Accurate and Automatic Refraction Statics in Large 3D Seismic Datasets

by

Atul Jhajhria

A Thesis Submitted to the College of
Graduate Studies and Research
in Partial Fulfillment of the
Requirements for the degree of
MASTER OF SCIENCE
in
Geophysics

Approved:

____
Dr. Igor Morozov, Thesis Advisor

____
Dr. Samuel Butler

____
Dr. Jim Merriam

____
Dr. Brian Russell, External Examiner

____
Dr. Kevin Ansdell, Chair

University of Saskatchewan
March 2009

To my parents

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First of all, I would like to thanks my supervisor, Dr. Igor Morozov for his constant guidance and financial support. Often I went to his office unannounced and he was always kind enough to answer my questions and help me with any difficulties in my research work. Even during weekends, he replied to my emails. I am thankful to him for going through my thesis at least five times and making corrections.

I am thankful to Drs. Samuel Butler, Jim Merriam, Kevin Ansdell, and Brian Russell for serving on my committee. I am thankful to Ricki Elder for proof-reading my thesis and correcting grammar. My sincere thanks to Brian Reilkoff and Jennifer Hadley for computer-related help. I am thankful to Dr. Bhaskar Pandit, who provided much help in using ProMax. I thank all colleagues with whom I had many useful conversations. I am thankful to all my friends who made my stay in Saskatoon enjoyable.

In this work, I used Matlab, GMT programs (Wessel and Smith, 1995), IGeoS, Hampson-Russell, and Landmark Graphics software. Financial support for this study from Saskatchewan Energy and Resources is also gratefully acknowledged.

SYMBOLS AND ABBREVIATIONS

Symbol	**Definition**
1D	One dimensional
2D	Two dimensional
3D	Three dimensional
GLI	Generalized linear inverse
QC	Quality control
IGeoS	Integrated GeoScience (software package)
p	Slowness/Ray-parameter
t-x	Time versus offset
τ-p	Intercept time versus slowness
$\mathbf{d}$	Data vector (travel-times)
$\mathbf{L}$	Kernel matrix in the forward travel-time equation
$\mathbf{m}$	Model vector (depth values at each #node)
m	Metre
km	Kilometre
$\mathbf{R}_\mathrm{m}$	Model resolution matrix
SIRT	Simultaneous Iterative Reconstruction Technique

ABSTRACT

Inversion for refraction statics is a key part of three-dimensional (3D) reflection seismic processing. The present thesis has two primary goals directed toward improvement of refraction statics inversion. First, I attempt to improve the quality of the travel-time data right at the beginning of the processing sequence and before any inversion. Any error in the travel times or geometry caused during acquisition or processing would propagate into the resulting model and may harm the resulting image. To implement rigorous, model-independent data quality control, I view the first-arrival travel times as surfaces in 3D, which allows utilization of the travel-time reciprocity condition to check for errors in geometry and in first-arrival picking.

The second goal of this study is in development of a new inversion approach for refraction statics specifically for 3D seismic datasets. The first-break travel-times are decomposed by using a τ-p parameterization, which allows an automatic derivation of a high-quality initial subsurface model. This model is further improved by using accurate, multi-layer ray-tracing and inversion techniques to obtain accurate refraction statics. An iterative inversion scheme based on the Simultaneous Iterative Reconstruction Technique is utilized, and its performance is measured and discussed. To assess the quality of the inverse and establish the optimal grid sizes, I use several types of resolution tests. Finally, the surface consistent statics is calculated and applied to a real dataset from southern Saskatchewan. A comparison of the resulting statics model with statics calculated by using standard industry software is made, and the statics correction is incorporated in seismic processing.

An overall result of this study is in demonstration that the fully 3D, τ-p based travel-time inversion method works, is applicable to large seismic datasets, and results in detailed shallow subsurface models and reliable statics solutions. Several recommendations for extending and improving the proposed approaches are also made.

