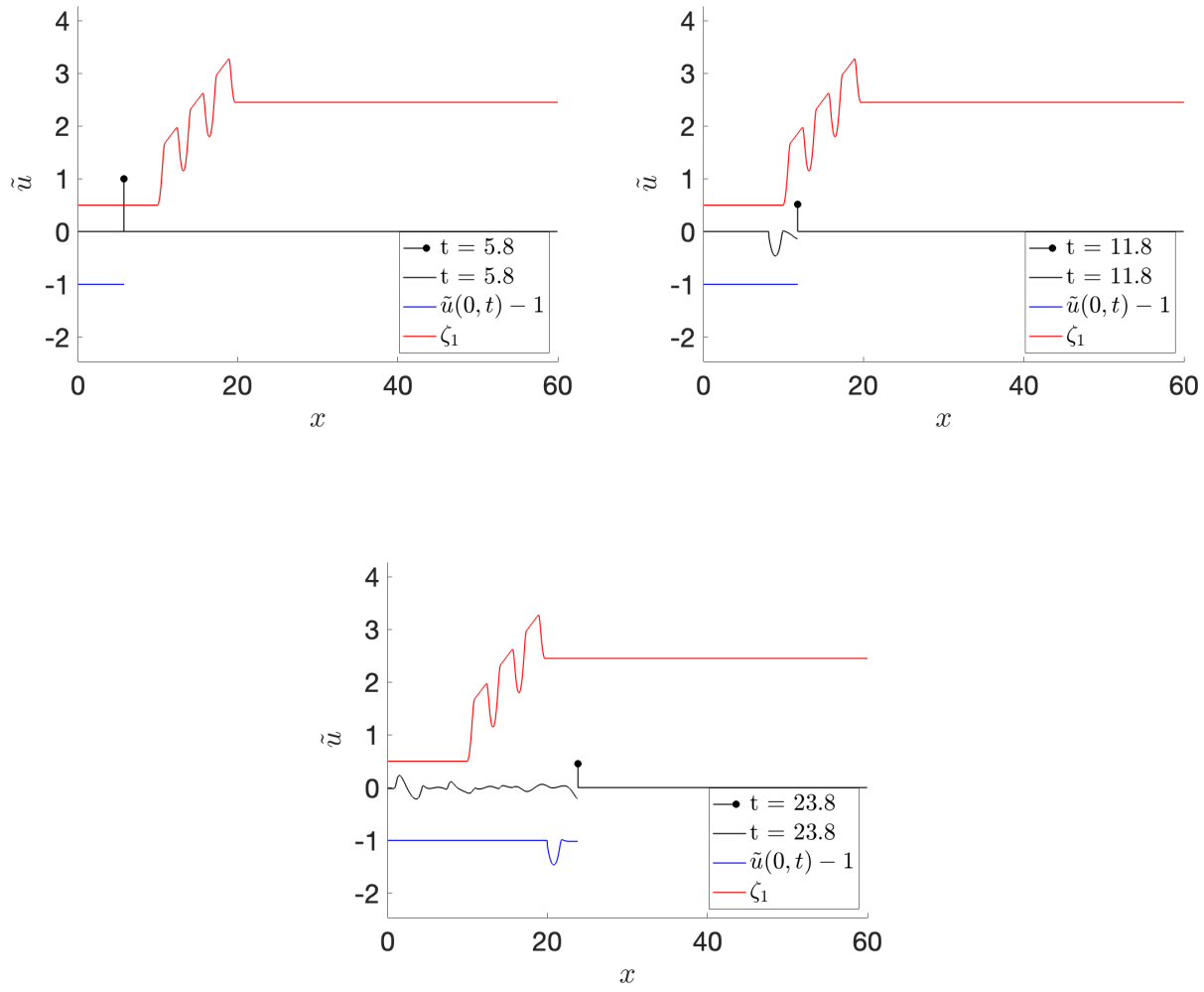
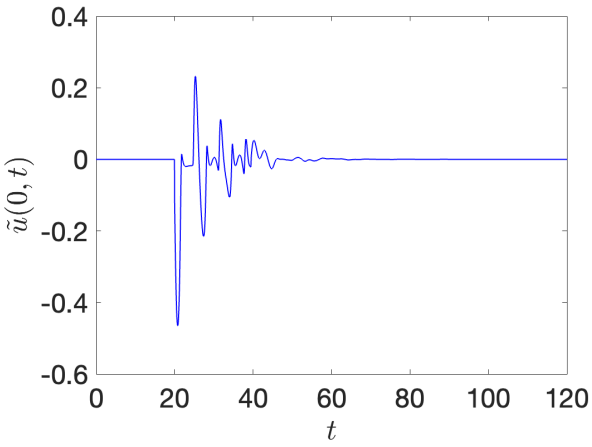
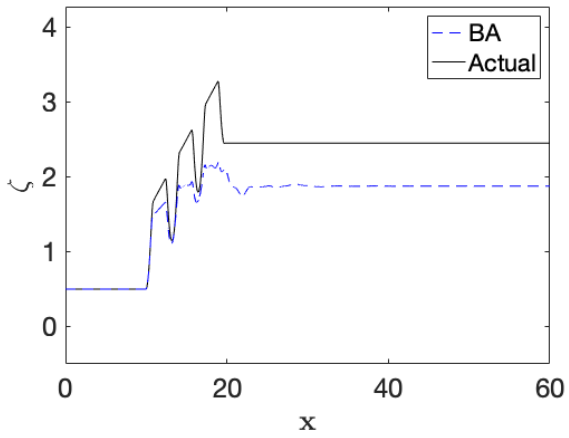


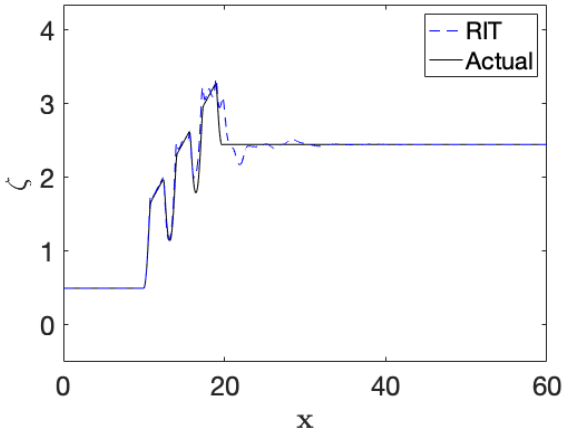
FIGURE 5.22: Scale Factor: 32; Top left: The Dirac source wavelet before passing through varying reflectivities. Top right: The Dirac source wavelet passing through varying reflectivities. Bottom: The Dirac source wavelet after passing through varying reflectivities.



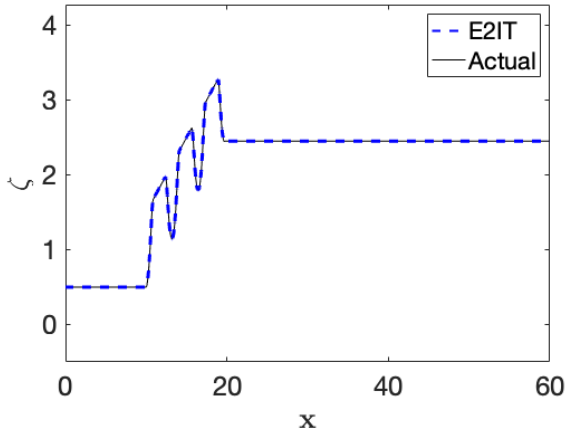
(A) Dirac measured data



(B) Born approximation



(C) Refined impedance transform

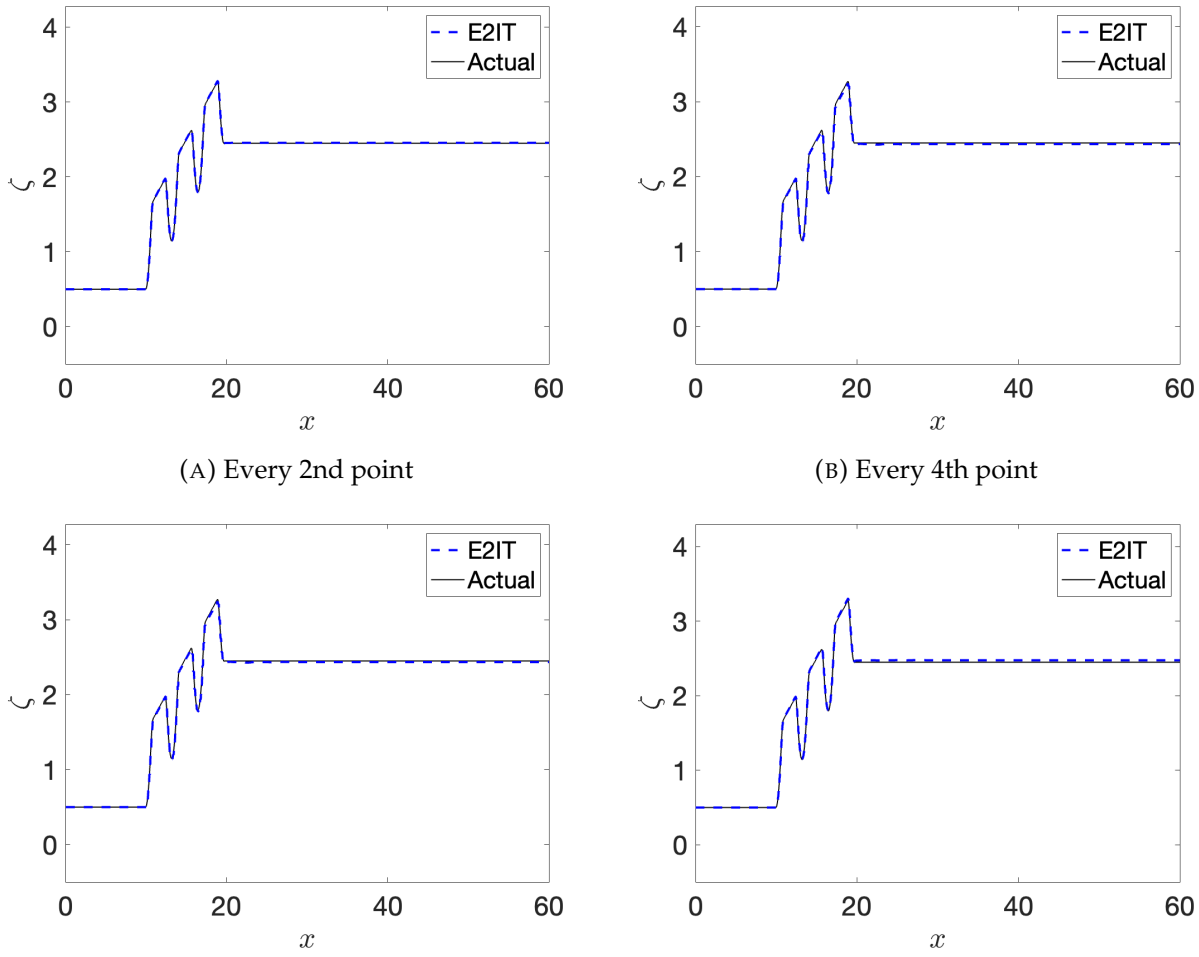


(D) Echoes-to-impedance transform

FIGURE 5.23: Scattering and Inverse Scattering of Dirac in Toothed Ramp

	l_2 Errors
BA	0.0248
RIT	0.0021
E2IT	**2.2118E-4**

TABLE 5.6: l_2 Relative Errors of Dirac in Toothed Ramp



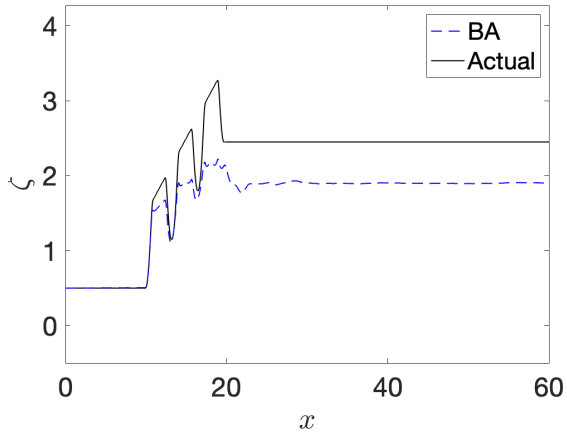
(A) Every 2nd point

(B) Every 4th point

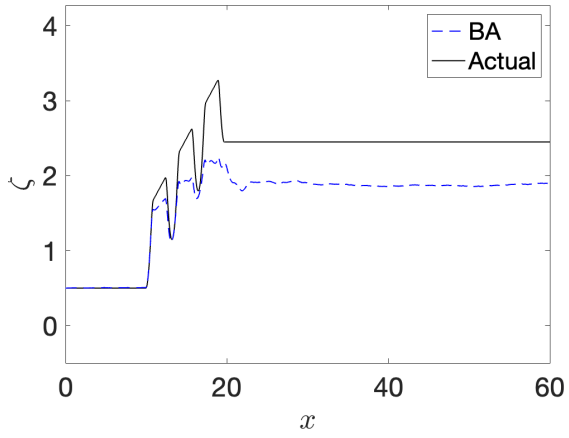
(C) Every 8th point

(D) Every 16th point

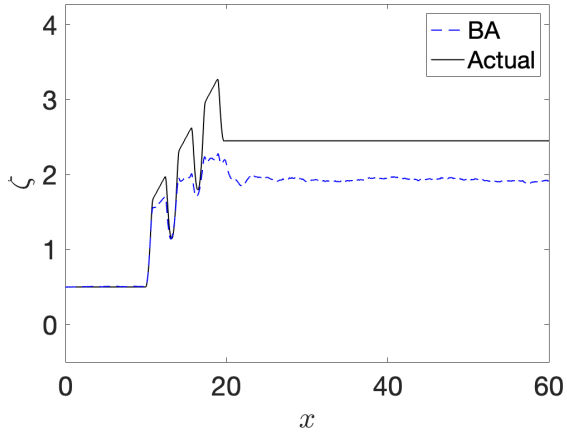
FIGURE 5.24: Data Downsampling on Dirac Inverse Scattering for Toothed Ramp. Limit: 27th point.



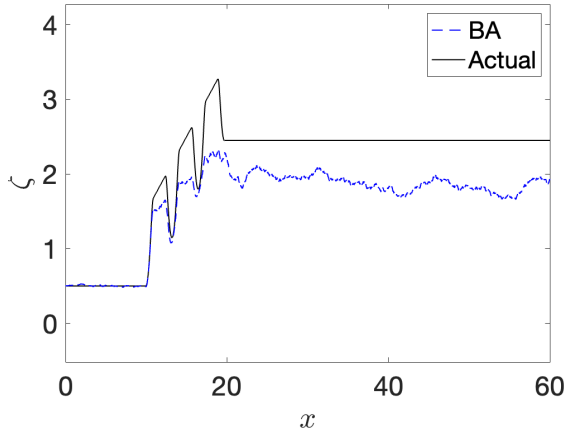
5% Gaussian Noise



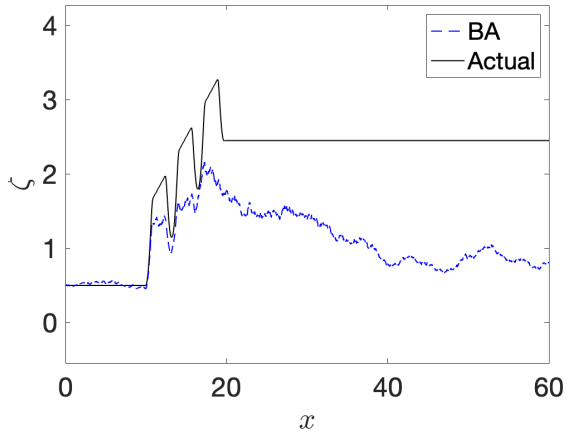
10% Gaussian Noise



25% Gaussian Noise



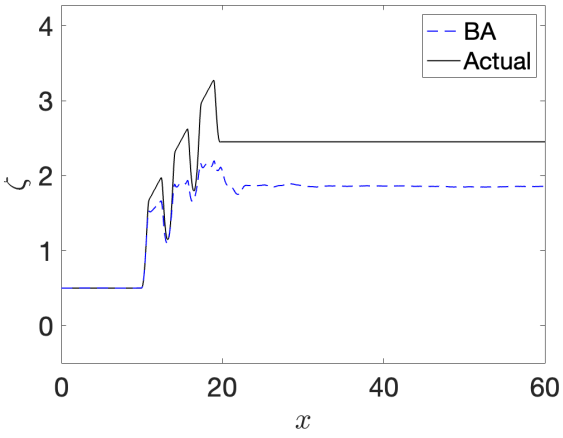
100% Gaussian Noise



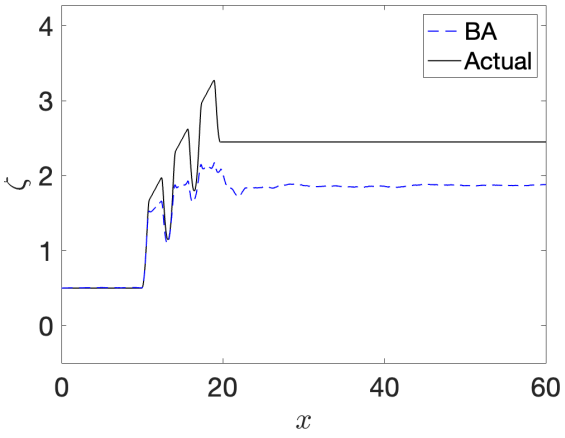
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.2266
10	0.2303
25	0.2120
100	0.2336
200	0.5360

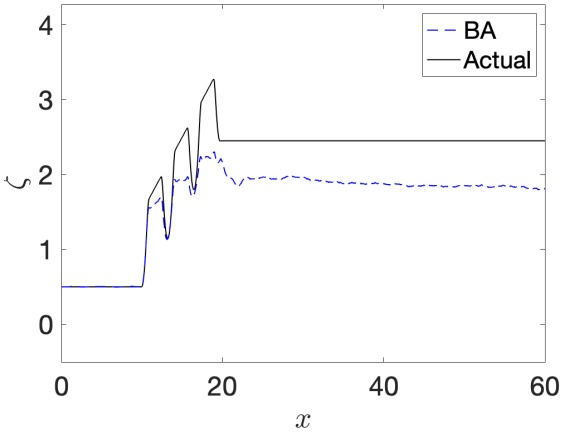
FIGURE 5.25: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



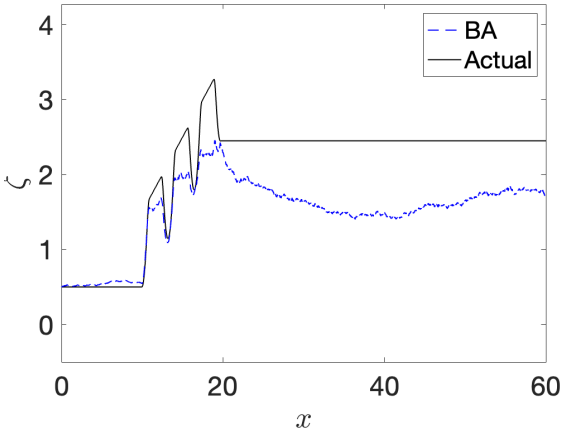
5% Uniform Noise



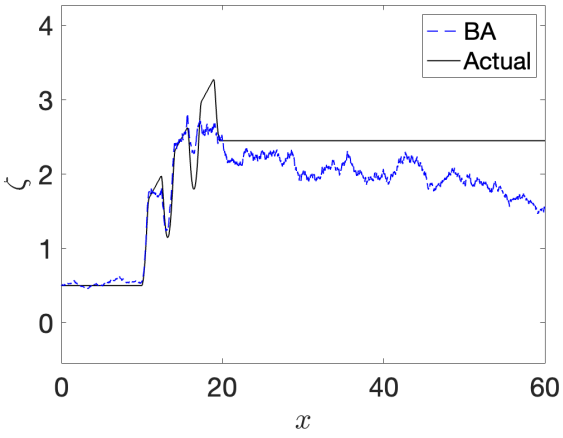
10% Uniform Noise



25% Uniform Noise



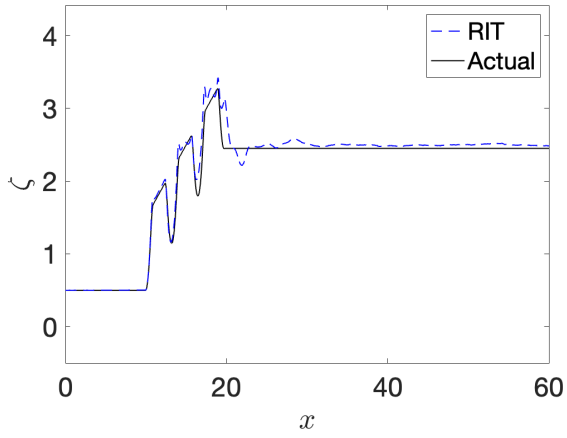
100% Uniform Noise



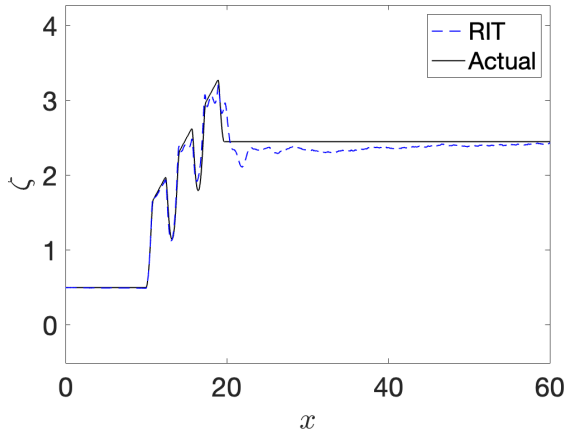
200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.2405
10	0.2410
25	0.2285
100	0.3100
200	0.1911

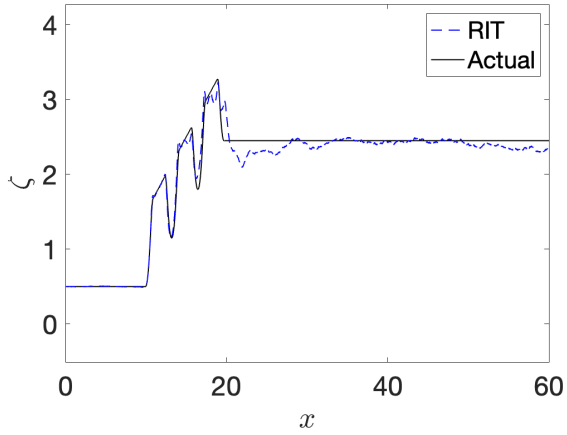
FIGURE 5.26: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



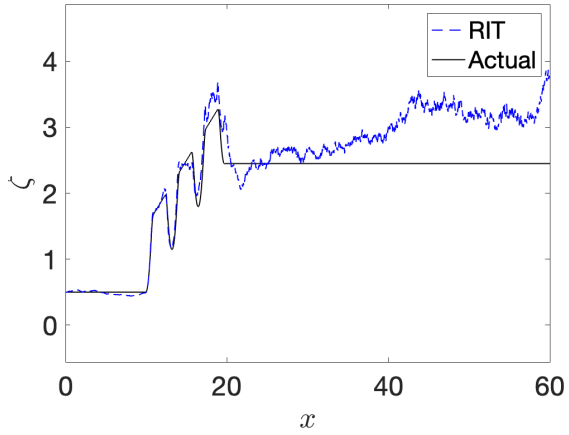
5% Gaussian Noise



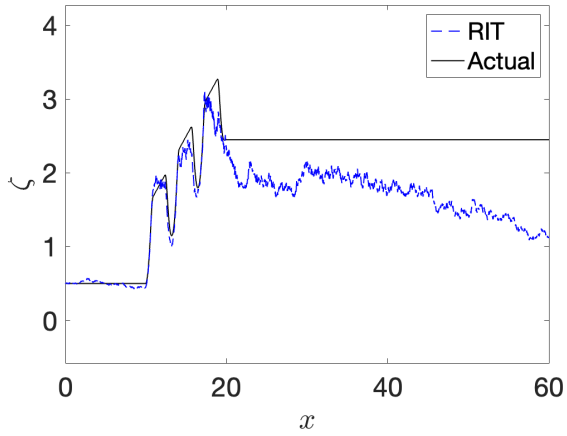
10% Gaussian Noise



25% Gaussian Noise



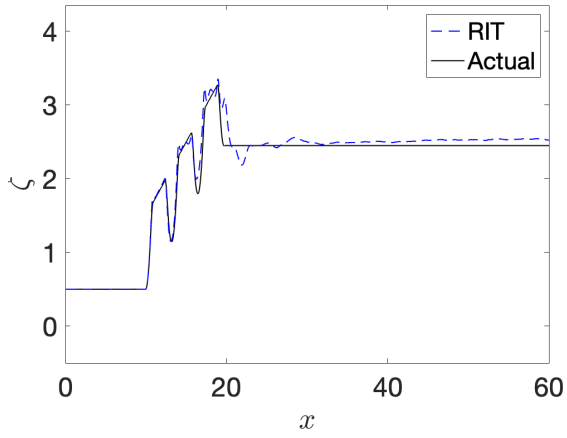
100% Gaussian Noise



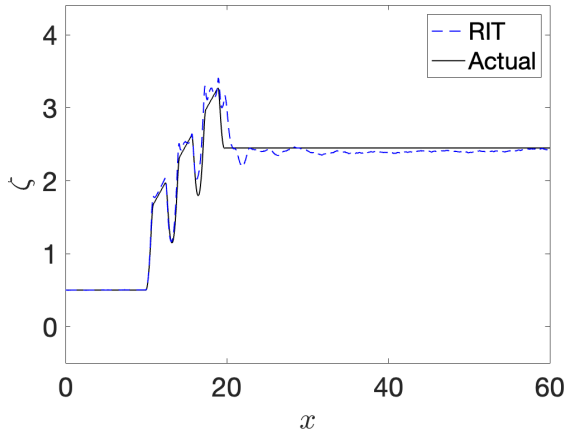
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0466
10	0.0470
25	0.0465
100	0.2250
200	0.2958

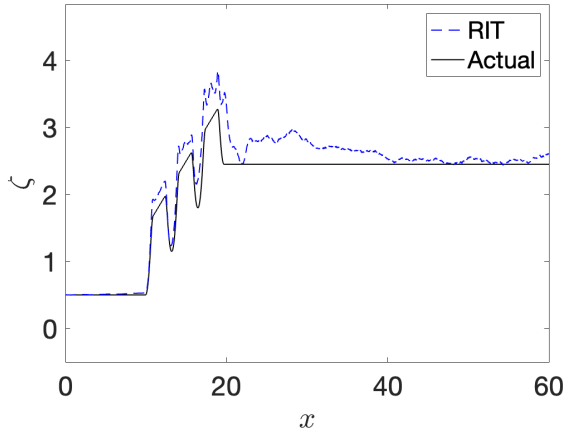
FIGURE 5.27: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



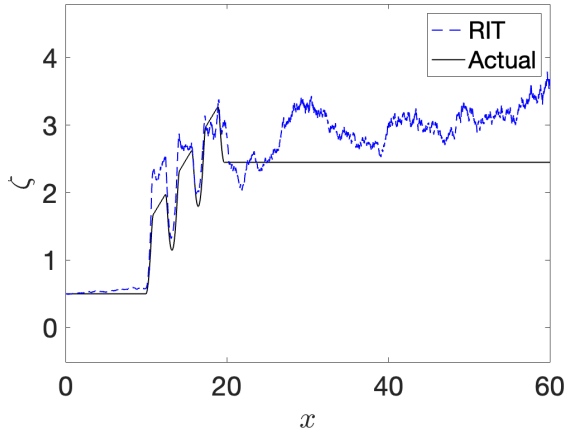
5% Uniform Noise



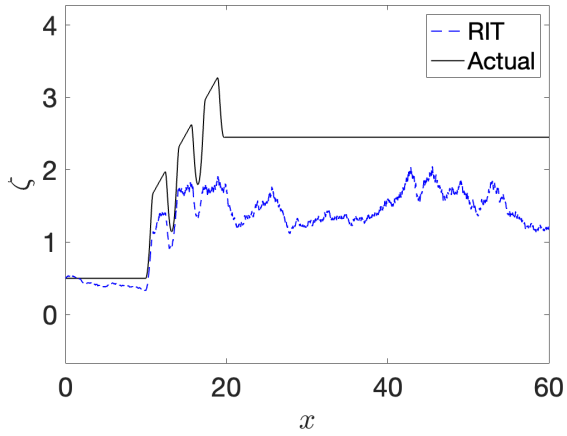
10% Uniform Noise



25% Uniform Noise



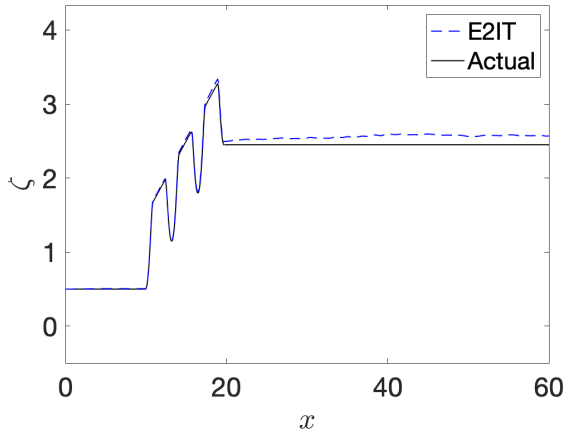
100% Uniform Noise



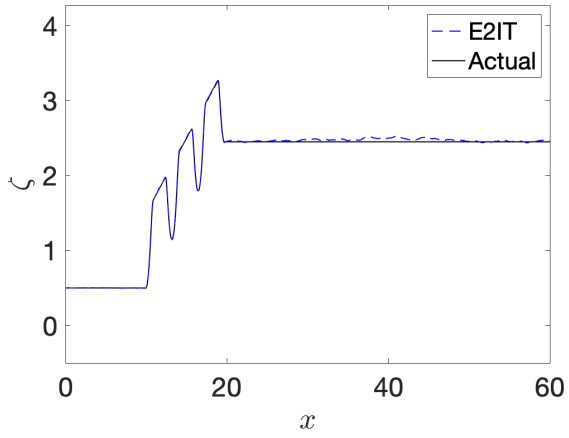
200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0454
10	0.0475
25	0.1116
100	0.2303
200	0.3979

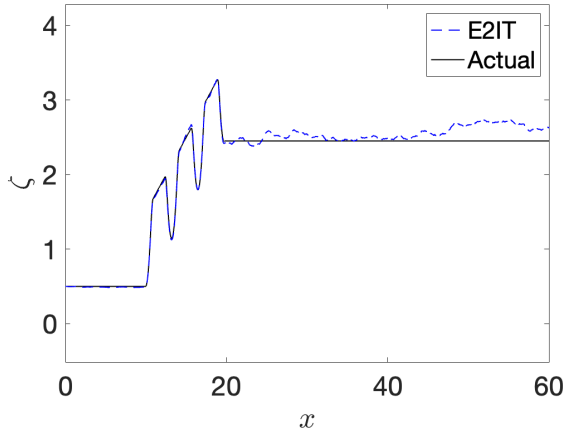
FIGURE 5.28: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



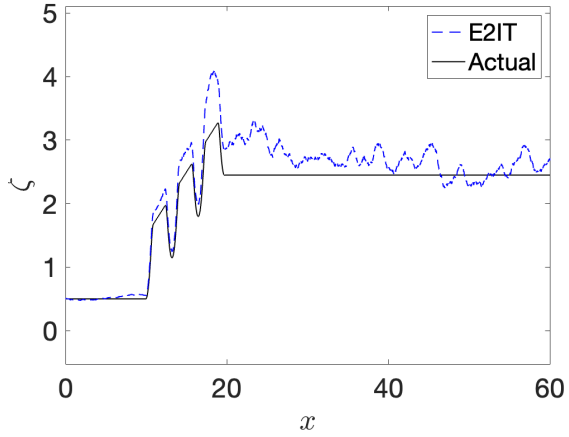
5% Gaussian Noise



10% Gaussian Noise



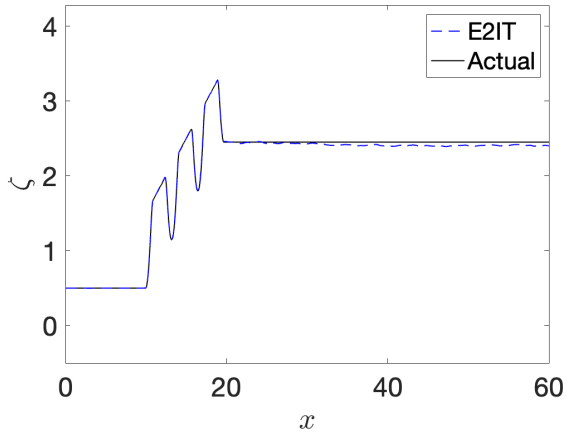
25% Gaussian Noise



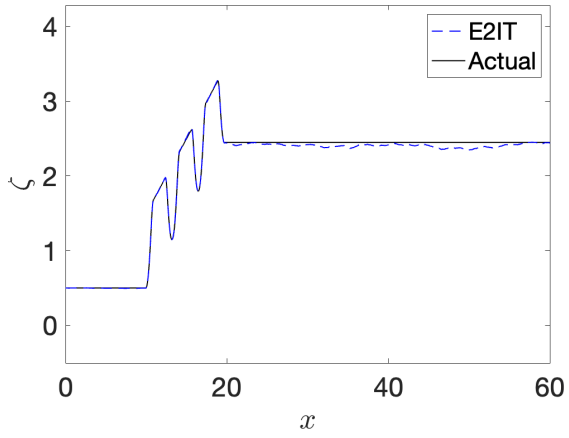
100% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0419
10	0.0112
25	0.0508
100	0.1426

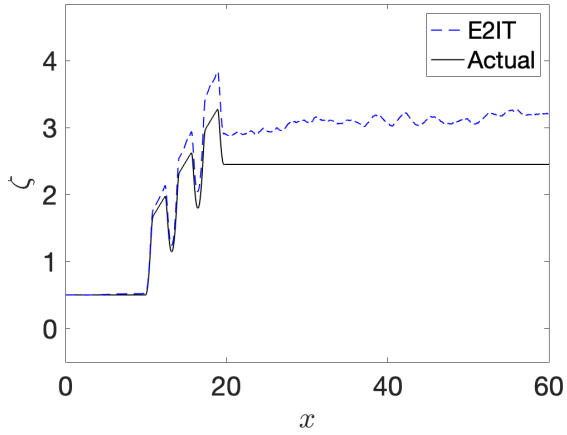
FIGURE 5.29: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



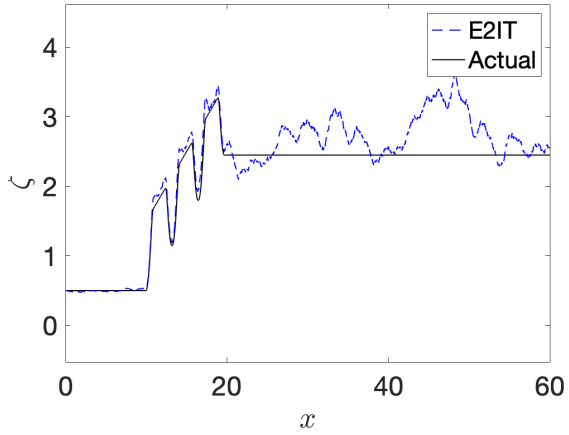
5% Uniform Noise



10% Uniform Noise



25% Uniform Noise



100% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0141
10	0.0179
25	0.2462
100	0.1554

FIGURE 5.30: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.

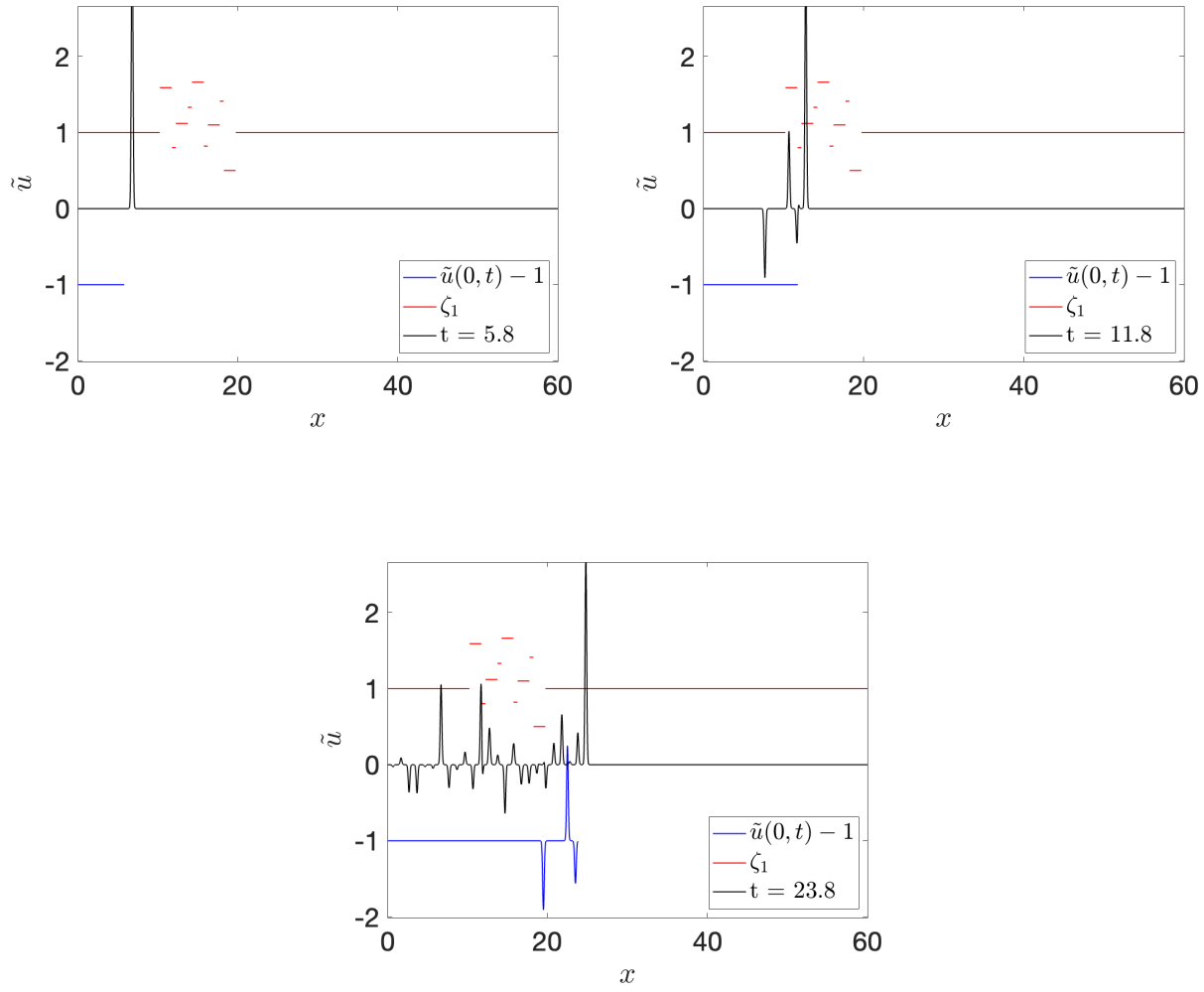
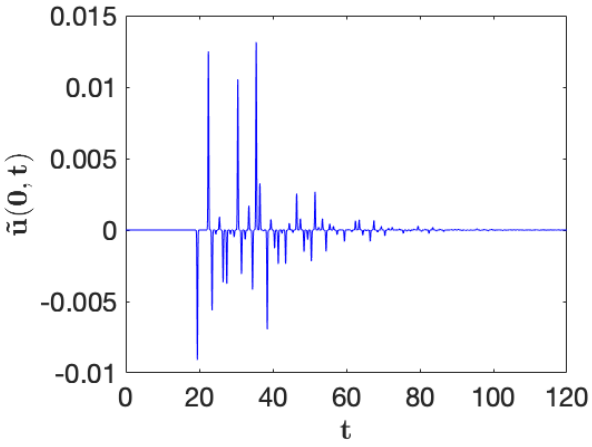
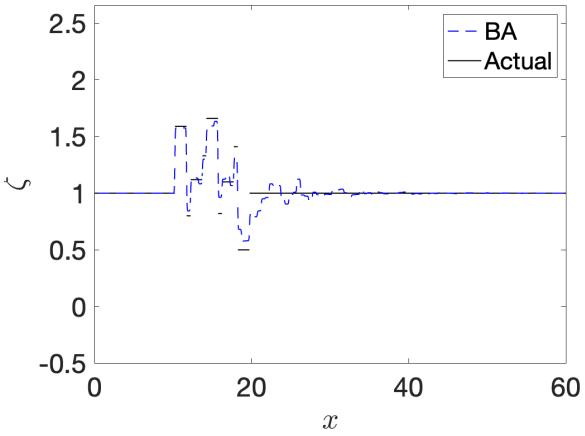


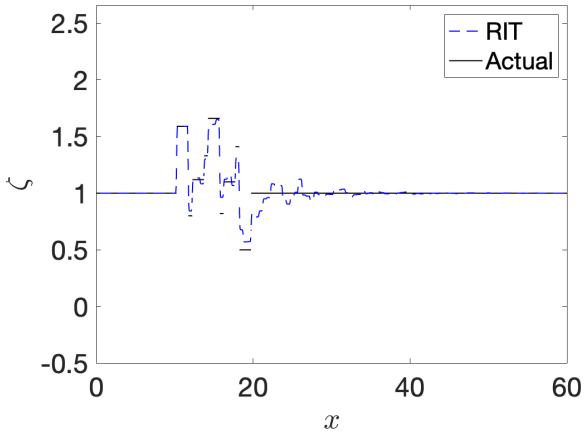
FIGURE 5.31: Scale Factor: 100; Top left: The Gaussian source wavelet before passing through varying reflectivities. Top right: The Gaussian source wavelet passing through varying reflectivities. Bottom: The Gaussian source wavelet after passing through varying reflectivities.



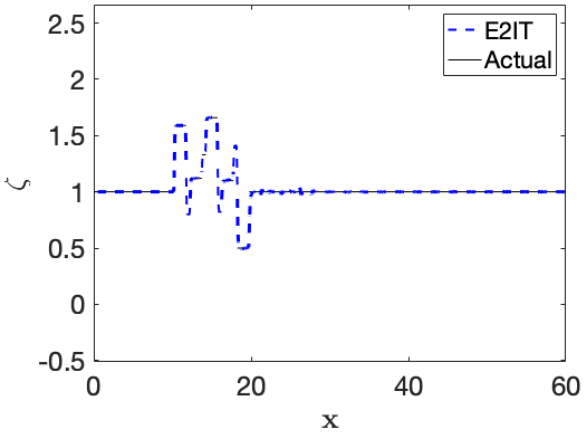
(A) Gaussian measured data



(B) Born approximation



(C) Refined impedance transform

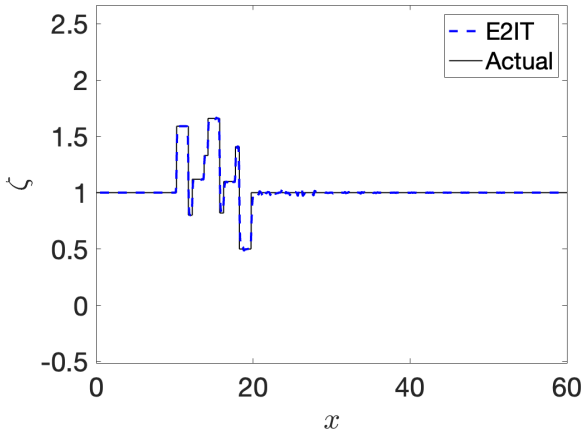


(D) Echoes-to-impedance transform

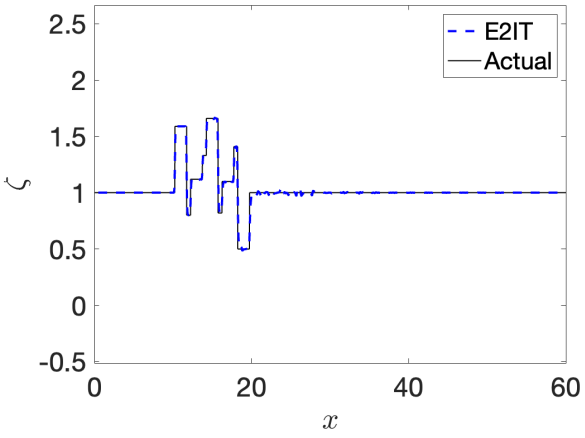
FIGURE 5.32: Scattering and Inverse Scattering of Gaussian in Random Steps

	l_2 Errors
BA	0.1346
RIT	0.1352
E2IT	0.1459

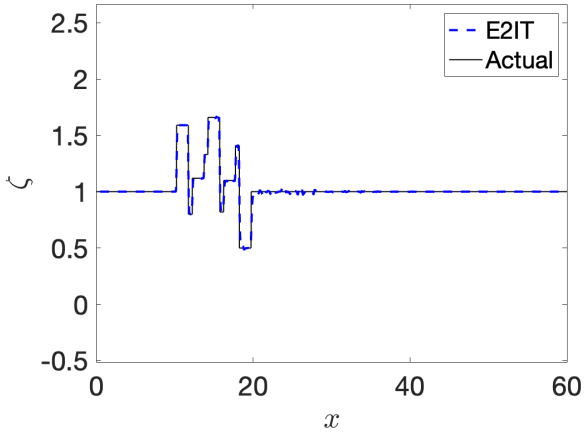
TABLE 5.7: l_2 Relative Errors for Gaussian in Random Steps



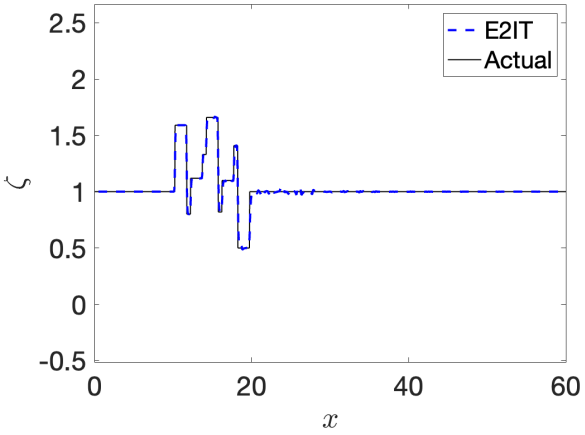
(A) Every 2nd point



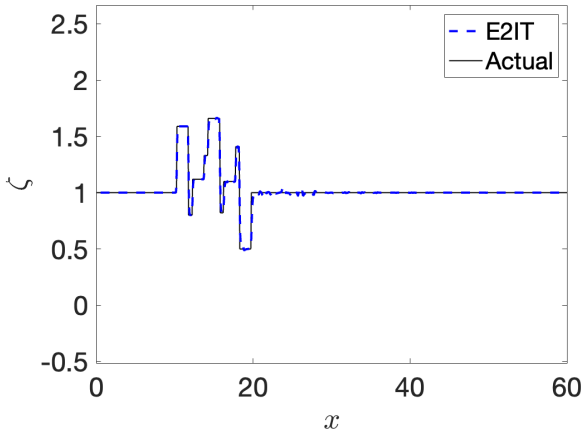
(B) Every 4th point



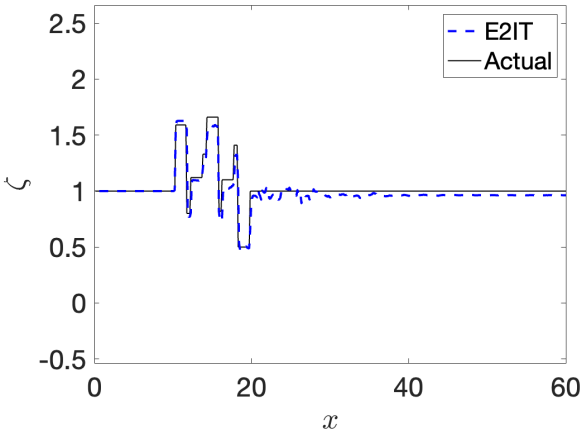
(C) Every 8th point



(D) Every 16th point



(E) Every 32th points



(F) Every 64th points

FIGURE 5.33: Data Downsampling on Gaussian Inverse Scattering for Random Steps. Limit: Every 73rd point

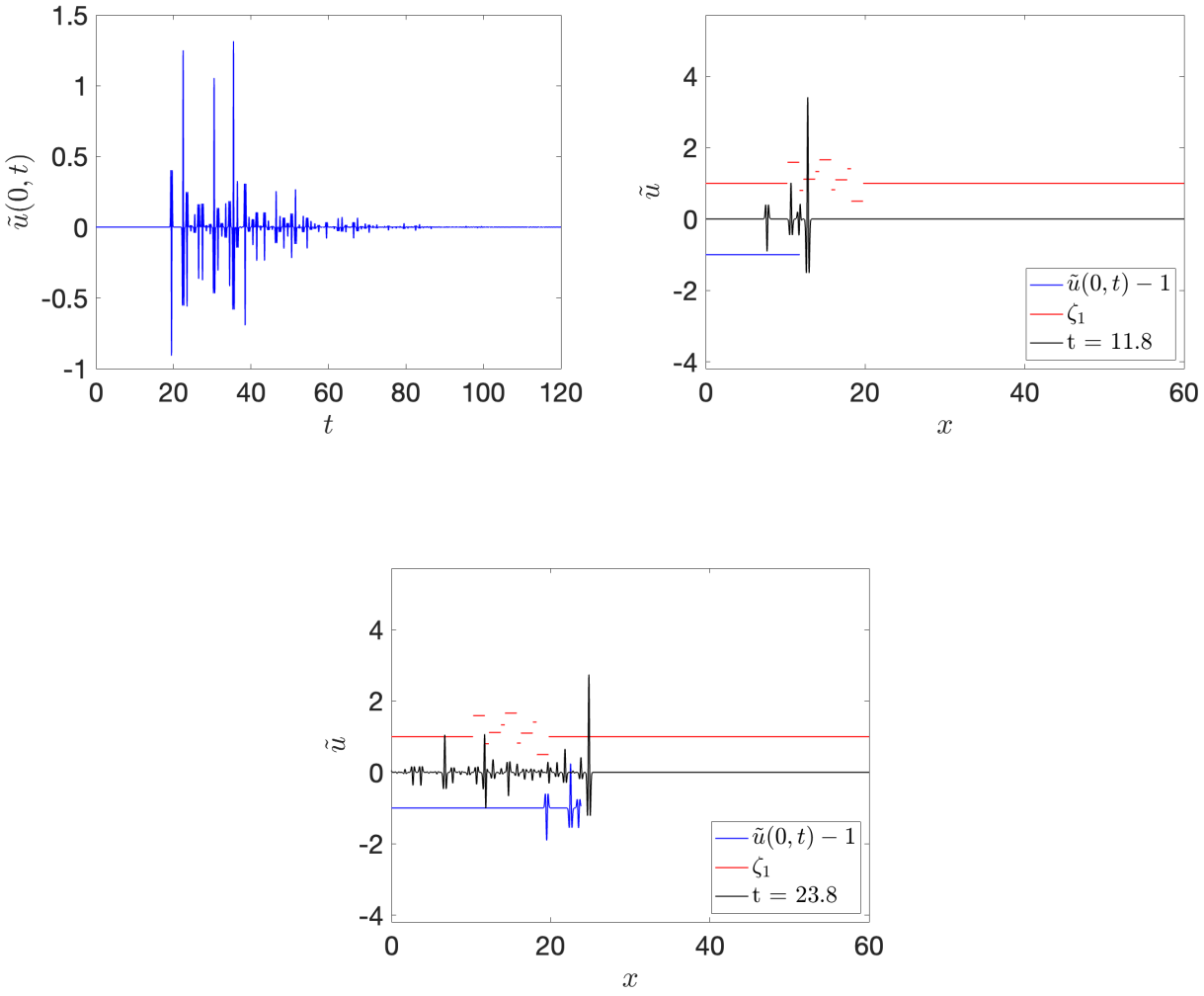
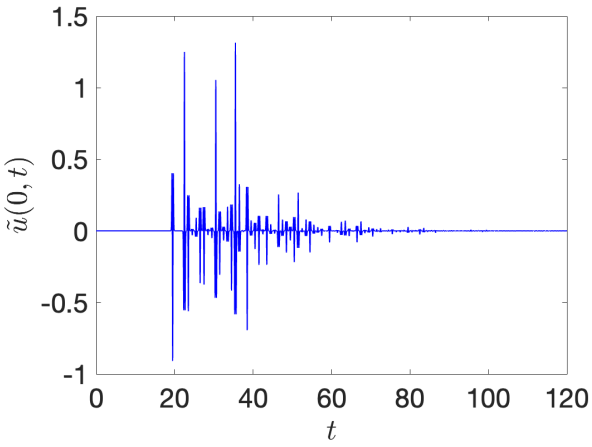
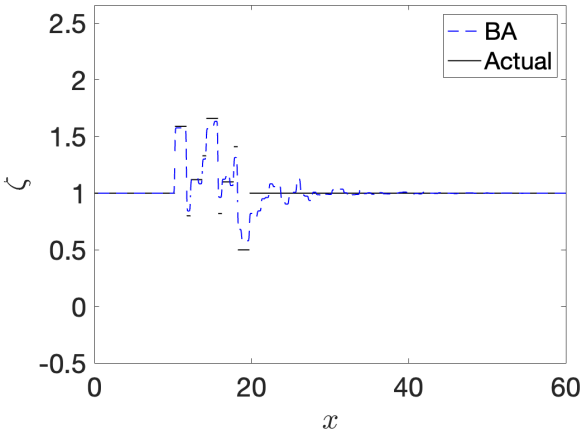


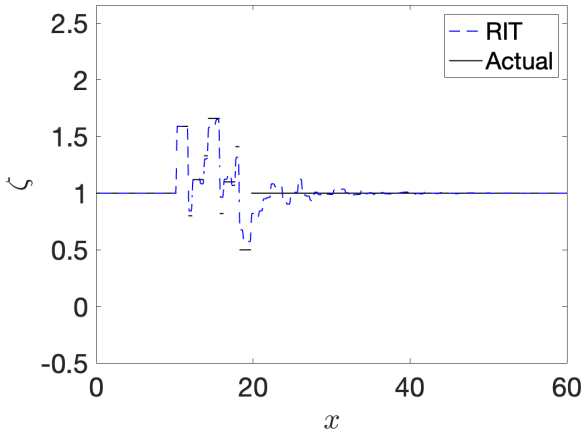
FIGURE 5.34: Top left: The Ricker source wavelet before passing through varying reflectivities. Top right: The Ricker source wavelet passing through varying reflectivities. Bottom: The Ricker source wavelet after passing through varying reflectivities.