1. Introduction

1.1 The Refraction Statics problem

The ultimate goal in reflection seismic data processing is to obtain an accurate image of the subsurface, which is critical for interpretation during exploration for hydrocarbons and other geological targets. The typical target of seismic interpretation is identification of features which could reveal the oil and gas prospects of the region of interest. The common ways to find potential reservoirs is to look for structural and stratigraphic traps with the help of sophisticated imaging and interpretation software. The images are obtained by using sequences of processing steps, and therefore the interpretation can only be reliable when all these steps are correct and sufficiently accurate.

One of the key steps of seismic data processing is the statics correction. The term statics denotes the highly variable travel times of reflected waves (blue ray in Figure 1.1) accumulated during their propagation within the shallow subsurface (Telford et al., 1990). The near-surface layer (weathered zone) is loosely consolidated and significantly more non-uniform compared to the deeper layers. The uneven thickness of the near-surface layers and low velocities lead to large (often up to ~50 ms or more), strongly variable time shifts of the reflected waves recorded from the deeper layers (Figure 1.1). Because reflected rays propagate nearly vertically within the low-velocity weathered zone, such time shifts are practically independent of the depth of reflections, and they are consequently called statics.

If not mitigated, static shifts are capable of completely disrupting the coherence of reflections during common midpoint stacking. Spurious reflection patterns and loss of depth resolution can also arise from incorrect or inaccurate statics. Such images could lead to erroneous interpretations which could be costly in terms of money and time. The process for compensating statics is referred to as static correction; this is one of the most critical and time-consuming steps in reflection data processing, and it is the central subject of this thesis.

Within the general irregular time shifts related to the weathered zone, several types of statics are differentiated (Telford et al., 1990). The statics due to the differences in surface elevations which affect both sources and receivers are called elevation statics. These statics can be corrected relatively easily if one knows the elevations and the near-surface seismic velocities. Sources typically have additional negative statics due to their being buried at variable depth below the surface; such statics can be compensated by using the "uphole" times measured by the wave propagation from the sources to the nearest receivers. Additional static shifts are also associated with velocity variations within the weathered zone itself, such as caused by layering or variations of its depth. By their relation to the source or receiver position, statics are also subdivided to source and receiver statics, and the "total" static of a seismic trace is the sum of all three statics at the corresponding source and at receiver locations. Finally, statics are called "surface-consistent" if they are only related to the surface locations of the source and receivers and not to their individual properties.

All of the statics above can be incorporated in the concept of "refraction statics" (Yilmaz, 2001). Refraction statics represent a group of methods based on constructing a realistic model of the shallow subsurface by inverting the refracted (first-break) arrivals (red ray in Figure 1.1). This model should incorporate the complete topography, depths of buried sources, as well as the variations in the structure of the weathered zone. This is the most complete and advanced approach to developing statics solutions, and it is used in the present thesis.

Refraction statics calculations are based on the use of refracted head waves to model the first-arrival travel times. Several refraction-statics methods are in broad use, such as the Plus-Minus method, Generalized Reciprocal method, and the Generalized Linear Inverse. These methods take the first-arrival times as input and use different kinds of travel-time modeling to derive estimates of the depths and/or subsurface velocities. Most of these travel time models are based on the following dependence of the head-wave travel time on the source-receiver distance *x* (Figure 1.1) in a horizontal one-layer case:

$$t(x) = \frac{2h_1}{v_1}\cos\theta_1 + px\,. \tag{1.1}$$

Here, h_1 is the thickness of the layer (Figure 1.1) v_1 – its velocity, v_2 is the velocity of bottom layer, and p ($\sin\theta_1/v_1 = 1/v_2$) is the ray parameter. This equation relates the observed property (time) to the physical properties (depth and velocity) of the layers beneath the source receiver locations. By analysing the dependence of t on x, model parameters v_1, and h_1 in this equation can be estimated. In practice, spatially-variable layer velocities and thicknesses are used, and multiple layers may be needed for accurate modeling of the subsurface structure (Figure 1.1). These differences in the models determine the differences between the various methods.