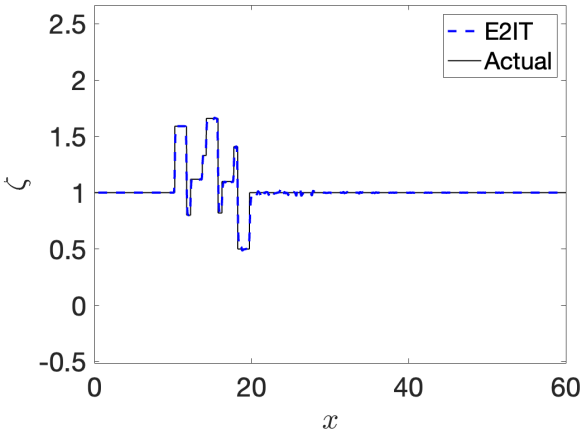
(A) Ricker measured data



(B) Born approximation



(C) Refined impedance transform

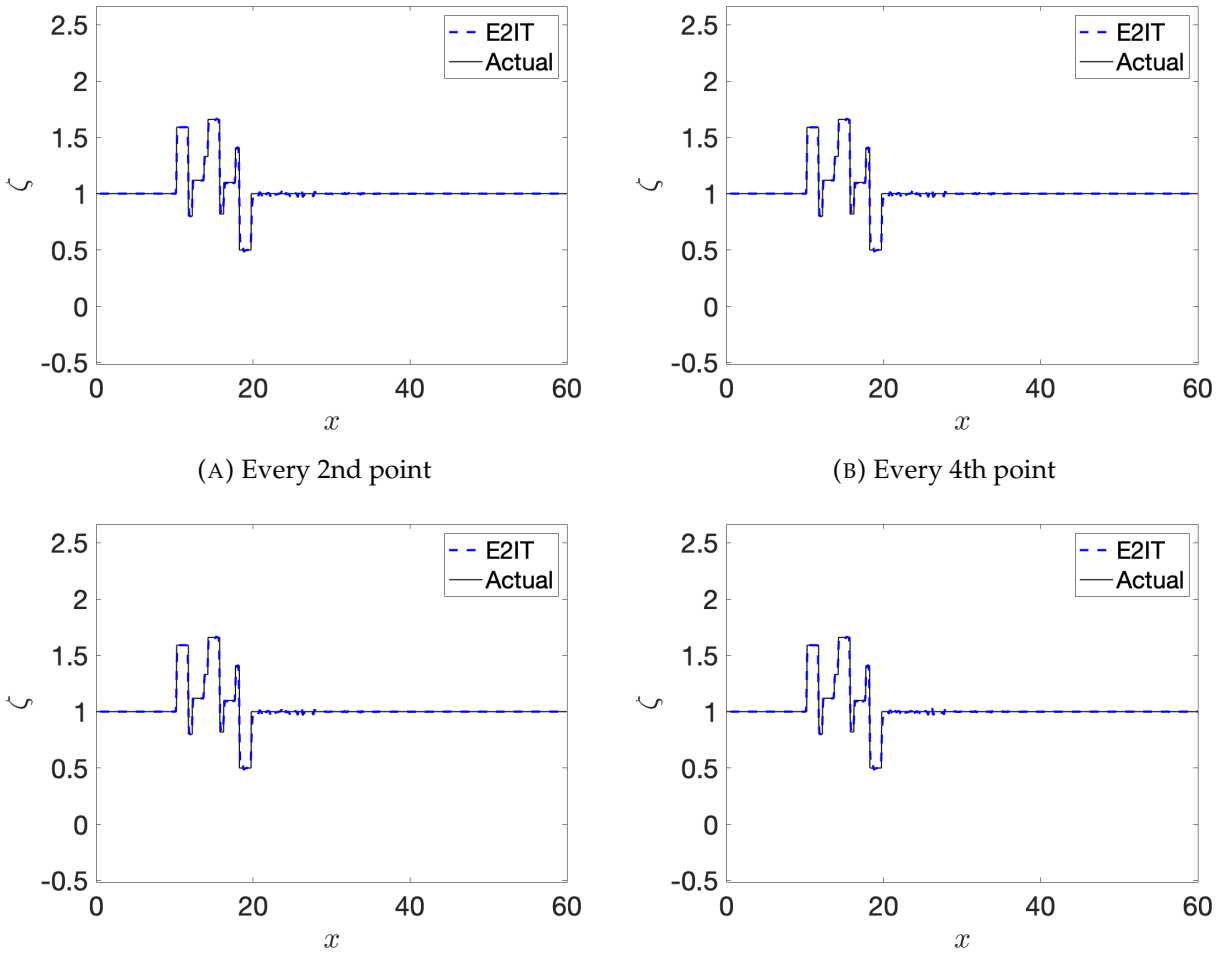


(D) Echoes-to-impedance transform

FIGURE 5.35: Scattering and Inverse Scattering of Ricker in Random Steps

	l_2 Errors
BA	0.1357
RIT	0.1363
E2IT	0.1473

TABLE 5.8: l_2 Relative Errors for Ricker in Random Steps



(A) Every 2nd point

(B) Every 4th point

(C) Every 8th point

(D) Every 16th point

FIGURE 5.36: Data Downsampling on Ricker Inverse Scattering for Random Steps. Limit: Every 27th point

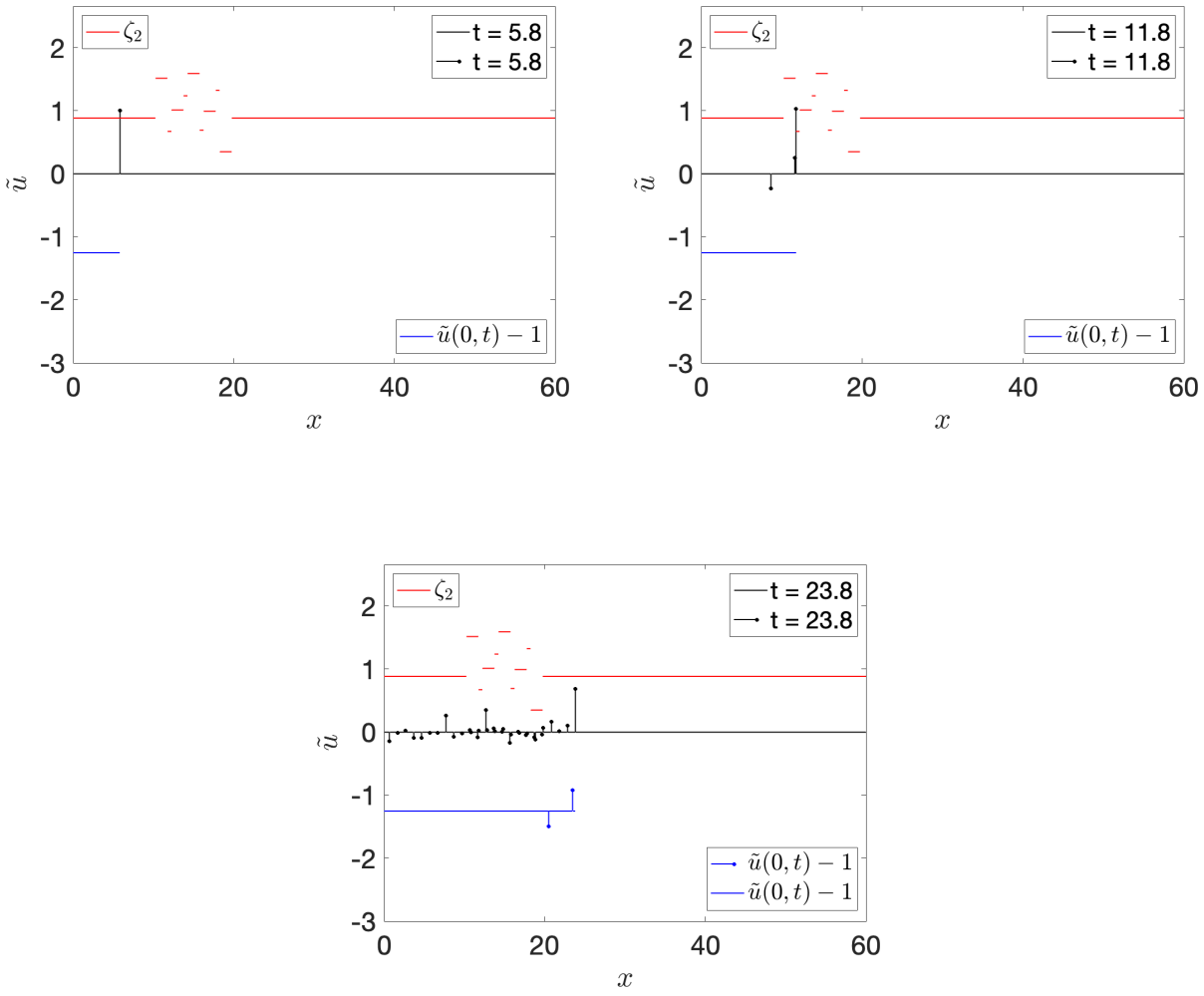
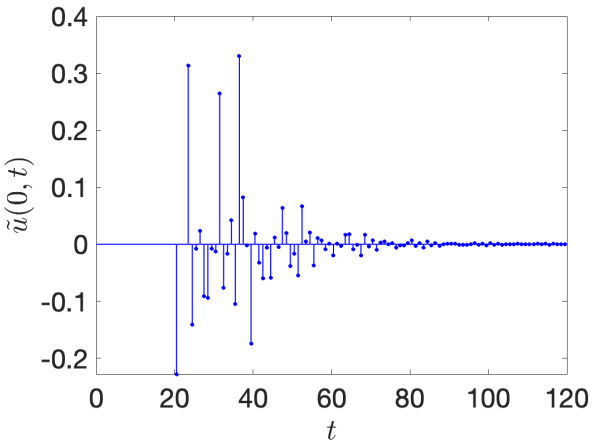
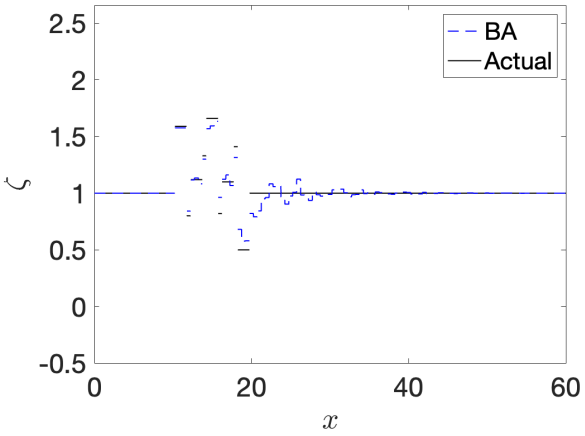


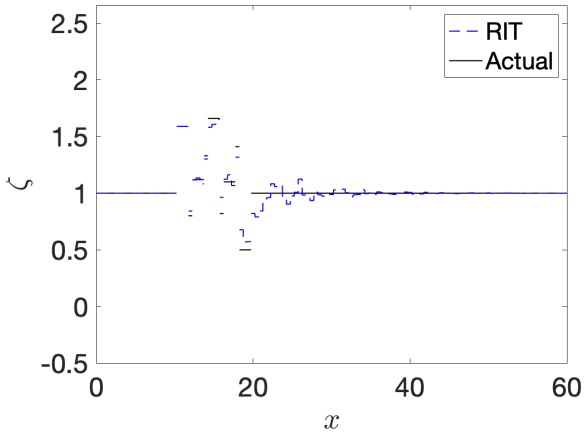
FIGURE 5.37: Top left: The Dirac source wavelet before passing through varying reflectivities. Top right: The Dirac source wavelet passing through varying reflectivities. Bottom: The Dirac source wavelet after passing through varying reflectivities.



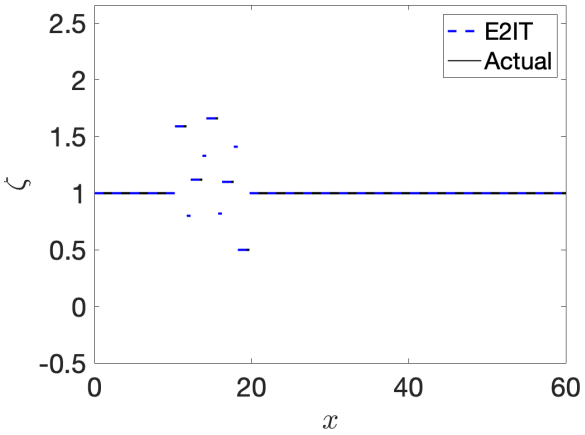
(A) Dirac measured data



(B) Born approximation



(C) Refined impedance transform

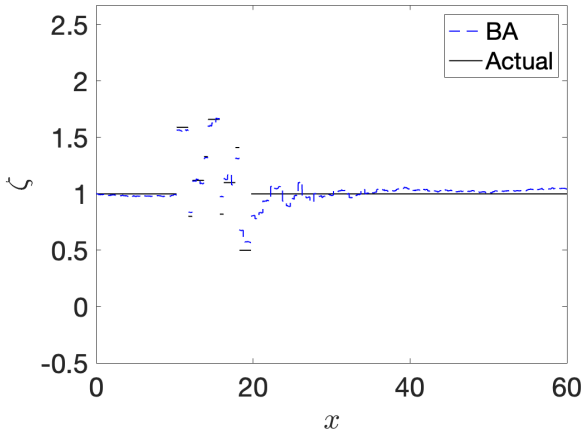


(D) Echoes-to-impedance transform

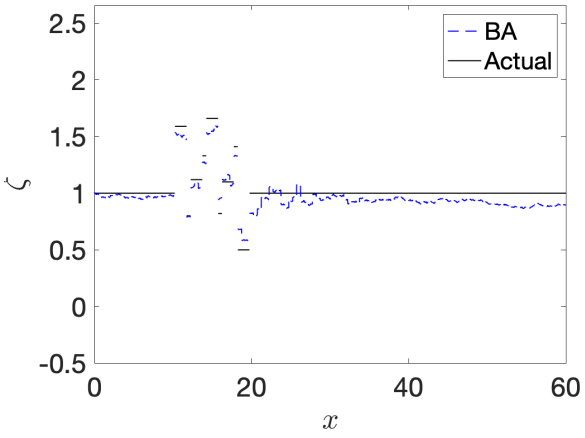
FIGURE 5.38: Scattering and Inverse Scattering of Dirac in Random Steps

	l_2 Errors
BA	0.0482
RIT	0.0476
E2IT	0

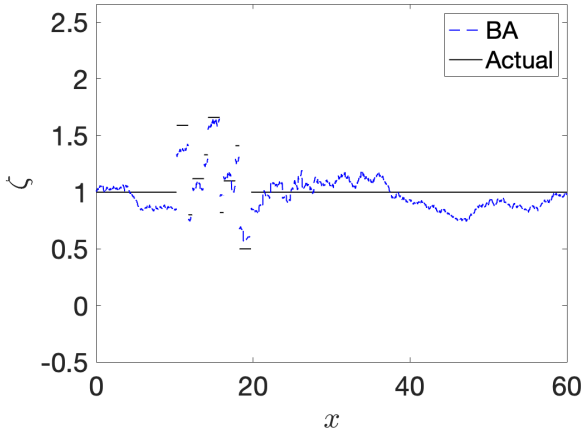
TABLE 5.9: l_2 Relative Errors of Dirac in Random Steps



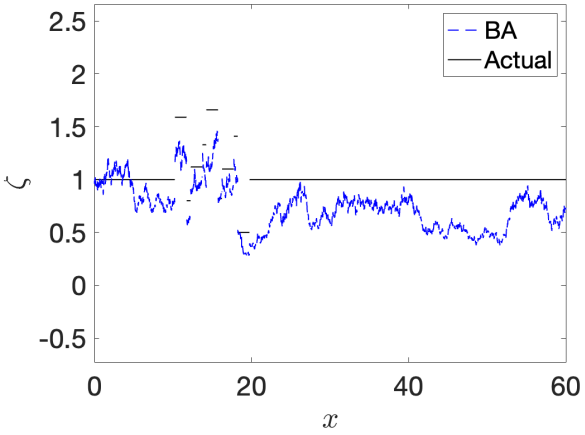
5% Gaussian Noise



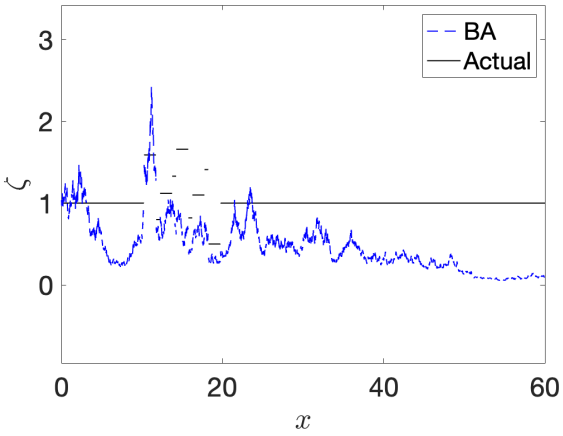
10% Gaussian Noise



25% Gaussian Noise



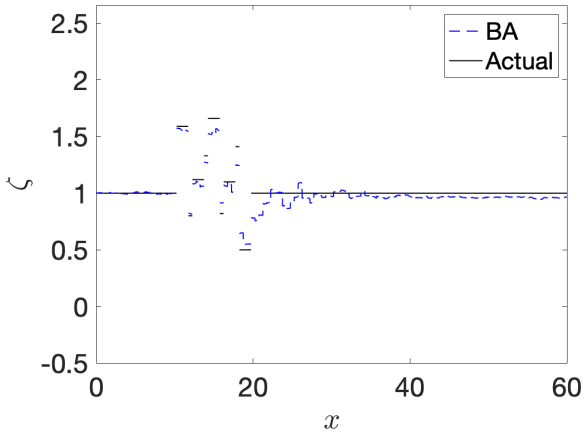
100% Gaussian Noise



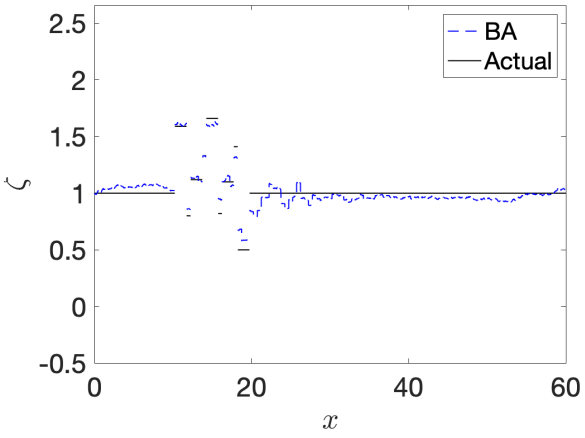
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0533
10	0.0781
25	0.1138
100	0.3232
200	0.6056

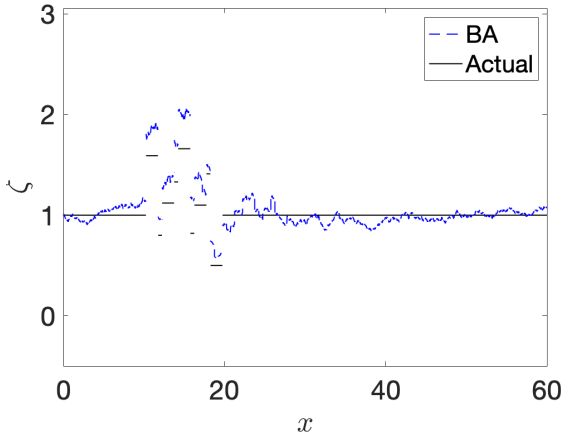
FIGURE 5.39: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



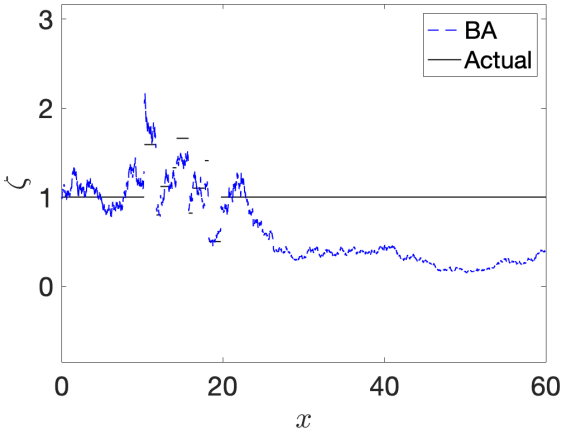
5% Uniform Noise



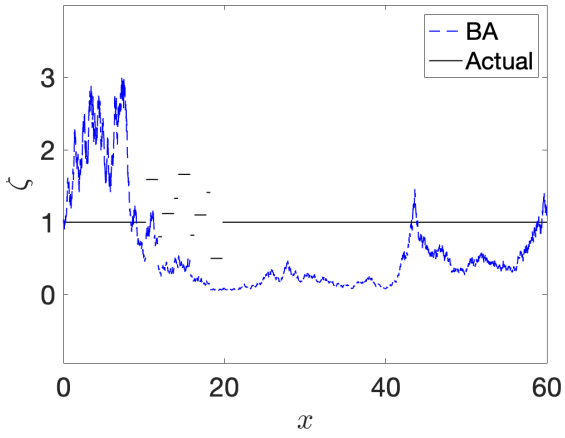
10% Uniform Noise



25% Uniform Noise



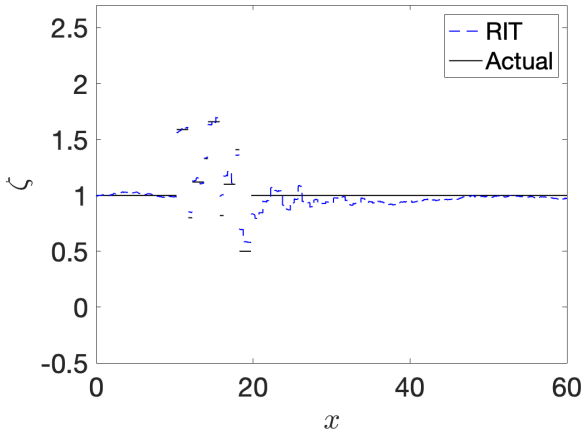
100% Uniform Noise



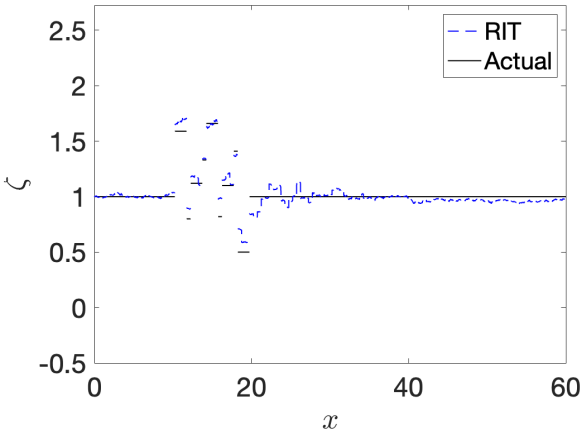
200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0599
10	0.0602
25	0.1185
100	0.5144
200	0.7590

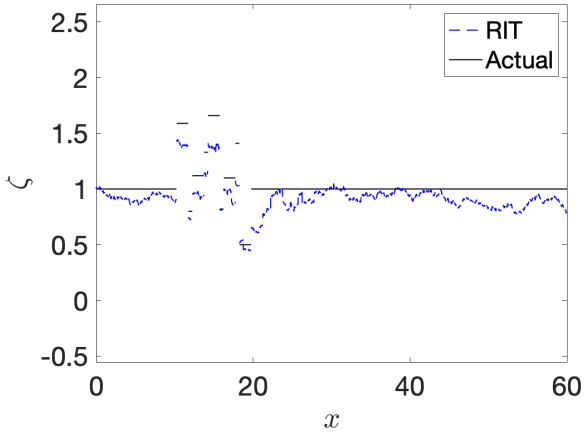
FIGURE 5.40: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



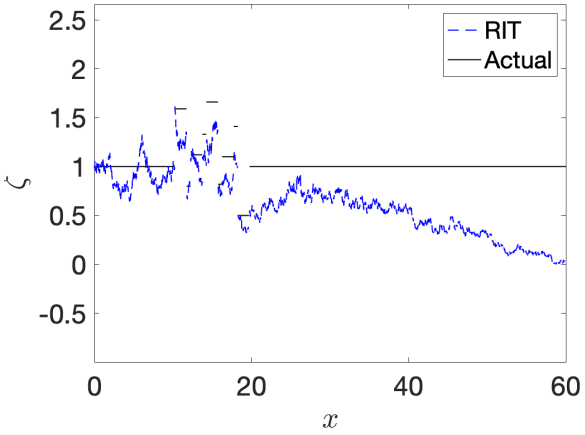
5% Gaussian Noise



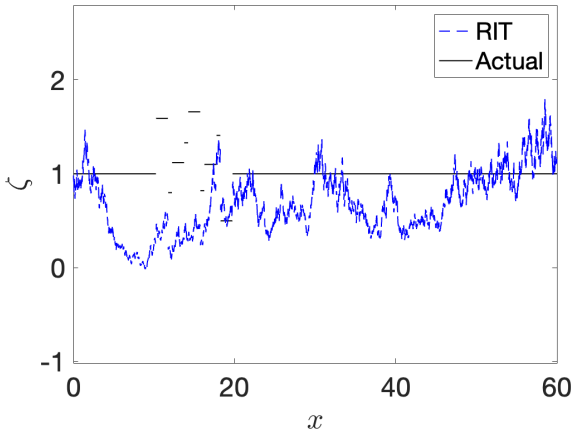
10% Gaussian Noise



25% Gaussian Noise



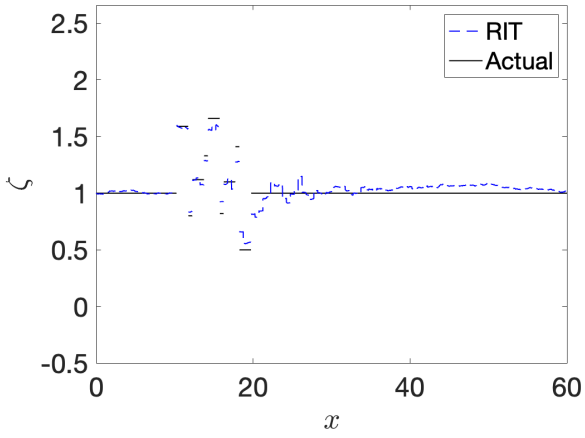
100% Gaussian Noise



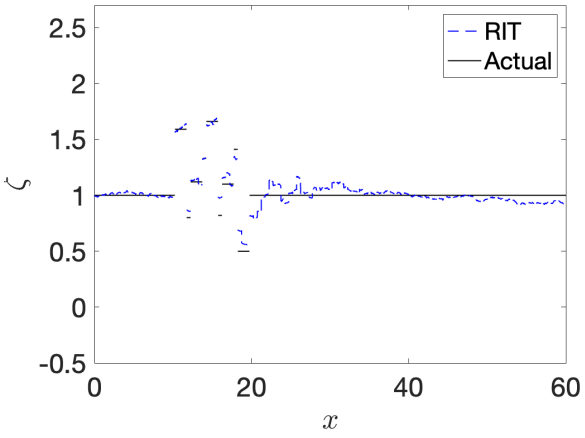
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0584
10	0.0555
25	0.1305
100	0.4847
200	0.5003

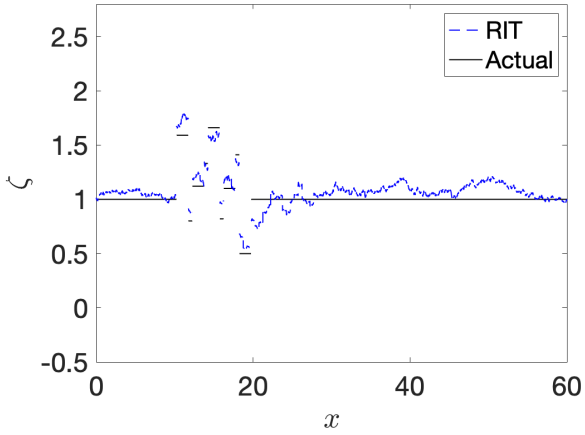
FIGURE 5.41: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



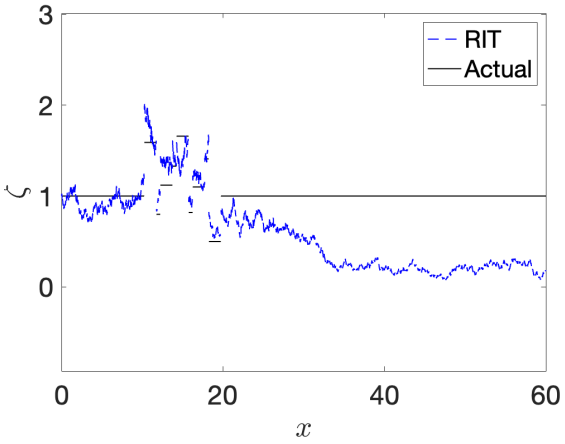
5% Uniform Noise



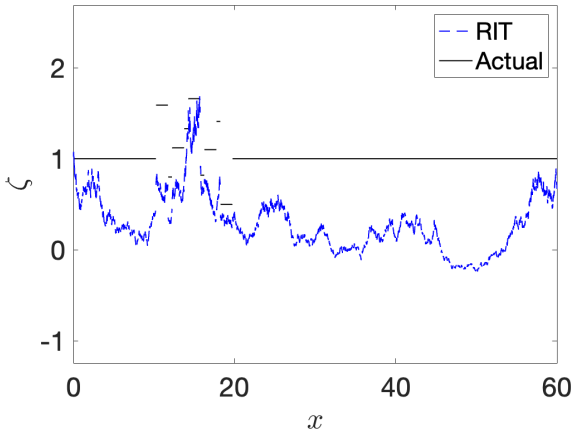
10% Uniform Noise



25% Uniform Noise



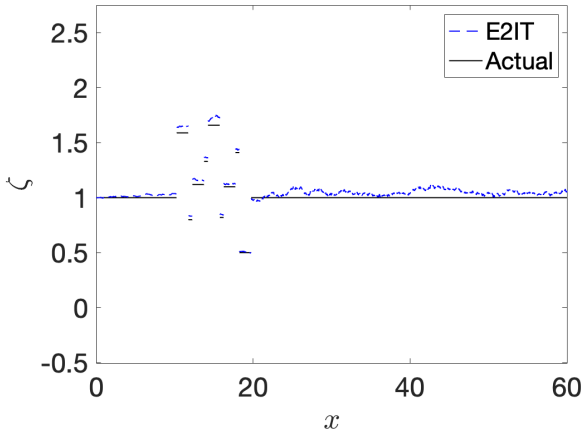
100% Uniform Noise



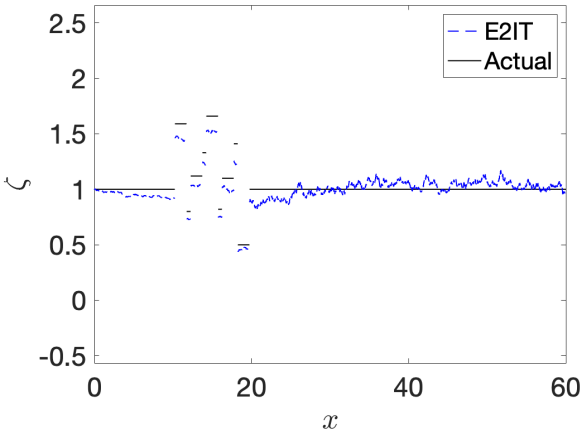
200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0595
10	0.0602
25	0.0932
100	0.5488
200	0.7549

FIGURE 5.42: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



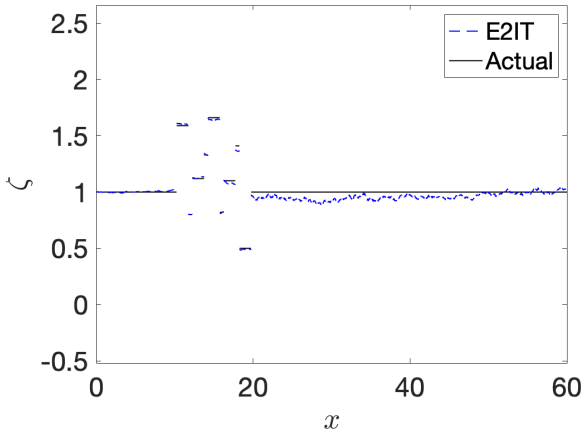
5% Gaussian Noise



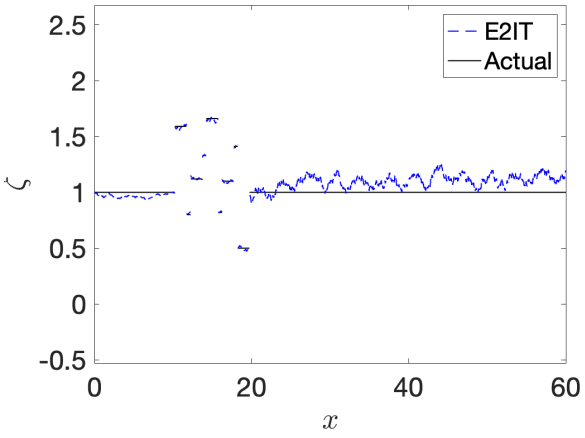
10% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0450
10	0.0687

FIGURE 5.43: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



5% Uniform Noise



10% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.0400
10	0.0896

FIGURE 5.44: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.

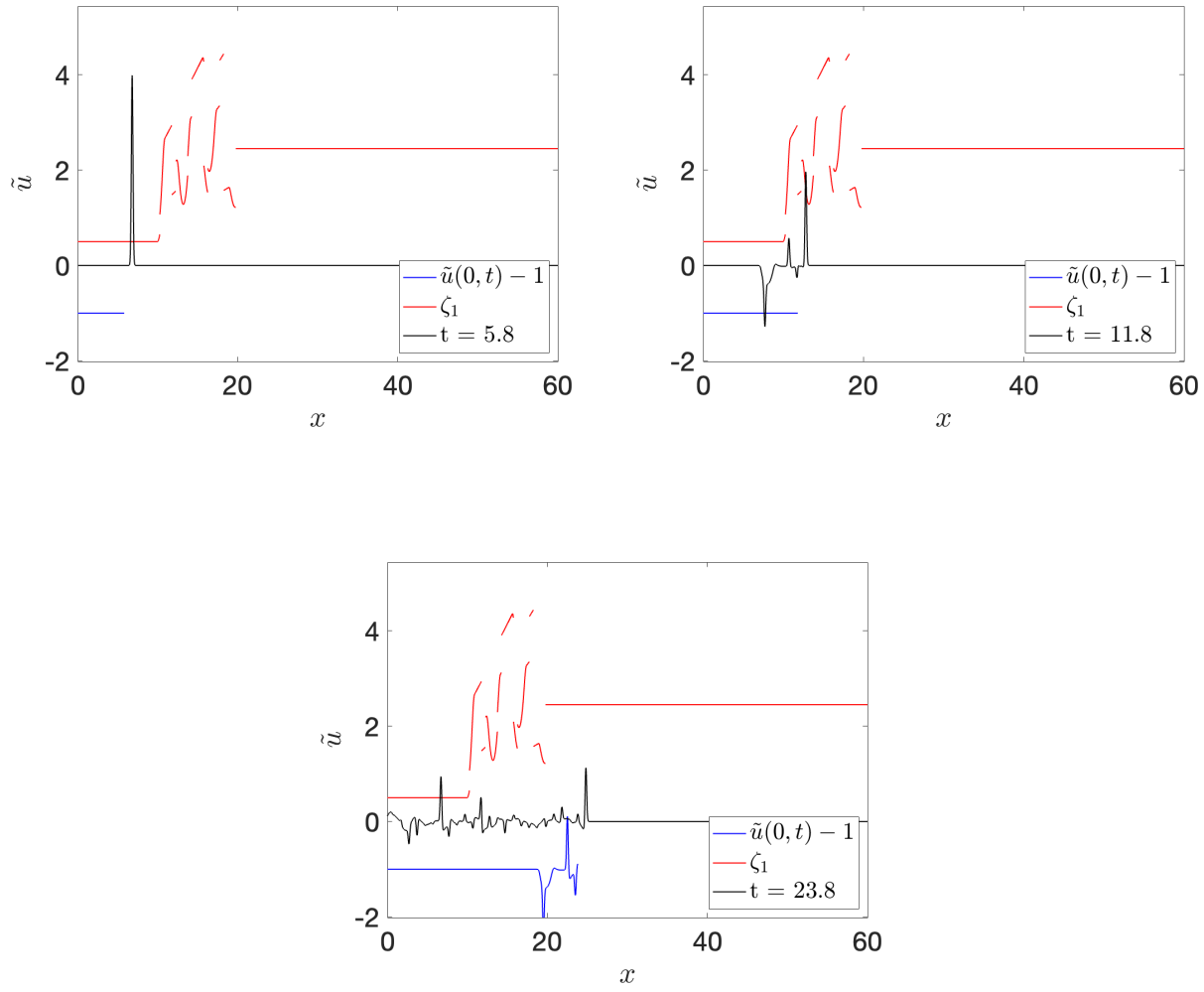
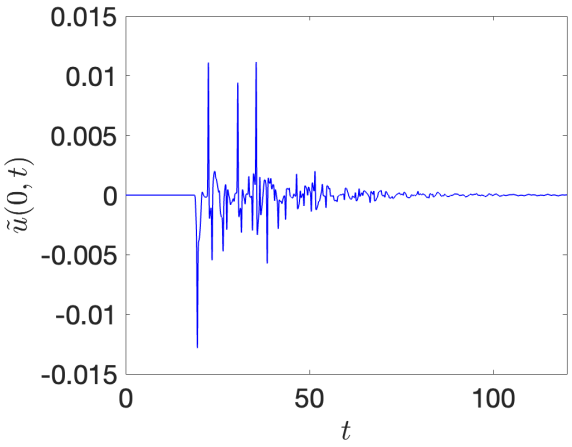
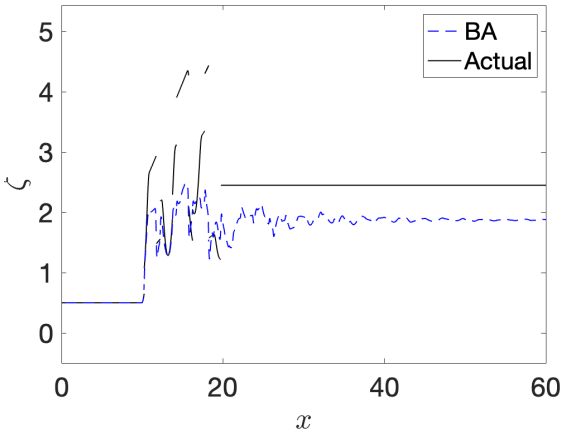


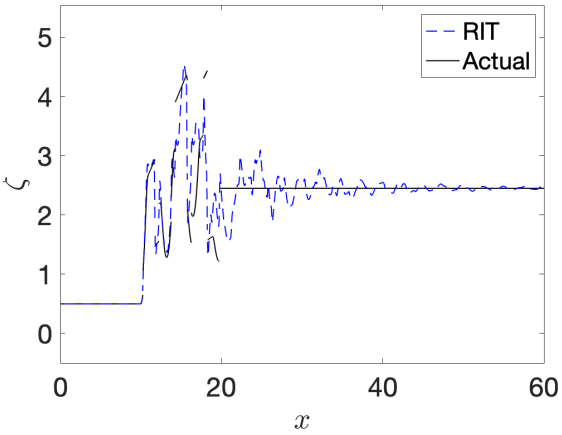
FIGURE 5.45: Scale Factor: 100; Top left: The Gaussian source wavelet before passing through varying reflectivities. Top right: The Gaussian source wavelet passing through varying reflectivities. Bottom: The Gaussian source wavelet after passing through varying reflectivities.



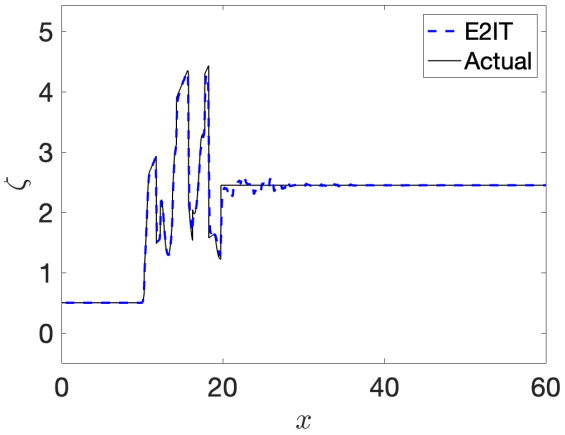
(A) Gaussian measured data



(B) Born approximation



(C) Refined impedance transform

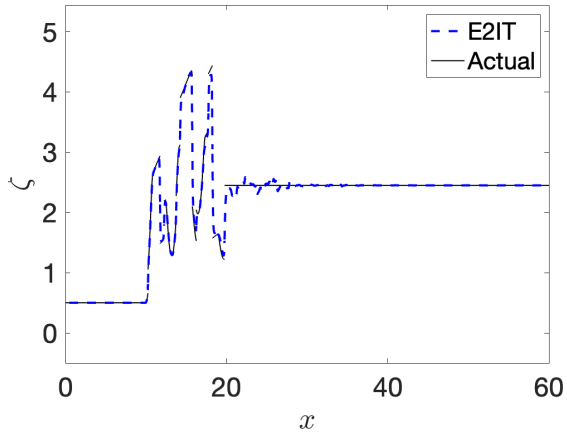


(D) Echoes-to-impedance transform

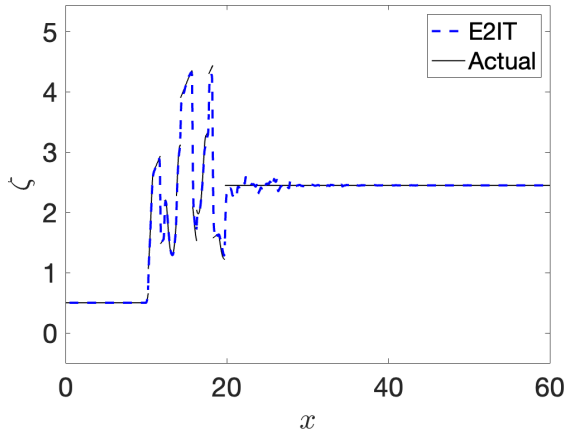
FIGURE 5.46: Scattering and Inverse Scattering of Gaussian in Discontinuous Toothed Ramp

	l_2 Errors
BA	0.2988
RIT	0.2164
E2IT	0.1889

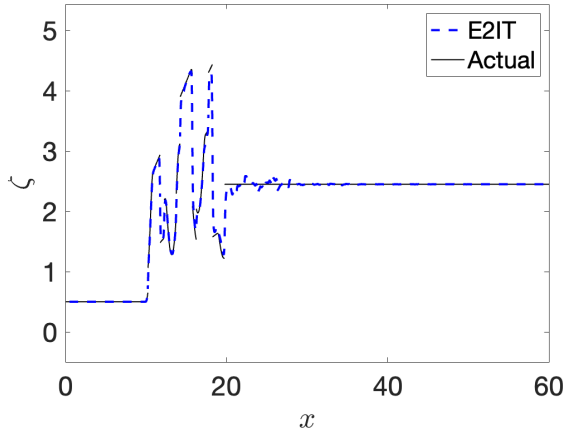
TABLE 5.10: l_2 Relative Errors for Gaussian in Discontinuous Toothed Ramp



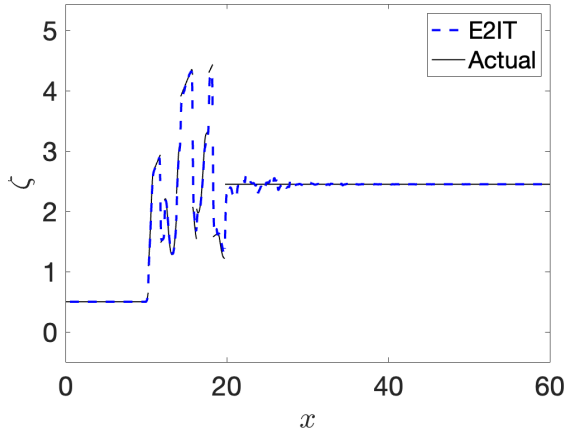
(A) Every 2nd point



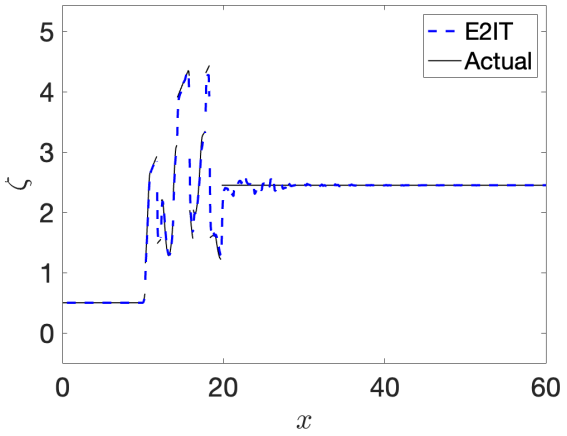
(B) Every 4th point



(C) Every 8th point



(D) Every 16th point



(E) Every 32th points

FIGURE 5.47: Data Downsampling on Gaussian Inverse Scattering for Discontinuous Toothed Ramp. Limit: Every 52nd point

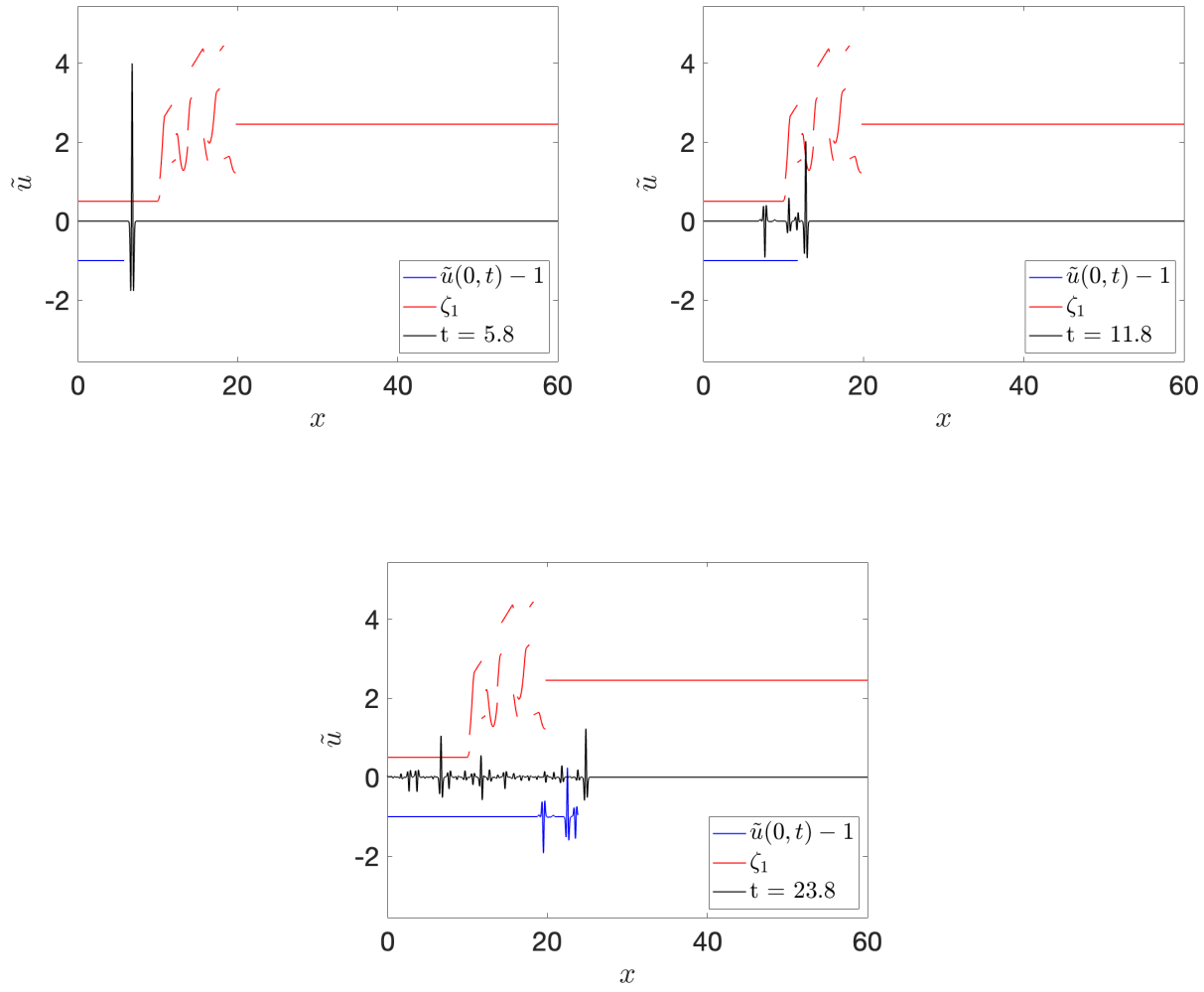
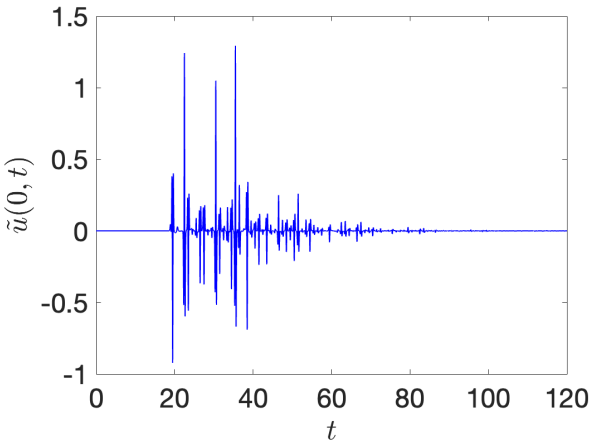
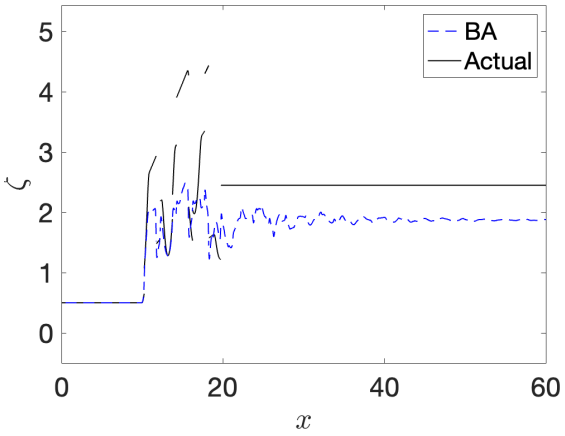


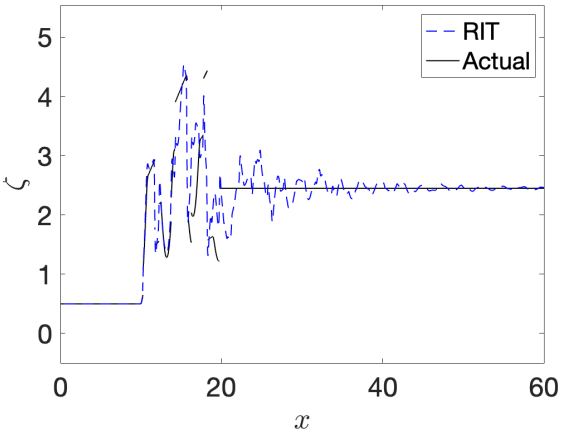
FIGURE 5.48: Top left: The Ricker source wavelet before passing through varying reflectivities. Top right: The Ricker source wavelet passing through varying reflectivities. Bottom: The Ricker source wavelet after passing through varying reflectivities.