In order to derive statics from a layered model, consider a nearly-vertically propagating ray shown in Figure 1.2. As I show below, for modeling and inversion, it is convenient to use models with multiple constant-velocity layers. For a single such layer, if the datum is located within the "base" layer beneath it (Figure 1.2), the total source static is:

$$t_s = \frac{E_S - D_S - E_{SLayer1}}{V_{Layer-1}} + \frac{E_{SLayer1} - E_D}{V_{\mathrm{Repl}}}. \qquad (1.2)$$

where E_S is the elevation at the surface directly above the source location,

D_S is the source depth, $E_{Slayer1}$ is the elevation at the base of layer directly below the source location, E_D is the elevation of the datum, and V_{Repl} is the replacement velocity.

Subtraction of this static value from travel times would effectively move the source (point S) to the datum (point $S^{'}$; Figure 1.2). The static at the receiver location can be calculated in the same way (without the D_S term), and the total trace static would be the sum of the source and receiver statics. This decomposition of the total refraction statics can be naturally extended to a multi-layer case.

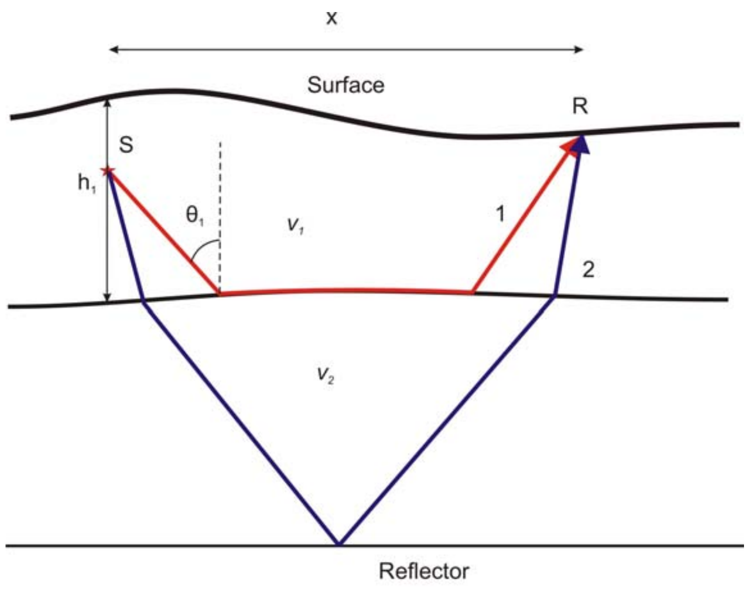


Figure 1.1: Schematic cross-section of a 2D reflection survey subsurface. The source is at position S and the receiver is located at R. Ray labelled "1" represents a head wave and ray "2" is a reflected wave.