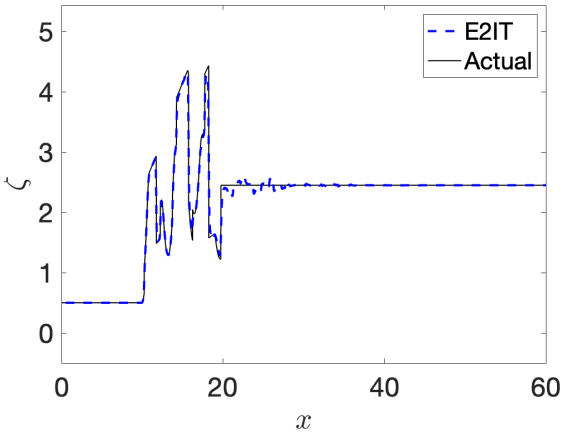
(A) Ricker measured data



(B) Born approximation



(C) Refined impedance transform

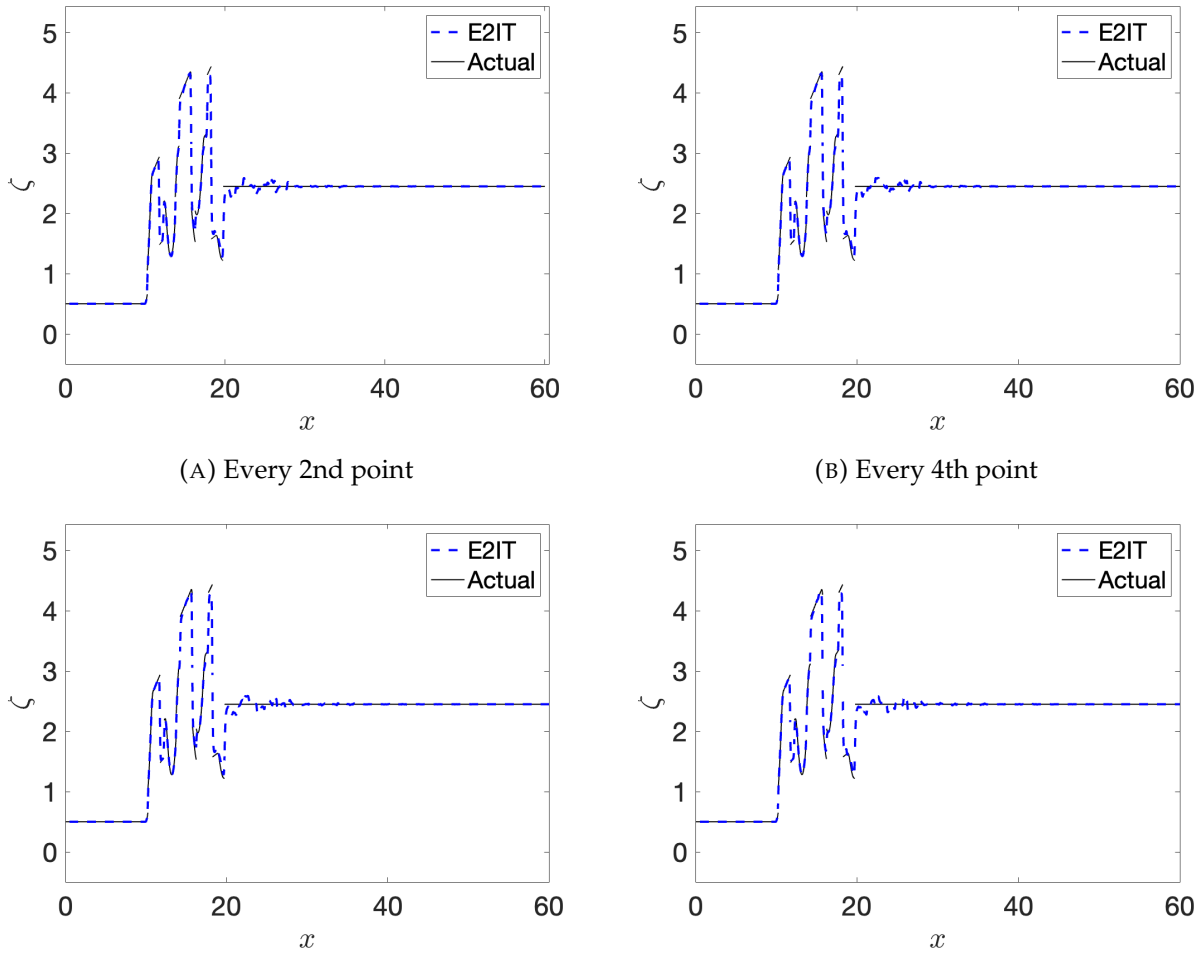


(D) Echoes-to-impedance transform

FIGURE 5.49: Scattering and Inverse Scattering of Ricker in Discontinuous Toothed Ramp

	l_2 Errors
BA	0.2990
RIT	0.2172
E2IT	0.1911

TABLE 5.11: l_2 Relative Errors for Ricker in Discontinuous Toothed Ramp



(A) Every 2nd point

(B) Every 4th point

(C) Every 8th point

(D) Every 16th point

FIGURE 5.50: Data Downsampling on Ricker Inverse Scattering for Discontinuous Toothed Ramp. Limit: Every 27th point

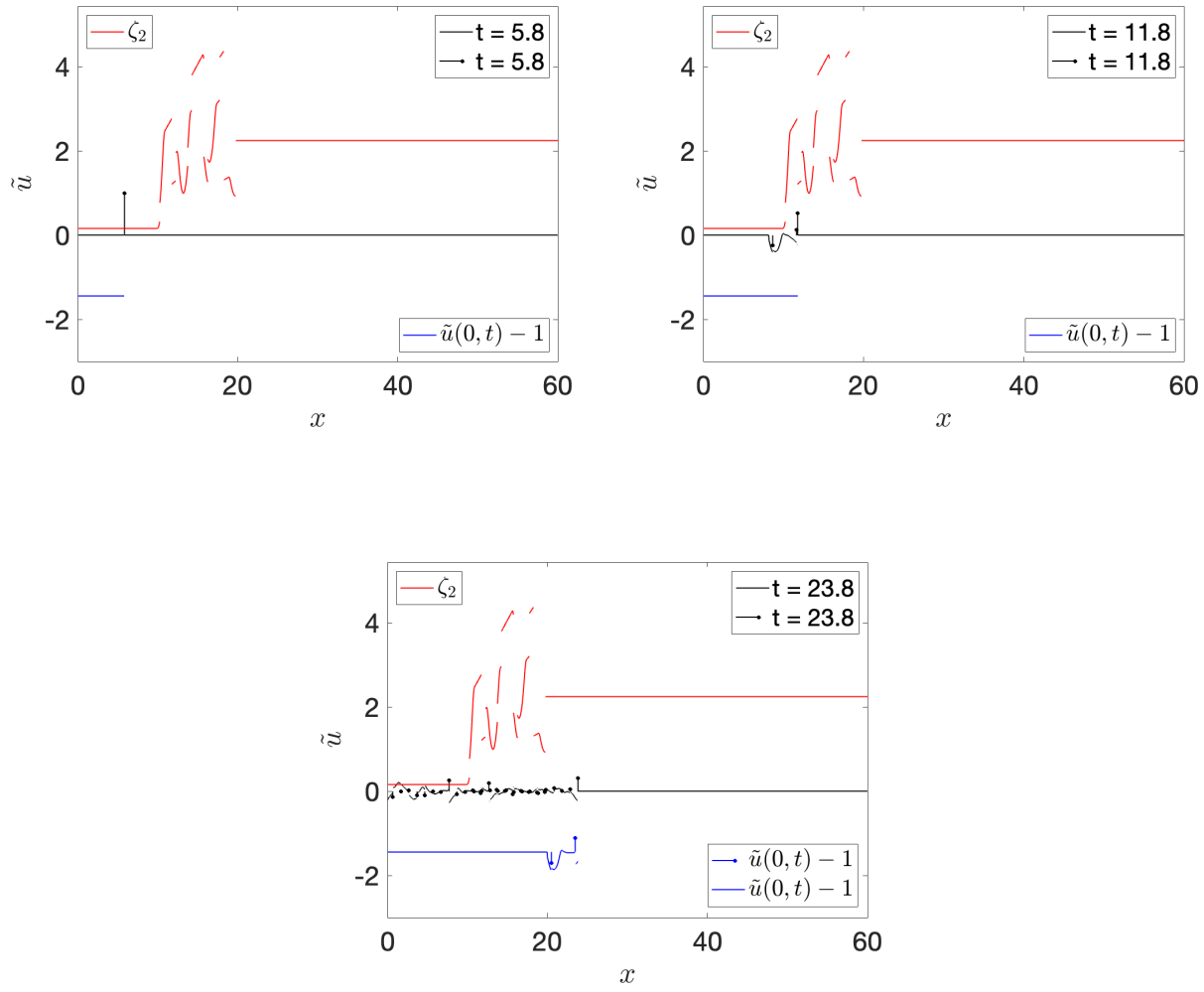
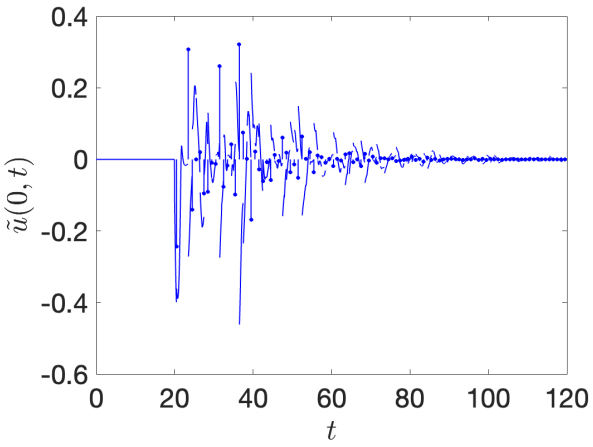
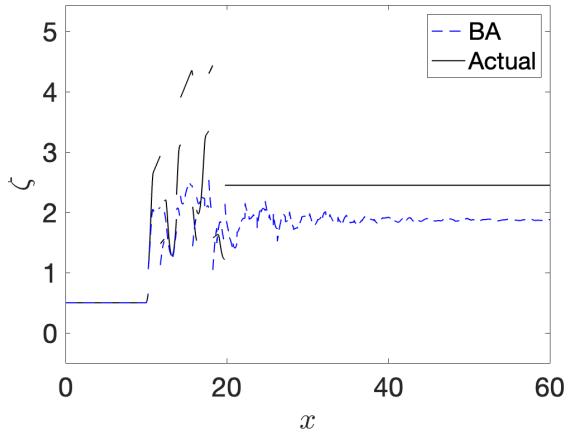


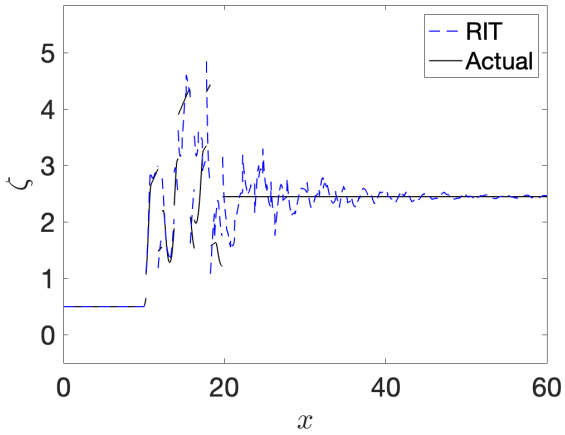
FIGURE 5.51: Top left: The Dirac source wavelet before passing through varying reflectivities. Top right: The Dirac source wavelet passing through varying reflectivities. Bottom: The Dirac source wavelet after passing through varying reflectivities.



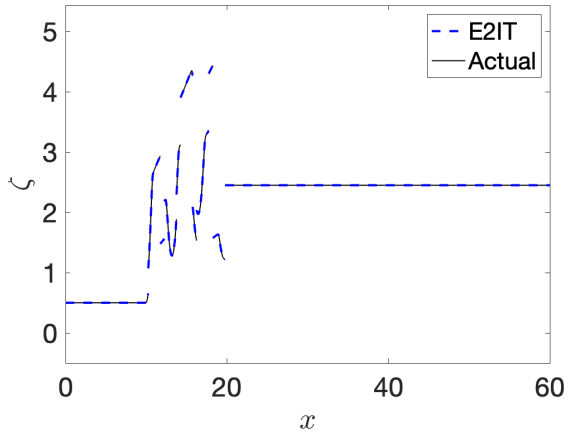
(A) Dirac measured data



(B) Born approximation



(C) Refined impedance transform

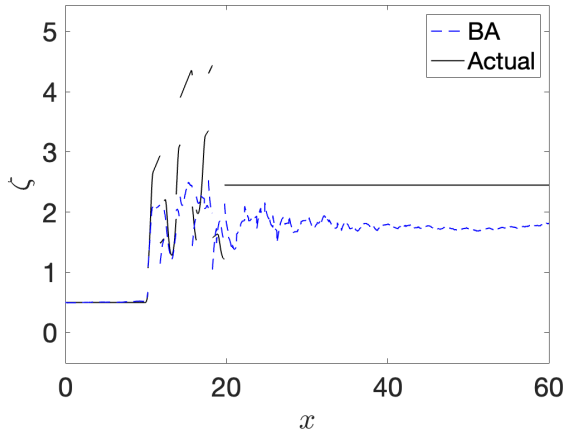


(D) Echoes-to-impedance transform

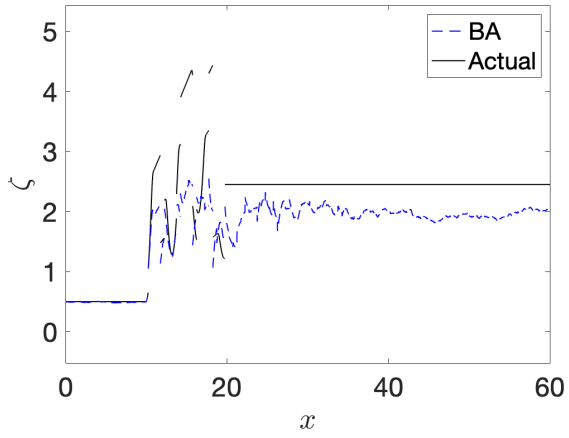
FIGURE 5.52: Scattering and Inverse Scattering of Dirac in Discontinuous Toothed Ramp

	l_2 Errors
BA	0.2735
RIT	0.1264
E2IT	**0.0014**

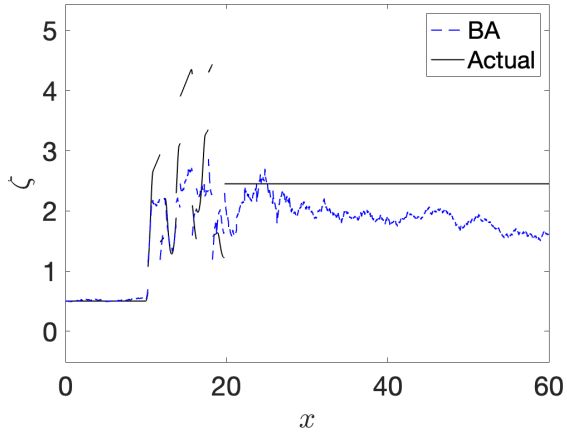
TABLE 5.12: l_2 Relative Errors of Dirac in Discontinuous Toothed Ramp



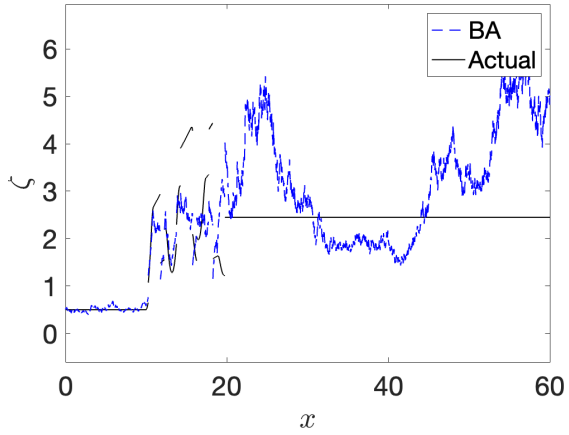
5% Gaussian Noise



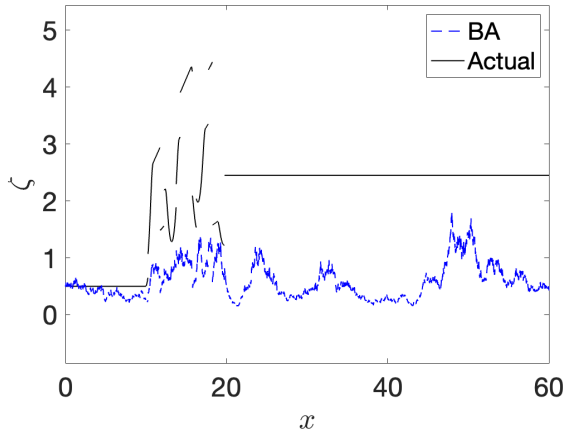
10% Gaussian Noise



25% Gaussian Noise



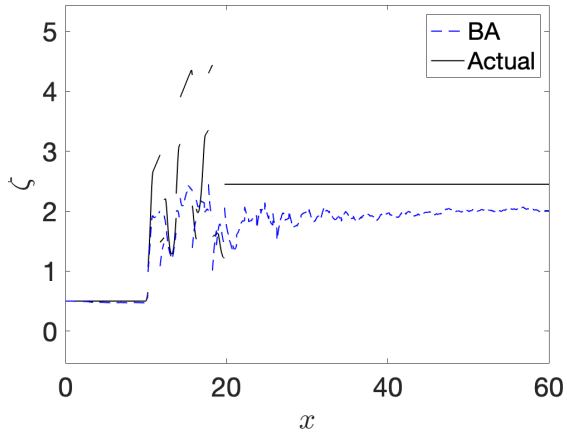
100% Gaussian Noise



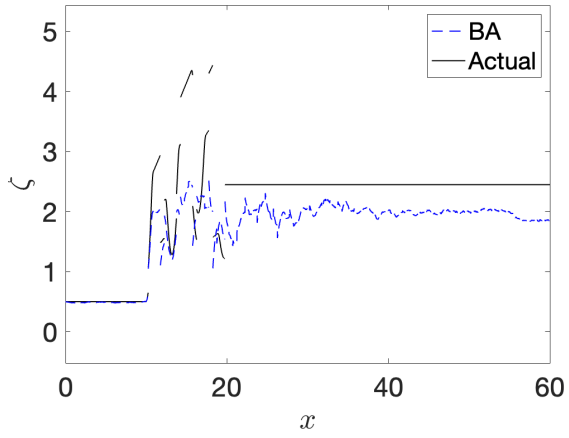
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.3018
10	0.2489
25	0.2581
100	0.5349
200	0.7584

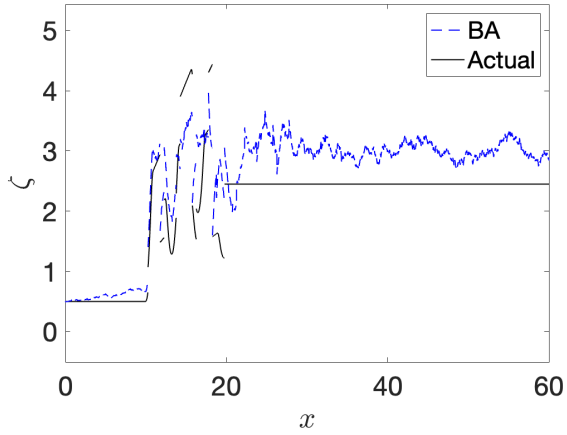
FIGURE 5.53: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



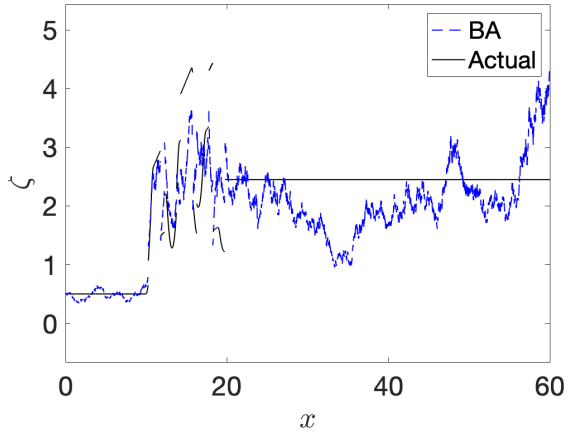
5% Uniform Noise



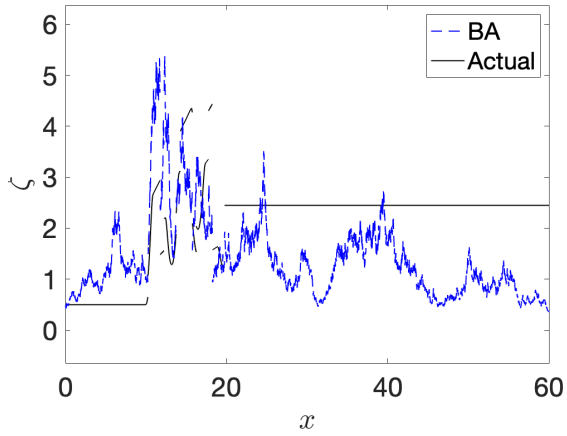
10% Uniform Noise



25% Uniform Noise



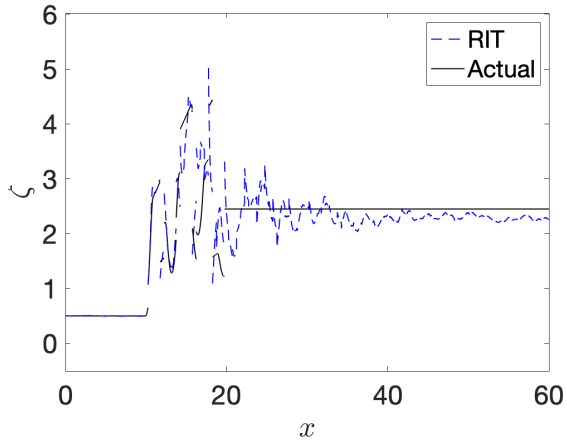
100% Uniform Noise



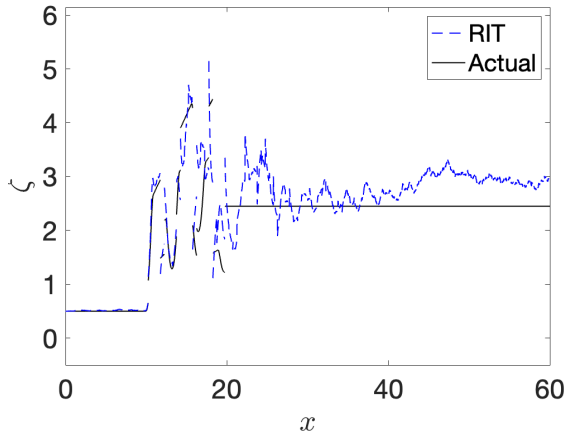
200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.2638
10	0.2500
25	0.2487
100	0.5144
200	0.5548

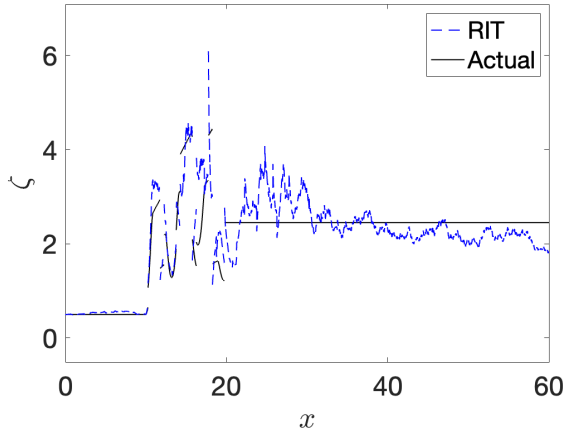
FIGURE 5.54: BA for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



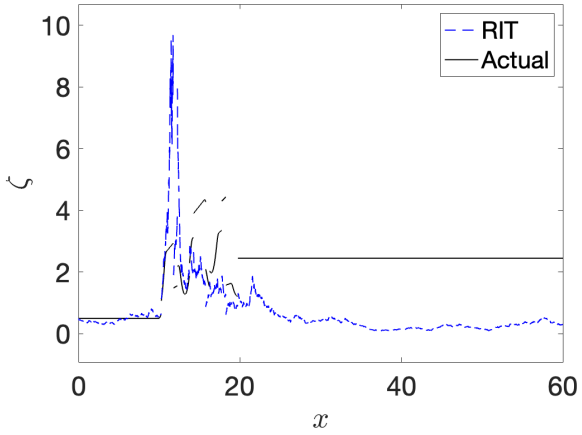
5% Gaussian Noise



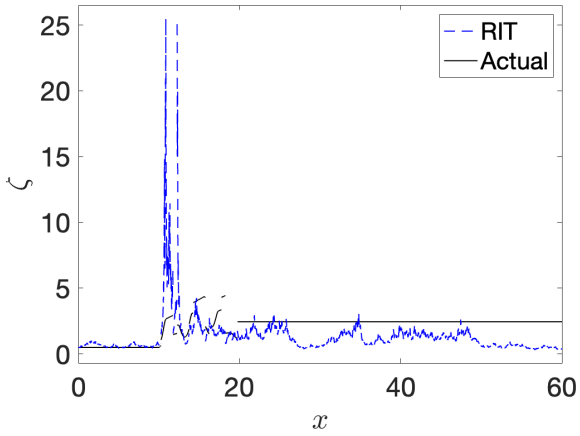
10% Gaussian Noise



25% Gaussian Noise



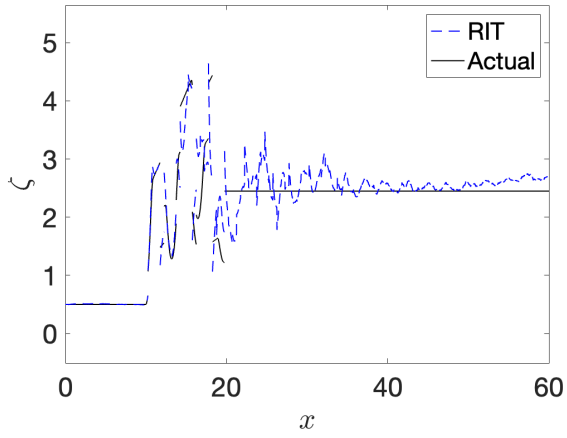
100% Gaussian Noise



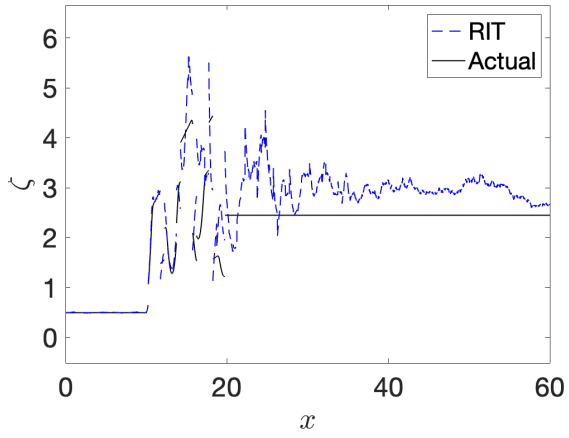
200% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.1389
10	0.1931
25	0.1868
100	0.8123
200	0.7500