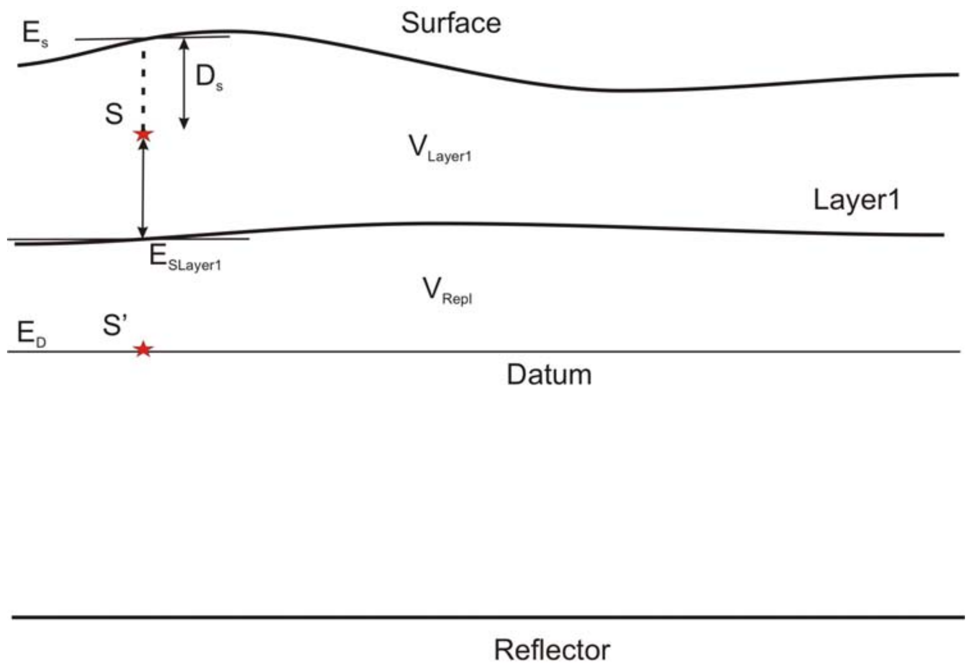


Figure 1.2: Schematic diagram for calculating source statics for a single-layer weathered zone. E_S, E_D, $E_{SLayer1}$ are the elevations at respective positions. V_{Layer1} is the velocity of layer 1.

The first step in calculating refraction statics is to pre-process the data for picking the first arrivals. This procedure includes loading the field geometry parameters, extensive quality control, and removal of auxiliary channels and bad traces.

Once geometry is loaded, the seismic data are sorted with source numbers as the primary keys, and line numbers and offsets as the secondary keys to organize for efficient travel-time picking. The next step is to pick the first breaks in these sorted shot gathers. Because of their large amplitudes, the first breaks can typically be easily recognized. However, noisy data may be more difficult or ambiguous to pick. Generally, the seismic processor selects the amplitude peaks, troughs, or zero crossings for travel-time picking, and tries maintaining its consistency throughout the entire dataset. In order to keep picking consistent, switching to other sort orders (e.g., by common receivers or midpoints, CMP) can be useful. In addition, reciprocal time analysis as described further in this thesis is also useful to achieve a consistent first-arrival time picking.

After an overview of the existing approaches, the following chapters explain in detail the underlying concept of refraction travel-time modeling, inversion, and the results using synthetic and real datasets. The Matlab code that I have written and used in the inversion is summarized in Appendix A and is also available at http://seisweb.usask.ca/students/atul.

1.2 Existing Approaches

Most of the refraction-statics methods, such as the Plus-Minus and the Generalized Reciprocal methods are based on the delay-time approximation of refracted travel times (Yilmaz, 2001) to solve for the statics. Consider a source located at point S and a receiver at point R at the surface (Figure 1.3). In the delay-time approximation, the refractor is considered as near-horizontal between the two points, and the distance between them is much greater than the critical distance. Generally, this implies that the velocity of the refractor (bedrock) is much larger than that of the overburden.

Under these approximations, the travel-time from S to R can then be separated to the source-side and receiver-side times:

$$t_{SR} = t_{SX} + t_{XR} . \tag{1.3}$$

Time t_{SX} can be represented as a sum of the travel time along the reflector and the "source delay" time:

$$t_{SX} = t_{SA} - t_{BA} + t_{BX} = t_{SDelay} + \frac{x}{v_2}. \tag{1.4}$$

For source delay, t_{SDelay}, we therefore have:

$$t_{SDelay} = \frac{SA}{v_1} - \frac{BA}{v_2} = \frac{h_S}{v_1 \cos i_c} - \frac{h_S \tan i_c}{v_2} = \frac{h_S \cos i_c}{v_1}. \tag{1.5}$$

In a similar way, the receiver delay time is defined, and the total time from the source to the receiver is:

$$t_{SR} = t_{SDelay} + t_{RDelay} + \frac{SR}{v_2}. \tag{1.6}$$

This equation relates the velocity of the bedrock and the depth of the weathering layer to the first-arrival travel times. This equation is further inverted to solve for the depths of the weathering layer near the sources and receivers, and the velocity of the refractor. Several inversion methods are commonly used, of which I briefly discuss the Plus-Minus method, the Generalized Reciprocal Method (GRM), and the Least Squares method, also known as the Generalized Linear Inverse (GLI) method.