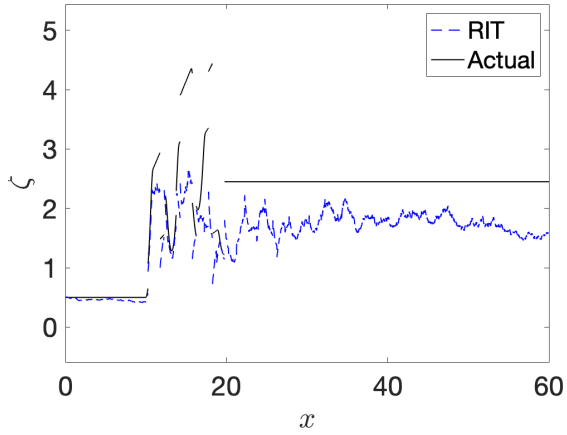
FIGURE 5.55: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



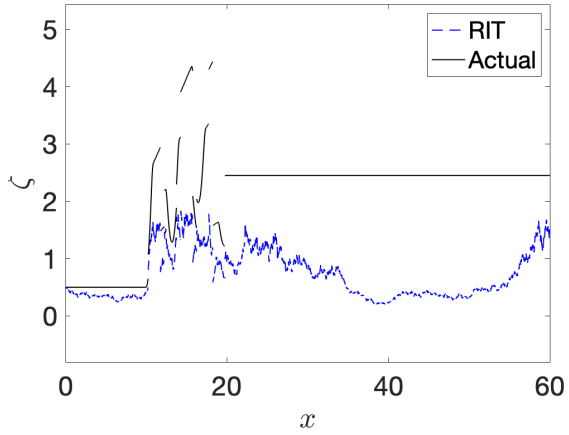
5% Uniform Noise



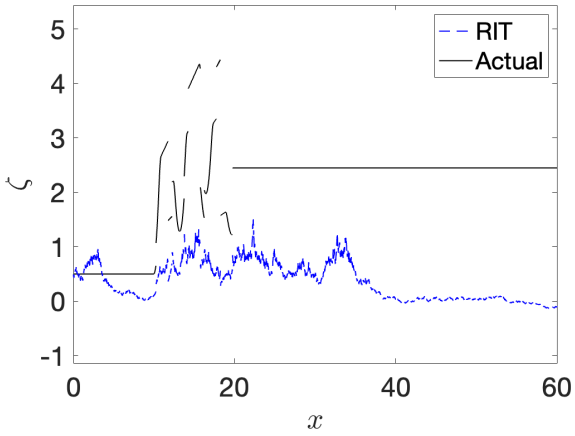
10% Uniform Noise



25% Uniform Noise



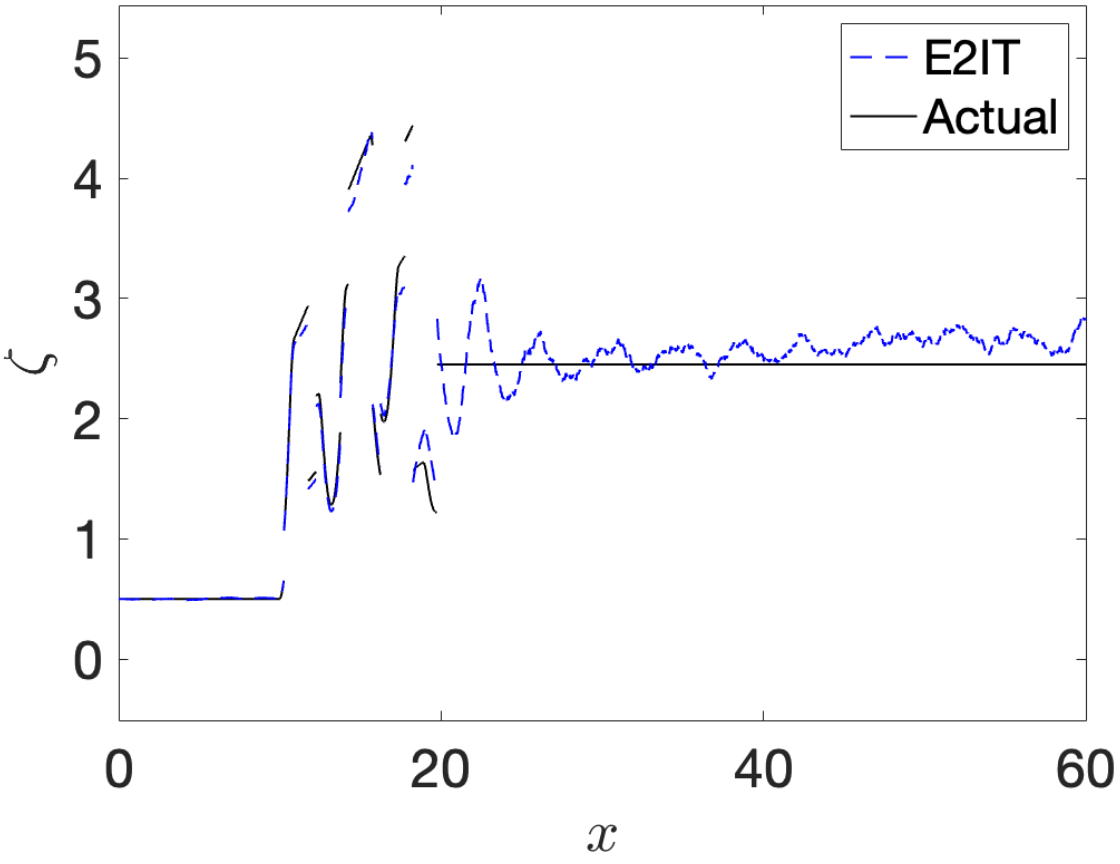
100% Uniform Noise



200% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.1346
10	0.2523
25	0.3253
100	0.6973
200	0.8622

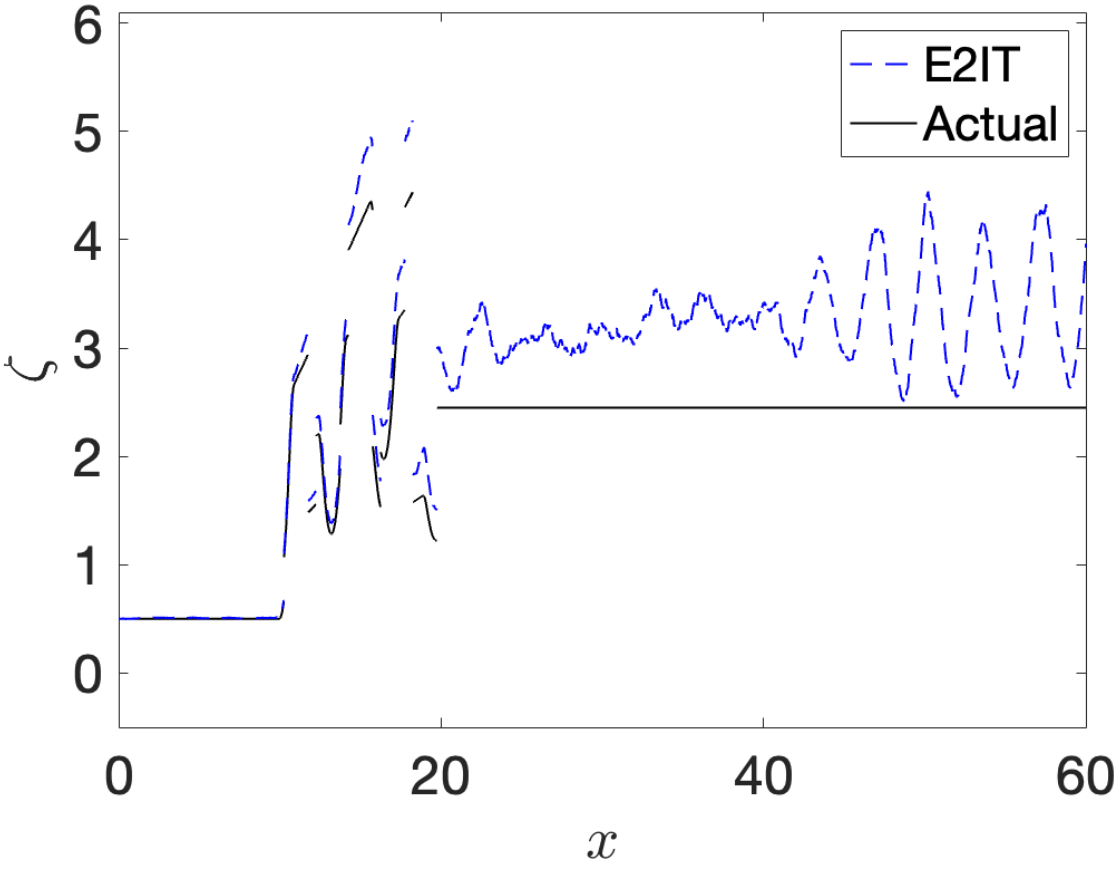
FIGURE 5.56: RIT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



5% Gaussian Noise

Gaussian Noise (%)	l_2 Relative Error
5	0.0798

FIGURE 5.57: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.



5% Uniform Noise

Uniform Noise (%)	l_2 Relative Error
5	0.3239

FIGURE 5.58: E2IT for Dirac Inverse Scattering with Figures and a Table of l_2 Relative Errors.

TABLE 5.13: l_2 Relative Errors and Computational Times for Smooth Ramp from Dirac Inverse Scattering

n	BA	RIT	E2IT
1000	0.0248	0.0029	6.6336E-4
2000	0.0248	0.0023	3.3175E-4
4000	0.0248	0.0021	1.6589E-4
8000	0.0248	0.0019	8.2950E-5
16000	0.0248	0.0019	4.1476E-5

(A) l_2 Relative Errors

n	BA	RIT	E2IT
1000	0.0049	0.0017	0.0662
2000	0.0096	0.0083	0.2416
4000	0.0083	0.0045	1.3367
8000	0.0166	0.0091	7.0221
16000	0.0144	0.0099	32.6738

(B) Computational Times (sec)

TABLE 5.14: l_2 Relative Errors and Computational Times for Toothed Ramp from Dirac Inverse Scattering

n	BA	RIT	E2IT
1000	0.2358	0.0417	0.0038
2000	0.2357	0.0401	0.0019
4000	0.2357	0.0395	0.0013
8000	0.2357	0.0392	4.7070E-4
16000	0.2357	0.0391	2.3536E-4

(A) l_2 Relative Errors

n	BA	RIT	E2IT
1000	0.0224	0.0026	0.0611
2000	0.0072	0.0031	0.1846
4000	0.0092	0.0044	1.2801
8000	0.0104	0.0070	7.4335
16000	0.0207	0.0142	32.3206

(B) Computational Times (sec)

TABLE 5.15: l_2 Relative Errors and Computational Times for Random Steps from Dirac Inverse Scattering

n	BA	RIT	E2IT
3000	0.0482	0.0476	4.6795E-16
15000	0.0453	0.0446	4.1542E-16

(A) l_2 Relative Errors for Random Steps

n	BA	RIT	E2IT
3000	0.0155	0.0069	0.6772
15000	0.0212	0.0155	27.0088

(B) Computational Times (sec)

TABLE 5.16: l_2 Relative Errors and Computational Times for Discontinuous Toothed Ramp from Dirac Inverse Scattering

n	BA	RIT	E2IT
3000	0.2735	0.1232	0.0014
15000	0.2730	0.1232	2.8596E-4

(A) l_2 Relative Errors for Discontinuous Toothed Ramp

n	BA	RIT	E2IT
3000	0.0306	0.0087	0.6839
15000	0.0602	0.0200	26.1341

(B) Computational Times (sec)

6 Conclusions

6.1 Summary

There are many confirmations on what was said in the previous chapters. We have verified that the echoes-to-impedance transform is the best inverse scattering algorithm, especially with the Dirac source wavelet, based on the evidence in its l_2 relative errors and inversion plots. Even though the running time grows at an exponential rate, it is not too slow in computing $\tilde{\zeta}$ and the change in its relative errors is the highest when n increases. Its relative errors involving noise is still also the lowest; however, the echoes-to-impedance transform seems to be very sensitive to noise, especially when ζ gets more complicated. A new discovery is the echoes-to-impedance transform can still invert high noisy data to $\tilde{\zeta}$ for the tested continuous ζ when the scattering-based algorithm uses the Dirac source wavelet. Finally, the echoes-to-impedance transform with data downsampling has no problem in plotting $\tilde{\zeta}$ that looks a lot like ζ.

The Born approximation is a quicker inverse scattering algorithm, but its accuracy is relatively low even when n increases. However, for its speed, seismologists may still use it as a protocol to determine what lies underneath a layered medium. It works fine with the smooth ramp and the random steps; but it starts to fail when it comes to the toothed ramp

and discontinuous toothed ramp. It is surprising to see that the Born approximation return lower l_2 relative errors using a regular source wavelet for a random step; nonetheless, when it is the Dirac source wavelet, the echoes-to-impedance transform gives the lowest l_2 relative error. Because of this, it seems that the Born approximation cannot work well when ζ becomes more oscillatory. It is the least sensitive to noise, which sometimes lowers its l_2 relative errors due to randomness considering it is not the most accurate when there is no noise.

The refined impedance transform is the quickest inverse scattering algorithm. In a competition of accuracy, it beats the Born approximation, but loses to the echoes-to-impedance transform. It defeats the Born approximation because it uses measurement operators to avoid single scattering. This combination of tactics allows it to compute a better $\tilde{\zeta}$ and can actually return the far side of ζ, which leads to lower l_2 relative errors. However, it loses to the echoes-to-impedance transform because it still uses the classical approach of reflection Green's function, which depends highly on ζ. Thus, as ζ gets more complicated, its accuracy declines. Its sensitivity to noise seems to fall between the Born approximation and the echoes-to-impedance transform, and like the Born approximation, may return better outcomes due to randomness since it is not perfect without noise.

Regardless of any inverse scattering algorithms, they all work best with the Dirac source wavelet. However, as ε increases, the Gaussian source wavelet begins to look like the Dirac source wavelet, which can improve its execution such that its l_2 relative errors get closer to the l_2 relative errors from a Dirac source wavelet. As noise increases, there are disfigurations with the original inverse scattering figure without noise, and so the l_2 relative errors did have an increasing pattern.

6.2 Limitations

The discussed methodologies only worked with (1.1), which is an initial value problem of a one-dimensional linear wave equation for a plane wave travelling through a layered medium. Further research is necessary to expand upon how it is possible to implement the algorithms for an initial value problem of three-dimensional and/or nonlinear wave equations. Moreover, the thesis compared the three inverse scattering algorithms using synthetic reflected data from the scattering-based algorithm where the acoustic impedance, ζ, of the layered medium is known. However, analysts normally deal with scattered data without knowing ζ, which means the scattering-based algorithm will not play into their research and it is not easy to check how well each inverse scattering algorithm works using (4.30). Finally, the methodologies can work with a Dirac source wavelet mathematically and always showed the best results; however, it is impossible to generate the Dirac delta function physically in practice.

Furthermore, analysts tend to work with methodologies that depend upon linear time-invariant (LTI) systems as opposed to linear time-variant (LTV) systems because the properties of linearity and time invariance simplify the process in acoustic imaging in [25]. LTI systems involve deconvolution, which can lead to huge errors in finding a more accurate acoustic impedance considering the band-limited nature of a wavelet. The echoes-to-impedance transform avoids this issue using the theory of OPUC; however, another issue to consider that the measured parameters can change over time as in a LTV systems. An example is NASA's non-destructive testing of the aluminum-brazed titanium honeycomb structure, which may develop flaws as it ages.

6.3 Current Research

Without the knowledge of ζ, if analysts have the scattered data and know the parameters k, w, ζ_l and Δ, then they can still implement each inverse scattering algorithm and model an approximated acoustic impedance of any layered medium. The means to use known ζ's for this thesis is to demonstrate that the echoes-to-impedance transform will still perform better than the other two inverse scattering algorithms regardless of knowing the acoustic impedance of a layered medium. Numerical experiments have provided evidence that the echoes-to-impeddance transform still returns better results with more complex ζ's of a layered medium, and this often occurs in realistic situations that scientists encounter from their sensor's recorded reflected data.

The scattering-based algorithm and the WKB approximation discussed in §2.4 are totally different methods. However, we note that the formula (3.8) for the coefficient of a travelling Dirac source wavelet, which is the singular part of the wave field from the scattering-based algorithm, also arises in the WKB approximation after applying a change of variable to (2.7b) [6]. Furthermore, it is possible to compare the numerical solution with the analytical solution of the wave equation and see how far off the scattering-based algorithm and the WKB approximation are away from each other.

Bibliography

[1] N. Bleistein, J. K. Cohen, and J. W. Stockwell Jr. *Mathematics of multidimensional seismic imaging, migration, and inversion*. Vol. 13. Interdisciplinary Applied Mathematics. Geophysics and Planetary Sciences. Springer-Verlag, New York, 2001, pp. xxx+510. ISBN: 0-387-95061-3. DOI: 10.1007/978-1-4613-0001-4. URL: https://doi-org.ezproxy.library.yorku.ca/10.1007/978-1-4613-0001-4.

[2] David P. Bourne et al. "An inverse problem for Voronoi diagrams: A simplified model of non-destructive testing with ultrasonic arrays". In: *Math. Methods Appl. Sci.* 44.5 (2021), pp. 3727–3745. ISSN: 0170-4214. DOI: 10.1002/mma.6977. URL: https://doi-org.ezproxy.library.yorku.ca/10.1002/mma.6977.

[3] Henri Calandra et al. "Recent advances in numerical methods for solving the wave equation in the context of seismic depth imaging". In: *SMAI J. Comput. Math.* S5 (2019), pp. 47–65. DOI: 10.5802/smai-jcm.51. URL: https://doi-org.ezproxy.library.yorku.ca/10.5802/smai-jcm.51.

[4] Kees van den Doel and U. Ascher. "Real-Time Numerical Solution of Webster's Equation on A Nonuniform Grid". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 16 (Sept. 2008), pp. 1163 –1172. DOI: 10.1109/TASL.2008.2001107.

[5] Markus Sebastian Doktor et al. "Characterization of steel buildings by means of non-destructive testing methods". In: *J. Math. Ind.* 8 (2018), Paper No. 10, 17. DOI:

10.1186/s13362-018-0052-5. URL: https://doi-org.ezproxy.library.yorku.ca/10.1186/s13362-018-0052-5.

[6] William C. (William Cronk) Elmore. *Physics of waves*. eng. New York: Dover Publications, 1985 - 1969. ISBN: 0486649261.

[7] Charles L. Epstein. *Introduction to the mathematics of medical imaging*. Second. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008, pp. xxxiv+761. ISBN: 978-0-89871-642-9. DOI: 10.1137/1.9780898717792. URL: https://doi-org.ezproxy.library.yorku.ca/10.1137/1.9780898717792.

[8] Elise C. Fear. "Microwave Imaging of the Breast". In: *Technology in Cancer Research & Treatment* 4.1 (2005). PMID: 15649090, pp. 69–82. DOI: 10.1177/153303460500400110. eprint: https://doi.org/10.1177/153303460500400110. URL: https://doi.org/10.1177/153303460500400110.

[9] Gerald B. Folland. *Fourier analysis and its applications*. The Wadsworth & Brooks/Cole Mathematics Series. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, 1992, pp. x+433. ISBN: 0-534-17094-3.

[10] A.P French. *Vibrations and Waves*. eng. MIT introductory physics series. CRC Press, 2017. ISBN: 0748744479.

[11] Peter C. Gibson. "A scattering-based algorithm for wave propagation in one dimension". In: *Numer. Methods Partial Differential Equations* 34.2 (2018), pp. 442–450. ISSN: 0749-159X. DOI: 10.1002/num.22207. URL: https://doi.org/10.1002/num.22207.

[12] Peter C. Gibson. "Acoustic imaging of layered media". In: *J. Comput. Phys.* 372 (2018), pp. 524–545. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2018.06.053. URL: https://doi.org/10.1016/j.jcp.2018.06.053.

[13] Peter C. Gibson. "On the measurement operator for scattering in layered media". In: *Inverse Probl. Imaging* 11.1 (2017), pp. 87–97. ISSN: 1930-8337. DOI: 10.3934/ipi.2017005. URL: https://doi-org.ezproxy.library.yorku.ca/10.3934/ipi.2017005.

[14] Peter C. Gibson. "The combinatorics of scattering in layered media". In: *SIAM J. Appl. Math.* 74.4 (2014), pp. 919–938. ISSN: 0036-1399. DOI: 10.1137/130923075. URL: https://doi-org.ezproxy.library.yorku.ca/10.1137/130923075.

[15] Peter C. Gibson. "The refined impedance transform for 1D acoustic reflection data". In: *Inverse Problems* 34.7 (2018), pp. 075013, 14. ISSN: 0266-5611. DOI: 10.1088/1361-6420/aac51d. URL: https://doi.org/10.1088/1361-6420/aac51d.

[16] D.J. Hagemaier. "NDT of aluminium-brazed titanium honeycomb structure". In: *Non-Destructive Testing* 9.3 (1976), pp. 107–116. ISSN: 0029-1021. DOI: https://doi.org/10.1016/0029-1021(76)90238-3. URL: https://www.sciencedirect.com/science/article/pii/0029102176902383.

[17] Siamak Hassanzadeh, ed. *Mathematical methods in geophysical imaging. III*. Vol. 2571. Proceedings. SPIE. SPIE—The International Society for Optical Engineering, Bellingham, WA, 1995, pp. viii+240. ISBN: 0-8194-1930-3.

[18] Kazufumi Ito and Bangti Jin. *Inverse problems*. Vol. 22. Series on Applied Mathematics. Tikhonov theory and algorithms. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2015, pp. x+318. ISBN: 978-981-4596-19-0.

[19] Andreas Kirsch. "The MUSIC algorithm and the factorization method in inverse scattering theory for inhomogeneous media". In: *Inverse Problems* 18.4 (2002), pp. 1025–1040. ISSN: 0266-5611. DOI: 10.1088/0266-5611/18/4/306. URL: https://doi-org.ezproxy.library.yorku.ca/10.1088/0266-5611/18/4/306.

[20] Andreas Kirsch and Natalia Grinberg. *The factorization method for inverse problems*. Vol. 36. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford, 2008, pp. xiv+201. ISBN: 978-0-19-921353-5.

[21] Institute of Medicine and National Research Council. *Mammography and Beyond: Developing Technologies for the Early Detection of Breast Cancer*. Ed. by Sharyl J. Nass, I. Craig Henderson, and Joyce C. Lashof. Washington, DC: The National Academies Press, 2001. ISBN: 978-0-309-21656-2. DOI: 10.17226/10030. URL: https://www.nap.edu/catalog/10030/mammography-and-beyond-developing-technologies-for-the-early-detection-of.

[22] Lauri Oksanen and Mikko Salo. "Inverse problems in imaging and engineering science [Editorial]". In: *Math. Eng.* 2.2 (2020), pp. 287–289. DOI: 10.3934/mine.2020014. URL: https://doi-org.ezproxy.library.yorku.ca/10.3934/mine.2020014.

[23] D. W. Oldenburg, T. Scheuer, and S. Levy. "Recovery of the acoustic impedance from reflection seismograms". In: *GEOPHYSICS* 48.10 (1983), pp. 1318–1337. DOI: 10.1190/1.1441413. eprint: https://doi.org/10.1190/1.1441413. URL: https://doi.org/10.1190/1.1441413.

[24] R. A Peterson, W. R Fillippone, and F. B Coker. "The Sythesis of Seisomograms from Well Log Data". In: *Geophysics* 20.3 (1955), pp. 516–538. ISSN: 0016-8033. DOI: 10.1190/1.1438155. URL: https://doi.org/10.1190/1.1438155.

[25] Charles L. Phillips. *Signals, systems, and transforms*. eng. 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2003. ISBN: 0130412074.

[26] Belén Riveiro and Mercedes Solla. *Non-Destructive Techniques for the Evaluation of Structures and Infrastructure*. eng. London: CRC Press, 2016. ISBN: 9781138028104.

[27] Gerard T. Schuster. *Seismic Imaging, Overview*. Ed. by Harsh K. Gupta. Dordrecht: Springer Netherlands, 2011, pp. 1121–1134. ISBN: 978-90-481-8702-7. DOI: 10.1007/978-90-481-8702-7_167. URL: https://doi.org/10.1007/978-90-481-8702-7_167.

[28] Ping Tong et al. "Acoustic wave-equation-based earthquake location". In: *Geophysical Journal International* 205.1 (Feb. 2016), pp. 464–478. ISSN: 0956-540X. DOI: 10.1093/gji/ggw026. eprint: https://academic.oup.com/gji/article-pdf/205/1/464/8036964/ggw026.pdf. URL: https://doi.org/10.1093/gji/ggw026.

[29] T.C. Williams, E.C. Fear, and D.T. Westwick. "Tissue sensing adaptive Radar for breast cancer detection-investigations of an improved skin-sensing method". In: *IEEE Transactions on Microwave Theory and Techniques* 54.4 (2006), pp. 1308–1314. DOI: 10.1109/TMTT.2006.871224.

[30] Smaine Zeroug and Sandip Bose. "Recent advances in the use of acoustics across the frequency spectrum in the oil and gas industry". In: *AIP Conference Proceedings* 1949.1 (2018), p. 020014. DOI: 10.1063/1.5031511. eprint: https://aip.scitation.org/doi/pdf/10.1063/1.5031511. URL: https://aip.scitation.org/doi/abs/10.1063/1.5031511.

[31] George Zweig and Christopher A. Shera. "The origin of periodicity in the spectrum of evoked otoacoustic emissions". In: *The Journal of the Acoustical Society of America* 98.4 (1995), pp. 2018–2047. DOI: 10.1121/1.413320. eprint: https://doi.org/10.1121/1.413320. URL: https://doi.org/10.1121/1.413320.

Appendices

A.1 Scattering-Based Algorithm

A.1.1 Regular Source Wavelet

```
close all
clear
clc

% Inputs
varepsilon = .1;
c = 0;
d = 30;
n = 3000;
T = 2*(d - c);
x_shift = 2;
w_scale = 1;

% Impedance Profile
zeta = @(x) toothedramp(x);

% Right-Travelling Wave
alpha = @(x) w_scale*ricker(varepsilon, x - x_shift);

% Left-Travelling Wave
beta = @(x) 0;

% Spatial Grid
delta = (d-c)/n;
for j = 1:n+1
    x(j) = c + (j-1)*delta;
```

```
end

% Temporal Grid
m = floor(T/delta);
for k = 1:m+1
    t(k) = (k-1)*delta;
end

% Weights
for j = 1:n
    r(j) = (zeta(x(j)) - zeta(x(j + 1)))/(zeta(x(j)) + zeta(x(j + 1)));
end

% Wave Field Preallocation
u_approx = zeros(m + 1, n + 1);

% One-way Wave Field Preallociation
v = zeros(m + 1, 2*n + 2);

% Initialization
for j = 1:n+1
    v(1,2*j-1) = alpha(x(j));
    v(1,2*j) = beta(x(j));
    u_approx(1,j) = v(1, 2*j - 1) + v(1, 2*j);
end

for k = 2:m+1
    % left-boundary
    v(k, 1) = alpha(x(1) - t(k));
    % right-boundary
    v(k, 2*n + 2) = beta(x(n+1) + t(k));
end

% Iterative Step
for k = 1:m
    for j = 2:n+1
        % right-moving component
        v(k+1, 2*j-1) = (1 + r(j-1))*v(k, 2*j-3) - r(j-1)*v(k, 2*j);
    end
    for j = 1:n
```

```
        % left-moving component
        v(k+1, 2*j) = r(j)*v(k, 2*j-1) + (1 - r(j))*v(k, 2*j+2);
    end
    for j = 1:n+1
        u_approx(k+1, j) = v(k+1, 2*j-1) + v(k+1, 2*j);
    end
end

% Discontinuity Points
dis = NaN;

% Measured Data
for i = 1:m + 1
    if i < 2
        reflectedwave(i) = 0;
    else
        reflectedwave(i) = u_approx(i,1);
    end
end

% Shift
y = 1:m+1;
y_shift = 1;

% Amplitude Scaling
scale = 100;

% Frame Setup
u_min = min(min(u_approx));
u_max = max(max(u_approx));
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
reflectedwave_min = min(scale*reflectedwave) - y_shift;
reflectedwave_max = max(scale*reflectedwave) + y_shift;
SBA_frame_min = min(min(u_min, zeta_min),reflectedwave_min) - 1;
SBA_frame_max = max(max(u_max, zeta_max), reflectedwave_max) + 1;

% Frame-by-Frame Wavefield Plot and Zeta Plot
for k = 1:10:(m+1) % for faster animations i.e. skip frames
    plot(t(1:k), scale*reflectedwave(y(1:k)) - y_shift)
```

```
    hold on
    myplot(dis, t, zeta(t), 'r');
    hold on
    plot(x, scale*u_approx(k,:), 'k');
    legend(sprintf('t = %.1f', t(k)));
    xlim([c 2*d]);
    ylim([SBA_frame_min SBA_frame_max]);
    pause(0);
    hold off
end

% Model of Measured Data
figure
plot(t, reflectedwave);
xlabel('t');
ylabel('$\tilde{u}(0,t)$','interpreter','latex');
```

A.1.2 Singular Source Wavelet for Smooth Impedance Profiles

```
close all
clear
clc

% Inputs
c = 0;
d = 25;
n = 3000;
T = 2*(d - c);

% Impedance Profile
zeta = @(x) modsawblade(x);

% Right-Travelling Wave
alpha = @(x) diracimpulse(x, c, d, n);

% Left-Travelling Wave
beta = @(x) 0;
```

```
% Spatial Grid
delta = (d-c)/n;
for j = 1:n+1
    x(j) = c + (j-1)*delta;
end

% Temporal Grid
m = floor(T/delta);
for k = 1:m+1
    t(k) = (k-1)*delta;
end

% Weights
for j = 1:n
    r(j) = (zeta(x(j)) - zeta(x(j+1)))/(zeta(x(j)) + zeta(x(j+1)));
end

% Wave Field Preallocation
u_approx = zeros(m+1, n+1);

% One-way Wave Field Preallociation
v = zeros(m+1, 2*n+2);

% Initialization
for j = 2:n+1
    v(1, 2*j-1) = alpha(x(j));
    v(1, 2*j) = beta(x(j));
    u_approx(1, j) = v(1, 2*j-1) + v(1, 2*j);
end

for k = 2:m+1
    % left-boundary
    v(k, 1) = alpha(x(1) - t(k));
    % right-boundary
    v(k, 2*n+2) = beta(x(n+1) + t(k));
end

% Iterative Step
for k = 1:m
    for j = 2:n+1
```

```
        % right-moving component
        v(k+1, 2*j-1) = (1 + r(j-1))*v(k, 2*j-3) - r(j-1)*v(k, 2*j);
    end
    for j = 1:n
        % left-moving component
        v(k+1, 2*j) = r(j)*v(k, 2*j-1) + (1-r(j))*v(k, 2*j+2);
    end
    for j = 1:n+1
        u_approx(k+1, j) = v(k+1, 2*j-1) + v(k+1, 2*j);
    end
end

% Discontinuity Points
dis = NaN;

% Adjustment for Dirac Source due to Disjoint Components
for j = 1:n/2
    x_reg(j) = (x(2*j - 1) + x(2*j))/2;
end

for k = 1:m+1
    for j = 1:n/2
        u_approx_reg(k, j) = (u_approx(k, 2*j-1) ...
            + u_approx(k, 2*j))/(2*delta);
    end
end

% Measured Data
for i = 1:m+1
    if i < 2
        reflectedwave(i) = 0;
    else
        reflectedwave(i) = u_approx_reg(i, 1);
    end
end

% Shift
y = 1:m+1;
y_shift = 1;
```

```
% Amplitude Scaling
scale = 1;

% Frame Setup
u_min = min(min(u_approx));
u_max = max(max(u_approx));
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
reflectedwave_min = min(scale*reflectedwave) - y_shift;
reflectedwave_max = max(scale*reflectedwave) + y_shift;
SBA_frame_min = min(min(u_min, zeta_min),reflectedwave_min) - 1;
SBA_frame_max = max(max(u_max, zeta_max), reflectedwave_max) + 1;

% Frame-by-Frame Wavefield Plot and Zeta Plot
for k = 1:20:m+1 % for faster animations i.e. skip frame
    z = u_approx_reg(k,:);
    clf
    hold on
    stem(t(k), sqrt(zeta(c)/zeta(c + t(k))), 'filled', ...
        'k', 'ShowBaseLine', 'off');
    plot(x_reg (x_reg < t(k)), scale*z (x_reg < t(k)), 'k');
    plot(x_reg (x_reg > t(k)), 0*z (x_reg > t(k)), 'k');
    plot(t(1:2:k), scale*reflectedwave(y(1:2:k)) - y_shift, 'b');
    plot(t, zeta(t), 'r');
    legend(sprintf('t = %.1f', t(k)));
    grid on
    xlim([c 2*d]);
    xlabel('x');
    ylim([SBA_frame_min SBA_frame_max]);
    pause(0);
    hold off
end

% Model of Measured Data
figure
plot(t(1:2:end), reflectedwave(1:2:end));
xlabel('t');
ylabel('$\tilde{u}(0,t)$','interpreter','latex');
```

Modified (Quick Method)

```
close all
clear
clc

% Inputs
c = 0;
d = 40;
delta = 0.0005;
T = 2*(d - c);

% Impedance Profile
zeta = @(x) toothedramp(x);

n = 2*ceil(T/(2*delta) - .5) + 1;
m = (n+1)/2;

% Spatial Grid
for i = 1:m+1
    x(i) = (i-1)*delta;
end

% Temporal Grid
for i = 1:m
    t(i) = 2*(i-1)*delta;
end

% Reflection Coefficients
for j = 1:m
    r(j) = (zeta(x(j)) - zeta(x(j+1)))/(zeta(x(j)) + zeta(x(j+1)));
end

% Initialization
p = floor((n-1)/4);
q = (n-1)/2;

w = zeros(m+1, 1);
v = zeros(m+1, 1);
```

```
data = zeros(m, 1);

% Iterative Steps
w(1) = r(1);
w(2) = 1 + r(1);
data(1) = w(1);


for j = 1:p
    for s = 1:j
        v(1) = 0;
        v(2*s) = r(2*s)*w(2*s) + (1-r(2*s))*w(2*s+1);
        v(2*s+1) = (1 + r(2*s))*w(2*s) - r(2*s)*w(2*s+1);
    end
    w = v;
    for s = 1:j+1
         v(2*s-1) = r(2*s-1)*w(2*s-1) + (1 - r(2*s-1))*w(2*s);
         v(2*s) = (1 + r(2*s-1))*w(2*s-1) - r(2*s-1)*w(2*s);
    end
    w = v;
    data(j+1) = w(1);
end

for j = p+1:q
    for s = 1:q+1-j
        v(1) = 0;
        v(2*s) = r(2*s)*w(2*s) + (1 - r(2*s))*w(2*s + 1);
    end
    for s = 1:q+1-j
        v(2*s+1) = (1 + r(2*s))*w(2*s) - r(2*s)*w(2*s + 1);
    end
    w = v;
    for s = 1:q+1-j
        v(2*s-1) = r(2*s-1)*w(2*s-1) + (1 - r(2*s-1))*w(2*s);
    end
    for s = 1:q+1-j
        v(2*s) = (1 + r(2*s-1))*w(2*s-1) - r(2*s-1)*w(2*s);
    end
    w = v;
```

```
    data(j+1) = w(1);
end

% Measured Data
data = data/(2*delta);

% Model of Measured Data
plot(t, data);
xlim([c T]);
xlabel('t');
ylabel('$\tilde{u}(0,t)$','interpreter','latex');
```

A.1.3 Singular Source Wavelet for Non-Smooth Impedance Profiles

```
close all
clear
clc

% Inputs
c = 0;
d = 30;
n = 1500;
T = 2*(d-c);

% Impedance Profile
zeta = @(x) sawblade(x - .25);

% Right-Travelling Wave
alpha = @(x) diracimpulse(x, c, d, n);

% Left-Travelling Wave
beta = @(x) 0;

% Spatial Grid
delta = (d-c)/n;
for j = 1:n+1
    x(j) = c + (j-1)*delta;
end
```

```
% Temporal Grid
m = floor(T/delta);
for k = 1:m+1
    t(k) = (k - 1)*delta;
end

% Weights
for j = 1:n
    r(j) = (zeta(x(j)) - zeta(x(j+1)))/(zeta(x(j)) + zeta(x(j+1)));
end

% Wave Field Preallocation
u_approx = zeros(m+1, n+1);

% One-way Wave Field Preallociation
v = zeros(m+1, 2*n+2);

% Initialization
for j = 2:n+1
    v(1, 2*j-1) = alpha(x(j));
    v(1, 2*j) = beta(x(j));
    u_approx(1, j) = (v(1, 2*j-1) + v(1, 2*j));
end

for k = 2:m+1
    % left-boundary
    v(k, 1) = alpha(x(1) - t(k));
    % right-boundary
    v(k, 2*n+2) = beta(x(n+1) + t(k));
end

% Iterative Step
for k = 1:m
    for j = 2:n+1
        % right-moving component
        v(k+1, 2*j-1) = (1 + r(j-1))*v(k, 2*j-3) - r(j-1)*v(k, 2*j);
    end
    for j = 1:n
        % left-moving component
```

```
        v(k+1, 2*j) = r(j)*v(k, 2*j-1) + (1 - r(j))*v(k, 2*j+2);
    end
    for j = 1:n+1
        u_approx(k+1, j) = v(k+1, 2*j-1) + v(k+1, 2*j);
    end
end

% Discontinuity Points
dis = [6.25 23.25];
p = round(dis/delta + .5);

% List of Reflections only at Discontinuities
r_ghost = zeros(1,length(r));
for j = 1:n
    for i = 1:length(p)
        if j == p(i)
            r_ghost(j) = r(p(i));
        end
    end
end

% Initialization for Intersection of Partial Tree with Whole Tree of
% Singular Part based on Ghost Reflections
for j = 2:n+1
    v_ghost(1, 2*j-1) = alpha(x(j));
    v_ghost(1, 2*j) = beta(x(j));
    u_ghost(1, j) = (v_ghost(1, 2*j-1) + v_ghost(1, 2*j));
end

% Iterative Step for Intersection of Partial Tree with Whole Tree of
% Singular Part based on Ghost Reflections
for k = 1:m
    for j = 2:n+1
        v_ghost(k+1, 2*j-1) = (1 + r_ghost(j-1))*v_ghost(k, 2*j-3) - ...
            r_ghost(j-1)*v_ghost(k, 2*j);
    end
    for j = 1:n
        v_ghost(k+1, 2*j) = r_ghost(j)*v_ghost(k, 2*j-1) + ...
            (1 - r_ghost(j))*v_ghost(k, 2*j+2);
    end
```

```
    for j = 1:n+1
        u_ghost(k+1, j) = v_ghost(k+1, 2*j-1) + v_ghost(k+1, 2*j);
    end
end

% Indices of the Intersection of Partial Tree with Whole Tree
% of Regular Part based on Ghost Reflection
[ii,jj] = find(u_ghost ~= 0);

% Dirac Impulse Preallocation
u_dirac = zeros(m+1, n+1);

% Extraction of u_approx Values for Dirac Impulses Based on the Indices of
% the Intersection of Partial Tree with Whole Tree based on Ghost
% Reflection
for i = 1:length(ii)
    u_dirac(ii(i), jj(i)) = u_approx(ii(i), jj(i));
end

% Change to Spatial Grid due to Disjoint Components
for j = 1:n/2
    x_reg(j) = (x(2*j-1) + x(2*j))/2;
end

% Change to u_approx due to Disjoint Components
for k = 1:m+1
    for j = 1:n/2
        u_approx_reg(k, j) = (u_approx(k, 2*j-1) + ...
            u_approx(k, 2*j))/(2*delta);
    end
end

% Iterative Step to Find Intersection of Partial Grid of u_dirac with
% Whole Grid u_dirac_reg
for k = 1:m+1
    for j = 1:n/2
        u_approx_sing(k, j) = u_dirac(k, 2*j-1) + u_dirac(k, 2*j);
    end
end
```

```
% Indices of the Intersections of Partial Tree and Whole Tree of Singular
% Part
[II, JJ] = find(u_approx_sing ~= 0);

% Intersection of u_approx_sing and u_approx_reg to Set those
% Peaks in u_approx_reg to Disappear
for i = 1:length(II)
    u_approx_reg(II(i), JJ(i)) = NaN;
end

regular = u_approx_reg;
singular = u_approx_sing;

for i = 1:length(II)
    regular(II(i), JJ(i)) = singular(II(i), JJ(i));
end
u_approx_combined = regular;

% Measured Data for Regular Part
for i = 1:m + 1
    if i < 2
        reflectedwave(i) = 0;
    else
        reflectedwave(i) = u_approx_reg(i, 1);
    end
end

% Measured Data for Singular Part
for i = 1:m+1
    if i < 2
        dirac_reflectedwave(i) = 0;
    else
        dirac_reflectedwave(i) = u_dirac(i, 1);
    end
end

% Singular Spatial Points
x_sing = x;

% Shift
```

```
y = 1:m+1;
y_shift = 1;

% Frame Setup
u_min = min(min(u_approx));
u_max = max(max(u_approx));
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
reflectedwave_min = min(reflectedwave) - y_shift;
reflectedwave_max = max(reflectedwave) + y_shift;
SBA_frame_min = min(min(u_min, zeta_min),reflectedwave_min) - 1;
SBA_frame_max = max(max(u_max, zeta_max), reflectedwave_max) + 1;

% Frame-by-Frame Wavefield Plot and Zeta Plot
for k = 1:20:m+1 % for faster animations i.e. skip frame
    z0 = u_approx_reg(k,:);
    z1 = u_dirac(k,:);
    clf
    hold on
    axis off
    ax(1) = axes;
    plot(x_reg, z0, 'k');
    if length(dis) == 1
        legend(sprintf('t = %.1f', t(k)));
    end
    grid on

    time = t(1:2:k);
    ref = reflectedwave(y(1:2:k));
    dirac_ref = dirac_reflectedwave(y(1:2:k));
    hold on

    stem1 = stem(ax(1), x_sing (z1 ~= 0), z1 (z1 ~= 0), 'filled', ...
        'k', 'MarkerSize', 3, 'DisplayName', sprintf('t = %.1f', t(k)));
    set(stem1, 'ShowBaseLine', 'off');
    lgd = legend(stem1);
    lgd.FontSize = 14;

    ax(2) = axes;
    axis off
```

```
    stem2 = stem(ax(2),time (dirac_ref ~= 0), ...
        dirac_ref (dirac_ref ~= 0) - y_shift, 'filled', 'b', ...
        'MarkerSize', 3, 'BaseValue', -y_shift);
    axis off
    hold on
    plot(ax(2),time, ref - y_shift, 'b');
    axis off
    set(stem2, 'ShowBaseLine', 'off');

    ax(3) = axes;
    myplot(dis, t, zeta(t), 'r');

    axis off
    limits = cell2mat(get(ax,'YLim')); % get both axes limits
    set(ax,'YLim',[-2 4]); % set the same values for both axes
    set(ax,'XLim',[c 2*d]);
    linkaxes(ax)
    axis off
    grid on
    pause(0);
    axis off
    hold off
end

% Model of Measured Data
figure
plot(t(1:2:end), reflectedwave(1:2:end), 'b');
hold on
stem(t (dirac_reflectedwave ~= 0), ...
    dirac_reflectedwave (dirac_reflectedwave ~= 0), 'filled', 'b', ...
    'ShowBaseLine', 'off');
xlabel('t');
ylabel('$\tilde{u}(0,t)$','interpreter','latex');
```

B.1 Born Approximation

B.1.1 Regular Source Wavelet

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 2;
w = -1*w_scale;
f = u_approx(:,1)/w;

% Temporal Grid
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
end

if k > 0
    for j = 1:k
        f = delta*cumsum(f);
    end
end
riem_dist = delta*cumsum(f);

% Born Approximation
born_approx = zeta_x0*exp(-2*riem_dist);

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
born_approx_min = min(born_approx);
born_approx_max = max(born_approx);
born_approx_frame_min = min(zeta_min, born_approx_min) - 1;
```

```
born_approx_frame_max = max(zeta_max, born_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k')
hold on
myplot(dis, x + delta/2, born_approx, 'b--');
ylim([born_approx_frame_min born_approx_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');

% Relative Error
BA_relative_error = norm(born_approx - zeta(x)')/norm(zeta(x)');
```

B.1.2 Singular Source Wavelet

No Noise

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
f = u_approx(:,1)/delta;

% Temporal Grid
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
```

```
end

if k > 0
    for j = 1:k
        f = delta*cumsum(f);
    end
end
riem_dist = delta*cumsum(f);

% Born Approximation
born_approx = zeta_x0*exp(-2*riem_dist);

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
born_approx_min = min(born_approx);
born_approx_max = max(born_approx);
BADirac_frame_min = min(zeta_min, born_approx_min) - 1;
BADirac_frame_max = max(zeta_max, born_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k')
hold on
myplot(dis, x + delta/2, born_approx, 'b--');
ylim([BADirac_frame_min BADirac_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');

% Relative Error
BA_relative_error = norm(born_approx - zeta(x)')/norm(zeta(x)');
```

Gaussian Noise

```
clc

% Inputs
```

```
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
f = u_approx(:,1)/delta;

% Create Gaussian Noise
gaussian_noise = randn(length(u_approx(:,1)), 1);

% Rescale Noise for Desired Proportion
p = 0.1;
guassian_noise_rescaled = p*norm(f)/norm(gaussian_noise)*gaussian_noise;

% Data with Noise
f_noise = f + guassian_noise_rescaled;

% Temporal Grid
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
end

if k > 0
    for j = 1:k
        f_noise = delta*cumsum(f_noise);
    end
end
riem_dist = delta*cumsum(f_noise);

% Born Approximation
born_approx = zeta_x0*exp(-2*riem_dist);

% Frame Setup
zeta_min = min(zeta(x));
```

```
zeta_max = max(zeta(x));
born_approx_min = min(born_approx);
born_approx_max = max(born_approx);
BADirac_frame_min = min(zeta_min, born_approx_min) - 1;
BADirac_frame_max = max(zeta_max, born_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k')
hold on
myplot(dis, x + delta/2, born_approx, 'b--');
ylim([BADirac_frame_min BADirac_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');

% Relative Error
BA_relative_error = norm(born_approx - zeta(x)')/norm(zeta(x)');
```

Uniform Noise

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
f = u_approx(:,1)/delta;

% Create Uniform Noise
uniform_noise = -1 + 2*rand(length(u_approx(:,1)), 1);

% Rescale Noise for Desired Proportion
p = 0.5;
uniform_noise_rescaled = p*norm(f)/norm(uniform_noise)*uniform_noise;

% Data with Noise
```

```
f_noise = f + uniform_noise_rescaled;

% Temporal Grid
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
end

if k > 0
    for j = 1:k
        f_noise = delta*cumsum(f_noise);
    end
end
riem_dist = delta*cumsum(f_noise);

% Born Approximation
born_approx = zeta_x0*exp(-2*riem_dist);

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
born_approx_min = min(born_approx);
born_approx_max = max(born_approx);
BADirac_frame_min = min(zeta_min, born_approx_min) - 1;
BADirac_frame_max = max(zeta_max, born_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k')
hold on
myplot(dis, x + delta/2, born_approx, 'b--');
ylim([BADirac_frame_min BADirac_frame_max]);
xlim([c d]);
xlabel('x');
```

```
ylabel('$\zeta$','interpreter','latex');

% Relative Error
BA_relative_error = norm(born_approx - zeta(x)')/norm(zeta(x)');
```

C.1 Refined Impedance Transform

C.1.1 Regular Source Wavelet

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1*w_scale;
f = u_approx(:,1)/w;

% Temporal Grid
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
end

if k > 0
    for j = 1:k
        f = delta*cumsum(f);
    end
end
riem_dist = delta*cumsum(f);
```

```
% Refined Impedance Transform
ref_zeta_approx = zeta_x0*((w - riem_dist)./(w + riem_dist));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
ref_zeta_approx_min = min(ref_zeta_approx);
ref_zeta_approx_max = max(ref_zeta_approx);
RIT_frame_min = min(zeta_min, ref_zeta_approx_min) - 1;
RIT_frame_max = max(zeta_max, ref_zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'r')
hold on
myplot(dis, x + delta/2, ref_zeta_approx, 'b--');
ylim([RIT_frame_min RIT_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');

% Relative Error
RIT_relative_error = norm(ref_zeta_approx - zeta(x)')/norm(zeta(x)');
```

C.1.2 Singular Source Wavelet

No Noise

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
f = u_approx(:,1)/delta;

% Temporal Grid
```

```
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
end

if k > 0
    for j = 1:k
        f = delta*cumsum(f);
    end
end
riem_dist = delta*cumsum(f);

% Refined Impedance Transform
ref_zeta_approx = zeta_x0*((w - riem_dist)./(w + riem_dist));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
ref_zeta_approx_min = min(ref_zeta_approx);
ref_zeta_approx_max = max(ref_zeta_approx);
RITDirac_frame_min = min(zeta_min, ref_zeta_approx_min) - 1;
RITDirac_frame_max = max(zeta_max, ref_zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k')
hold on
myplot(dis, x + delta/2, ref_zeta_approx, 'b--');
ylim([RITDirac_frame_min RITDirac_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');

% Relative Error
```

```
RIT_relative_error = norm(ref_zeta_approx - zeta(x)')/norm(zeta(x)');
```

Gaussian Noise

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
f = u_approx(:,1)/delta;

% Create Gaussian Noise
gaussian_noise = randn(length(u_approx(:,1)), 1);

% Rescale Gaussian Noise for Desired Proportion
p = 0.5;
guassian_noise_rescaled = p*norm(f)/norm(gaussian_noise)*gaussian_noise;

% Data with Gaussian Noise
f_noise = f + guassian_noise_rescaled;

% Temporal Grid
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
end

if k > 0
    for j = 1:k
        f_noise = delta*cumsum(f_noise);
```

```
    end
end
riem_dist = delta*cumsum(f_noise);

% Refined Impedance Transform
ref_zeta_approx = zeta_x0*((w - riem_dist)./(w + riem_dist));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
ref_zeta_approx_min = min(ref_zeta_approx);
ref_zeta_approx_max = max(ref_zeta_approx);
RITDirac_frame_min = min(zeta_min, ref_zeta_approx_min) - 1;
RITDirac_frame_max = max(zeta_max, ref_zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k')
hold on
myplot(dis, x + delta/2, ref_zeta_approx, 'b--');
ylim([RITDirac_frame_min RITDirac_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');

% Relative Error
RIT_relative_error = norm(ref_zeta_approx - zeta(x)')/norm(zeta(x)');
```

Uniform Noise

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
f = u_approx(:,1)/delta;
```

```
% Create Uniform Noise
uniform_noise = -1 + 2*rand(length(u_approx(:,1)), 1);

% Rescale Uniform Noise for Desired Proportion
p = 0;
uniform_noise_rescaled = p*norm(f)/norm(uniform_noise)*uniform_noise;

% Data with Uniform Noise
f_noise = f + uniform_noise_rescaled;

% Temporal Grid
m = floor(T/delta);
for K = 1:m+1
    t(K) = (K - 1)*delta;
end

% Spatial Grid
m = floor(T/delta);
for K = 1:m+1
    x(K) = (K - 1)*delta/2;
end

if k > 0
    for j = 1:k
        f_noise = delta*cumsum(f_noise);
    end
end
riem_dist = delta*cumsum(f_noise);

% Refined Impedance Transform
ref_zeta_approx = zeta_x0*((w - riem_dist)./(w + riem_dist));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
ref_zeta_approx_min = min(ref_zeta_approx);
ref_zeta_approx_max = max(ref_zeta_approx);
RITDirac_frame_min = min(zeta_min, ref_zeta_approx_min) - 1;
RITDirac_frame_max = max(zeta_max, ref_zeta_approx_max) + 1;
```

```
% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k')
hold on
myplot(dis, x + delta/2, ref_zeta_approx, 'b--');
ylim([RITDirac_frame_min RITDirac_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');

% Relative Error
RIT_relative_error = norm(ref_zeta_approx - zeta(x)')/norm(zeta(x)');
```

D.1 Echoes-to-Impedance Transform

D.1.1 Regular Source Wavelet

No Data Downsampling

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1*w_scale;
N = length(u_approx);
a = u_approx(:,1);

% Initialization
for j = 1:N
    x(j) = x0 + j*delta/2;
end

% Preallocation
U = zeros(N, N);
b = zeros(N, 1);
```

```
for i = 1:N
    for j = 1:N
        if i < j
            U(i, j) = 0;
        else
            U(i, j) = 1;
        end
    end
end

b = (delta^(k+1)/w)*U^k*a;

 for Loop for Elementwise m
m = zeros(N + 1,1);
m(1) = 1;

for j = 1:N
    m(j + 1) = dot(m(1:j), conj(flip(b(1:j))));
end
m(1) = 1;

q = zeros(N + 1, 1);
q(1) = 1;
r = zeros(N, 1);

for j = 1:N
    q_truncate =  q(1:find(q,1,'last'));
    q_reverse = flip(q_truncate');
    q_conjugate = conj(q_reverse);
    q_conjugate(numel(q)) = 0;
    q_star = q_conjugate;
    Sq = circshift(q,1);
    r(j) = dot(m, Sq)/dot(m, q_star');
    if j == N
        break
    end
    q = Sq' - r(j)*q_star;
end
```

```
% Echoes-to-Impedance Transform
zeta_approx = zeta_x0*exp(-2*U*atanh(r));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
zeta_approx_min = min(zeta_approx);
zeta_approx_max = max(zeta_approx);
E2IT_frame_min = min(zeta_min, zeta_approx_min) - 1;
E2IT_frame_max = max(zeta_max, zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
myplot(dis, x, zeta(x), 'k');
hold on
myplot(dis, x + x_shift/2, zeta_approx, 'b--');
ylim([E2IT_frame_min E2IT_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');
set(findobj(gca, 'Type', 'Line', 'Linestyle', '--'), 'LineWidth', 2);

% Relative Error
E2IT_relative_error = norm(zeta_approx - zeta(x)')/norm(zeta(x)');
```

Data Downsampling

```
clc
clear vars x

x0 = 0;
zeta_x0 = zeta(0);
k = 2;
w = -1*w_scale;

% Data Downsampling Setup
% p > 1 must be an integer.
p = 2;
N = length(reflectedwave(1:p:end));
```

```
a = reflectedwave(1:p:end)';

% Change in Delta
delta_fraunhofer = p*delta;

% Initialization
for j = 1:N
    x(j) = x0 + j*delta_fraunhofer/2;
end

% Preallocation
U = zeros(N, N);
b = zeros(N, 1);

for i = 1:N
    for j = 1:N
        if i < j
            U(i, j) = 0;
        else
            U(i, j) = 1;
        end
    end
end

b = (delta_fraunhofer^(k+1)/w)*U^k*a;

%for Loop for Elementwise m
m = zeros(N + 1,1);
m(1) = 1;

for j = 1:N
    m(j + 1) = dot(m(1:j), conj(flip(b(1:j))));
end
m(1) = 1;

q = zeros(N + 1, 1);
q(1) = 1;
r = zeros(N, 1);

for j = 1:N
```

```
    q_truncate =  q(1:find(q,1,'last'));
    q_reverse = flip(q_truncate');
    q_conjugate = conj(q_reverse);
    q_conjugate(numel(q)) = 0;
    q_star = q_conjugate;
    Sq = circshift(q,1);
    r(j) = dot(m, Sq)/dot(m, q_star');
    if j == N
        break
    end
    q = Sq' - r(j)*q_star;
end

% Echoes-to-Impedance Transform
zeta_approx = zeta_x0*exp(-2*U*atanh(r));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
zeta_approx_min = min(zeta_approx);
zeta_approx_max = max(zeta_approx);
E2IT_frame_min = min(zeta_min, zeta_approx_min) - 1;
E2IT_frame_max = max(zeta_max, zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k');
hold on
myplot(dis, x + x_shift/2, zeta_approx, 'b--');
ylim([E2IT_frame_min E2IT_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');
set(findobj(gca, 'Type', 'Line', 'Linestyle', '--'), 'LineWidth', 2);

% Relative Error
E2IT_relative_error = norm(zeta_approx - zeta(x)')/norm(zeta(x)');
```

D.1.2 Singular Source Wavelet

No Noise with No Data Downsampling

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
N = length(u_approx);
a = u_approx(:,1)/delta;

% Initialization
for j = 1:N
    x(j) = x0 + j*delta/2;
end

% Preallocation
U = zeros(N, N);
b = zeros(N, 1);

for i = 1:N
    for j = 1:N
        if i < j
            U(i, j) = 0;
        else
            U(i, j) = 1;
        end
    end
end

b = (delta^(k+1)/w)*U^k*a;

%for Loop for Elementwise m
m = zeros(N + 1,1);
m(1) = 1;

for j = 1:N
```

```
    m(j + 1) = dot(m(1:j), conj(flip(b(1:j))));
end
m(1) = 1;

q = zeros(N + 1, 1);
q(1) = 1;
r = zeros(N, 1);

for j = 1:N
    q_truncate =  q(1:find(q,1,'last'));
    q_reverse = flip(q_truncate');
    q_conjugate = conj(q_reverse);
    q_conjugate(numel(q)) = 0;
    q_star = q_conjugate;
    Sq = circshift(q,1);
    r(j) = dot(m, Sq)/dot(m, q_star');
    if j == N
        break
    end
    q = Sq' - r(j)*q_star;
end

% Echoes-to-Impedance Transform
zeta_approx = zeta_x0*exp(-2*U*atanh(r));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
zeta_approx_min = min(zeta_approx);
zeta_approx_max = max(zeta_approx);
E2ITDirac_frame_min = min(zeta_min, zeta_approx_min) - 1;
E2ITDirac_frame_max = max(zeta_max, zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k');
hold on
myplot(dis, x, zeta_approx, 'b--');
ylim([E2ITDirac_frame_min E2ITDirac_frame_max]);
xlim([c d]);
```

```
xlabel('x', 'fontweight','bold');
ylabel('$\zeta$','interpreter','latex');
set(findobj(gca, 'Type', 'Line', 'Linestyle', '--'), 'LineWidth', 2);

% Relative Error
E2IT_relative_error = norm(zeta_approx - zeta(x)')/norm(zeta(x)');
```

No Noise with Data Downsampling

```
clc
clear vars x;

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;

% Data Downsampling Setup
% y > 1 must be an integer
y = 2;
p = 2*y - 1;
u = u_approx(:,1)/delta;
N = length(u(1:p:end));
a = u(1:p:end);

% Change in Delta
delta_fraunhofer = p*delta;

% Initialization
for j = 1:N
    x(j) = x0 + j*delta_fraunhofer/2;
end

% Preallocation
U = zeros(N, N);
b = zeros(N, 1);
```

```
for i = 1:N
    for j = 1:N
        if i < j
            U(i, j) = 0;
        else
            U(i, j) = 1;
        end
    end
end

b = (delta_fraunhofer^(k+1)/w)*U^k*a;

%for Loop for Elementwise m
m = zeros(N + 1,1);
m(1) = 1;

for j = 1:N
    m(j + 1) = dot(m(1:j), conj(flip(b(1:j))));
end
m(1) = 1;

q = zeros(N + 1, 1);
q(1) = 1;
r = zeros(N, 1);

for j = 1:N
    q_truncate =  q(1:find(q,1,'last'));
    q_reverse = flip(q_truncate');
    q_conjugate = conj(q_reverse);
    q_conjugate(numel(q)) = 0;
    q_star = q_conjugate;
    Sq = circshift(q,1);
    r(j) = dot(m, Sq)/dot(m, q_star');
    if j == N
        break
    end
    q = Sq' - r(j)*q_star;
end

% Echoes-to-Impedance Transform
```

```
zeta_approx = zeta_x0*exp(-2*U*atanh(r));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
zeta_approx_min = min(zeta_approx);
zeta_approx_max = max(zeta_approx);
E2IT_frame_min = min(zeta_min, zeta_approx_min) - 1;
E2IT_frame_max = max(zeta_max, zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k');
hold on
myplot(dis, x, zeta_approx, 'b--');
ylim([E2IT_frame_min E2IT_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');
set(findobj(gca, 'Type', 'Line', 'Linestyle', '--'), 'LineWidth', 2);

% Relative Error
E2IT_relative_error = norm(zeta_approx - zeta(x)')/norm(zeta(x)');
```

Gaussian Noise

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
N = length(u_approx);
a = u_approx(:, 1)/delta;

% Create Gaussian Noise
gaussian_noise = randn(length(u_approx(:,1)), 1);
```

```
% Rescale Gaussian Noise for Desired Proportion
p = 0.005;
gaussian_noise_rescaled = p*norm(a)/norm(gaussian_noise)*gaussian_noise;

% Data with Gaussian Noise
a_noise = a + gaussian_noise_rescaled;

% Initialization
for j = 1:N
    x(j) = x0 + j*delta/2;
end

% Preallocation
U = zeros(N, N);
b = zeros(N, 1);

for i = 1:N
    for j = 1:N
        if i < j
            U(i, j) = 0;
        else
            U(i, j) = 1;
        end
    end
end

b = (delta^(k+1)/w)*U^k*a_noise;

%for Loop for Elementwise m
m = zeros(N + 1,1);
m(1) = 1;

for j = 1:N
    m(j + 1) = dot(m(1:j), conj(flip(b(1:j))));
end
m(1) = 1;

q = zeros(N + 1, 1);
q(1) = 1;
```

```
r = zeros(N, 1);

for j = 1:N
    q_truncate =  q(1:find(q,1,'last'));
    q_reverse = flip(q_truncate');
    q_conjugate = conj(q_reverse);
    q_conjugate(numel(q)) = 0;
    q_star = q_conjugate;
    Sq = circshift(q,1);
    r(j) = dot(m, Sq)/dot(m, q_star');
    if j == N
        break
    end
    q = Sq' - r(j)*q_star;
end

% Echoes-to-Impedance Transform
zeta_approx = zeta_x0*exp(-2*U*atanh(r));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
zeta_approx_min = min(zeta_approx);
zeta_approx_max = max(zeta_approx);
E2ITDirac_frame_min = min(zeta_min, zeta_approx_min) - 1;
E2ITDirac_frame_max = max(zeta_max, zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profiles
figure
myplot(dis, x, zeta(x), 'k');
hold on
myplot(dis, x, zeta_approx, 'b--');
ylim([E2ITDirac_frame_min E2ITDirac_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');
set(findobj(gca, 'Type', 'Line', 'Linestyle', '--'), 'LineWidth', 2);

% Relative Error
E2IT_relative_error = norm(zeta_approx - zeta(x)')/norm(zeta(x)');
```

Uniform Noise

```
clc

% Inputs
x0 = 0;
zeta_x0 = zeta(0);
k = 0;
w = 1;
N = length(u_approx);
a = u_approx(:,1)/(delta);

% Create Uniform Noise
uniform_noise = -1 + 2*rand(length(u_approx(:,1)), 1);

% Rescale Uniform Noise for Desired Proportion
p = 0.001;
uniform_noise_rescaled = p*norm(a)/norm(uniform_noise)*uniform_noise;

% Data with Uniform Noise
a_noise = a + uniform_noise_rescaled;

% Initialization
for j = 1:N
    x(j) = x0 + j*delta/2;
end

% Preallocation
U = zeros(N, N);
b = zeros(N, 1);

for i = 1:N
    for j = 1:N
        if i < j
            U(i, j) = 0;
        else
            U(i, j) = 1;
```

```
        end
    end
end

b = (delta^(k+1)/w)*U^k*a_noise;

% for Loop for Elementwise m
m = zeros(N + 1,1);
m(1) = 1;

for j = 1:N
    m(j + 1) = dot(m(1:j), conj(flip(b(1:j))));
end
m(1) = 1;

q = zeros(N + 1, 1);
q(1) = 1;
r = zeros(N, 1);

for j = 1:N
    q_truncate =  q(1:find(q,1,'last'));
    q_reverse = flip(q_truncate');
    q_conjugate = conj(q_reverse);
    q_conjugate(numel(q)) = 0;
    q_star = q_conjugate;
    Sq = circshift(q,1);
    r(j) = dot(m, Sq)/dot(m, q_star');
    if j == N
        break
    end
    q = Sq' - r(j)*q_star;
end

% Echoes-to-Impedance Transform
zeta_approx = zeta_x0*exp(-2*U*atanh(r));

% Frame Setup
zeta_min = min(zeta(x));
zeta_max = max(zeta(x));
zeta_approx_min = min(zeta_approx);
```

```
zeta_approx_max = max(zeta_approx);
E2ITDirac_frame_min = min(zeta_min, zeta_approx_min) - 1;
E2ITDirac_frame_max = max(zeta_max, zeta_approx_max) + 1;

% Figure of Approximated and Actual Impedance Profile
figure
myplot(dis, x, zeta(x), 'k');
hold on
myplot(dis, x, zeta_approx, 'b--');
ylim([E2ITDirac_frame_min E2ITDirac_frame_max]);
xlim([c d]);
xlabel('x');
ylabel('$\zeta$','interpreter','latex');
set(findobj(gca, 'Type', 'Line', 'Linestyle', '--'), 'LineWidth', 2);

% Relative Error
E2IT_relative_error = norm(zeta_approx - zeta(x)')/norm(zeta(x)');
```

E.1 Proof of (3.8) in §3.4.3

The One-Dimensional Wave Equation

We have the following one-dimensional wave equation:

$$u_{tt} - \frac{1}{\zeta}\left(\zeta u_x\right)_x = 0, \qquad \text{or} \tag{E.1a}$$

$$\zeta u_{tt} = \left(\zeta u_x\right)_x, \tag{E.1b}$$

where ζ is a smooth function of x. Based on the theory of partial differential equations, a right-moving Dirac impulse supported at $t = x - a$ determines the following solution to

(E.1a):

$$u(x,t) = f(t)\delta(x-t-a) + H\left[-(x-t-a)\right]s(x,t) \tag{E.2}$$

where

1. $f(t)\delta(x-t-a) =$ a Dirac impulse supported at $t = x-a$ and $f(t)$ is the amplitude of the impulse at a certain t.
2. $H\left[-(x-t-a)\right] =$ the Heaviside function flipped about the y-axis supported at $t = x-a$
3. $s(x,t) =$ a smooth function

The idea is to substitute (E.2) into (E.1b) and then compare the following coefficients on the LHS and RHS of (E.1b):

1. $\delta(x-t-a)$
2. $\delta'(x-t-a)$
3. $\delta''(x-t-a)$
4. $H(a+t-x)$.

As we are taking partial derivatives with respect to t and x, it is important to understand the symmetry and relationship of the Dirac delta function and the Heaviside function.

1. $\delta(-y) = \delta(y)$
2. $\delta'(-y) = -\delta'(y)$
3. $H'(y) = \delta(y)$
4. $H''(y) = \delta'(y)$.

LHS of (E.1b)

The first partial derivative of (E.2) with respect to t is the following:

$$\begin{aligned} u_t = & f'(t)\delta\left(x-t-a\right) - f(t)\delta'(x-t-a) \\ & + H'\left[-(x-t-a)\right]s(x,t) + H\left[-(x-t-a)\right]s_t(x,t) \end{aligned}$$

The second partial derivative of (E.2) with respect to t is the following:

$$\begin{aligned} u_{tt} &= f''(t)\delta(x-t-a) - 2f'(t)\delta'(x-t-a) + f(t)\delta''(x-t-a) \\ &+ H''[-(x-t-a)]s(x,t) + 2H'[-(x-t-a)]s_t(x,t) + H[-(x-t-a)]s_{tt}(x,t) \\ &= f''(t)\delta(x-t-a) - 2f'(t)\delta'(x-t-a) + f(t)\delta''(x-t-a) \\ &+ \delta'[-(x-t-a)]s(x,t) + 2\delta[-(x-t-a)]s_t(x,t) + H[-(x-t-a)]s_{tt}(x,t) \\ &= \left[f''(t) + 2s_t(x,t)\right]\delta(x-t-a) + \left[-2f'(t) - s(x,t)\right]\delta'(x-t-a) \\ &+ f(t)\delta''(x-t-a) + H(a+t-x)s_{tt}(x,t) \end{aligned}$$

Finally, we multiply u_{tt} by $\zeta(x)$.

$$\begin{aligned} \zeta(x)u_{tt} = & \left[\zeta(x)f''(t) + 2\zeta(x)s_t(x,t)\right]\delta(x-t-a) \\ & + \left[-2\zeta(x)f'(t) - \zeta(x)s(x,t)\right]\delta'(x-t-a) + \zeta(x)f(t)\delta''(x-t-a) \\ & + \zeta(x)s_{tt}(x,t)H(a+t-x) \end{aligned}$$

RHS of (E.1b)

The first partial derivative of (E.2) with respect to x is the following:

$$u_x = f(t)\delta'(x-t-a) - H'[-(x-t-a)]s(x,t) + H[-(x-t-a)]s_x(x,t)$$

Before we take the second partial derivative of (E.2) with respect to x, unlike with respect to t, we multiply u_x by $\zeta(x)$.

$$\begin{aligned}\zeta(x)u_x = f(t)\zeta(x)\delta'(x-t-a) - \zeta(x)H'[-(x-t-a)]s(x,t)\\ + \zeta(x)H[-(x-t-a)]s_x(x,t)\end{aligned}$$

Finally, the second partial derivative of (E.2) with respect to x, which now also involves $\zeta(x)$, is the following:

$$\begin{aligned}(\zeta u_x)_x &= f(t)[\zeta'(x)\delta'(x-t-a) + \zeta(x)\delta''(x-t-a)]\\
&- [\zeta'(x)H'[-(x-t-a)]s(x,t) - \zeta(x)H''[-(x-t-a)]s(x,t)\\
&+ \zeta(x)H'[-(x-t-a)]s_x(x,t)] + \zeta'(x)H[-(x-t-a)]s_x(x,t)\\
&- \zeta(x)H'[-(x-t-a)]s_x(x,t) + \zeta(x)H[-(x-t-a)]s_{xx}(x,t)\\
&= f(t)\zeta'(x)\delta'(x-t-a) + f(t)\zeta(x)\delta''(x-t-a)\\
&- [\zeta'(x)\delta(x-t-a)s(x,t) + \zeta(x)\delta'(x-t-a)s(x,t) + \zeta(x)\delta(x-t-a)s_x(x,t)]\\
&+ \zeta'(x)H(a+t-x)s_x(x,t) - \zeta(x)\delta(x-t-a)s_x(x,t) + \zeta(x)H(a+t-x)s_{xx}(x,t)\\
&= f(t)\zeta'(x)\delta'(x-t-a) + f(t)\zeta(x)\delta''(x-t-a)\\
&- \zeta'(x)s(x,t)\delta(x-t-a) - \zeta(x)s(x,t)\delta'(x-t-a) - \zeta(x)s_x(x,t)\delta(x-t-a)\\
&+ \zeta'(x)s_x(x,t)H(a+t-x) - \zeta(x)s_x(x,t)\delta(x-t-a) + \zeta(x)H(a+t-x)s_{xx}(x,t)\\
&= [-\zeta'(x)s(x,t) - 2\zeta(x)s_x(x,t)]\delta(x-t-a)\\
&+ [f(t)\zeta'(x) - \zeta(x)s(x,t)]\delta'(x-t-a) + f(t)\zeta(x)\delta''(x-t-a)\\
&+ [\zeta'(x)s_x(x,t) + \zeta(x)s_{xx}(x,t)]H(a+t-x)\end{aligned}$$

The Four Equations

We have the following four equations from the four coefficients:

$$\zeta(x)f''(t) + 2\zeta(x)s_t(x,t) = -\zeta'(x)s(x,t) - 2\zeta(x)s_x(x,t) \tag{E.3a}$$

$$-2\zeta(x)f'(t) - \zeta(x)s(x,t) = f(t)\zeta'(x) - \zeta(x)s(x,t) \tag{E.3b}$$

$$\zeta(x)f(t) = f(t)\zeta(x) \tag{E.3c}$$

$$\zeta'(x)s_x(x,t) + \zeta(x)s_{xx}(x,t) = \zeta(x)s_{tt}(x,t) \tag{E.3d}$$

From (E.3b), because $\delta'(x-t-a) = 0$ if $x \neq t+a$, (E.3b) only exists if and only if $x - t - a = 0$. Thus, we have the following:

$$\begin{aligned}
-2f'(t)\zeta(x) - \zeta(x)s(x,t) &= f(t)\zeta'(x) - \zeta(x)s(x,t) \\
-2f'(t)\zeta(x) &= f(t)\zeta'(x) \\
-2\frac{f'(t)}{f(t)} &= \frac{\zeta'(x)}{\zeta(x)} \\
-2\frac{f'(t)}{f(t)} &= \frac{\zeta'(a+t)}{\zeta(a+t)}
\end{aligned}$$

By a change of variable of t to s and integrating both sides with respect to s, which is within the interval $[0, t]$, we can find $f(t)$.

$$\begin{aligned}
-2\int_0^t \frac{f'(s)}{f(s)}\,ds &= \int_0^t \frac{\zeta'(a+s)}{\zeta(a+s)}\,ds \\
-2\ln f(s)\Big|_0^t &= \ln \zeta(a+s)\Big|_0^t \\
-2\ln\frac{f(t)}{f(0)} &= \ln\frac{\zeta(a+t)}{\zeta(a)} \\
\ln\frac{f(t)}{f(0)} &= \ln\sqrt{\frac{\zeta(a)}{\zeta(a+t)}} \\
f(t) &= f(0)\sqrt{\frac{\zeta(a)}{\zeta(a+t)}}
\end{aligned}$$

Since the initial amplitude of the Dirac impulse for our numerical experiments is $f(0) = 1$, the formula simplifies to (3.8) where $f(t) = u_{\text{sing}}$.

F.1 MATLAB's `myplot` Function

```
function myplot(discontinuities, varargin)

%Same as plot(), except that first argument is a list of
%discontinuities. No line joining across these discontinuities will
%appear in the plot


if nargin < 3 || ~isnumeric(varargin{2})
   varargin = [{1:length(varargin{1})}, varargin];
end

d = [-inf;discontinuities(:);inf];
intervals = [d(1:end-1), d(2:end)];
N = length(intervals);
idx = find(diff(cellfun(@isnumeric, varargin)) == 0);
```

```
M = length(idx);

for i = 1:N
  args = varargin;
  for j = idx
      xdata = args{j};
       cut = xdata <= intervals(i,1) | xdata >= intervals(i,2);
      args{j}(cut) = nan;
      args{j+1}(cut) = nan;
  end
  plot(args{:}); hold on
end

hold off
```

G.1 Running Time of the Code in Appendix D.1.2

It is possible to estimate the computation time to run the echoes-to-impedance transform for a Dirac source wavelet propagating through a smooth impedance profile. From Table 5.13b, it seems to increase exponentially as n[1] doubles. By using a log-log scatter plot and finding a equation for the line of best fit, the equation can calculate an approximate computational time to run the code. The equation is useful for devices that can only run up to a maximum number of n points, which is $n = 16000$ for the device used for this thesis. Fig. G.1 displays the log-log scatter plot along with its line of best fit and equation, which is (G.4).

$$\ln T = 2.1987 \ln n - 17.3108 \tag{G.4}$$

[1] Note that n is the number of spatial grid points, but the inverse scattering algorithm works with the number of temporal grid points, t, which is twice as many as n.

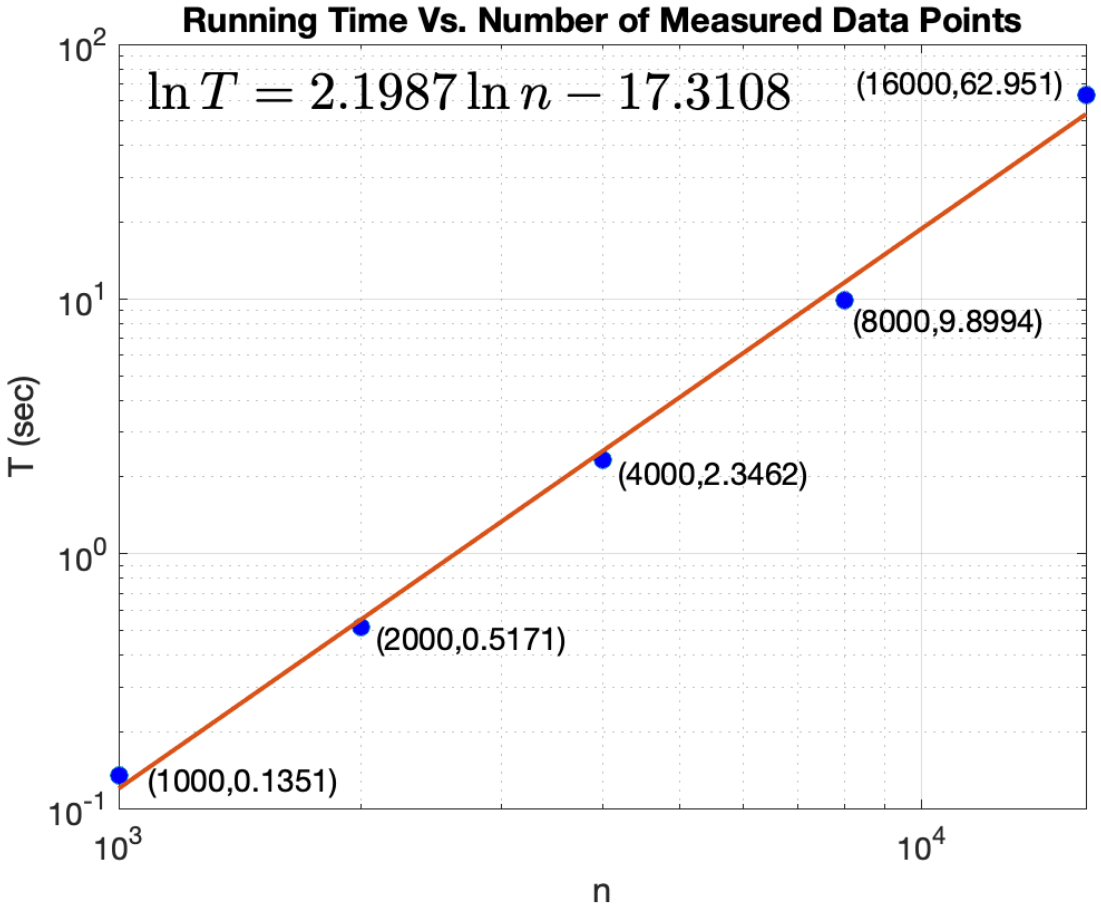


FIGURE G.1

For example, (G.5) calculates the computational time for the echoes-to-impedance transform if $n = 32000$.

$$\begin{aligned} \ln T &= 2.1987 \ln 32000 - 17.3108 \\ &= 5.4974 \\ T &= 244.0554\text{s} \end{aligned} \tag{G.5}$$

Index