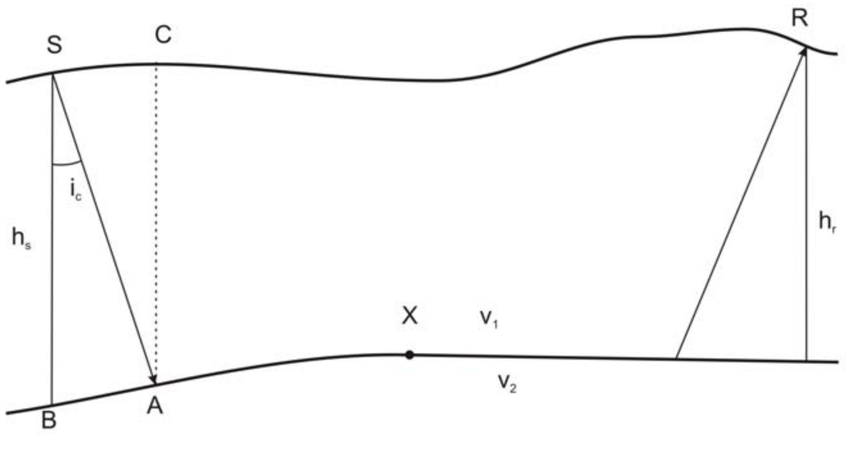


Figure 1.3: Head wave traveling from source S to receiver R or vice versa. Depth below source is denoted h_S and depth below receiver is h_R. Point X separates the head-wave path into source- and receiver-related parts.

1.2.1 Plus-Minus method

The Plus-Minus method is based on manipulating the observed reversed head wave times at locations between the sources and receivers, as shown in Figures 1.4 and 1.5. This method was given by Hagedoorn (1959) and uses the delay-time concept. For this method, the velocity of the weathering layer is assumed to be measured from near-shot travel times and is denoted as v_1.

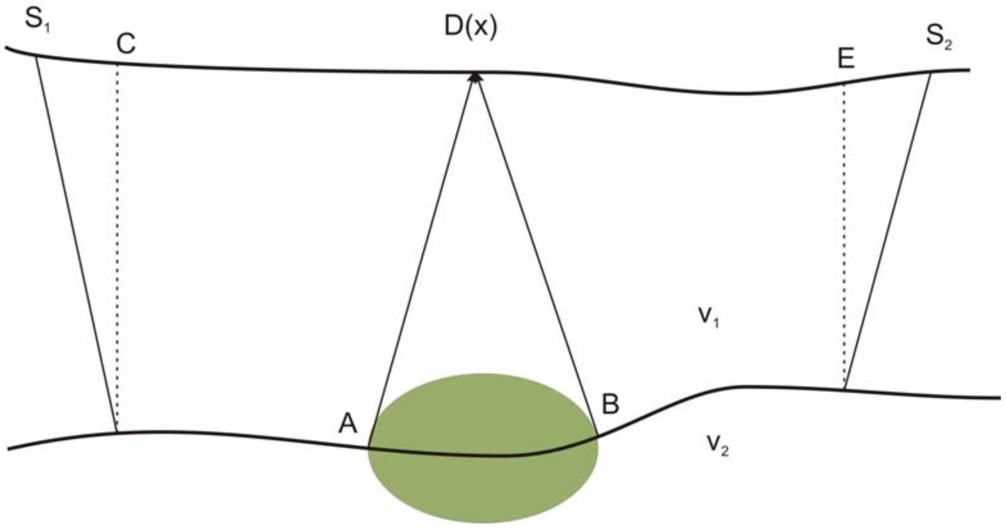


Figure 1.4: Schematic diagram of the plus-minus method. Travel times from sources S_1 and S_2 are measured at various locations D(x). The region between points A and B is the pre-critical region. Its extent (green ellipse) restricts the horizontal resolution of the method.

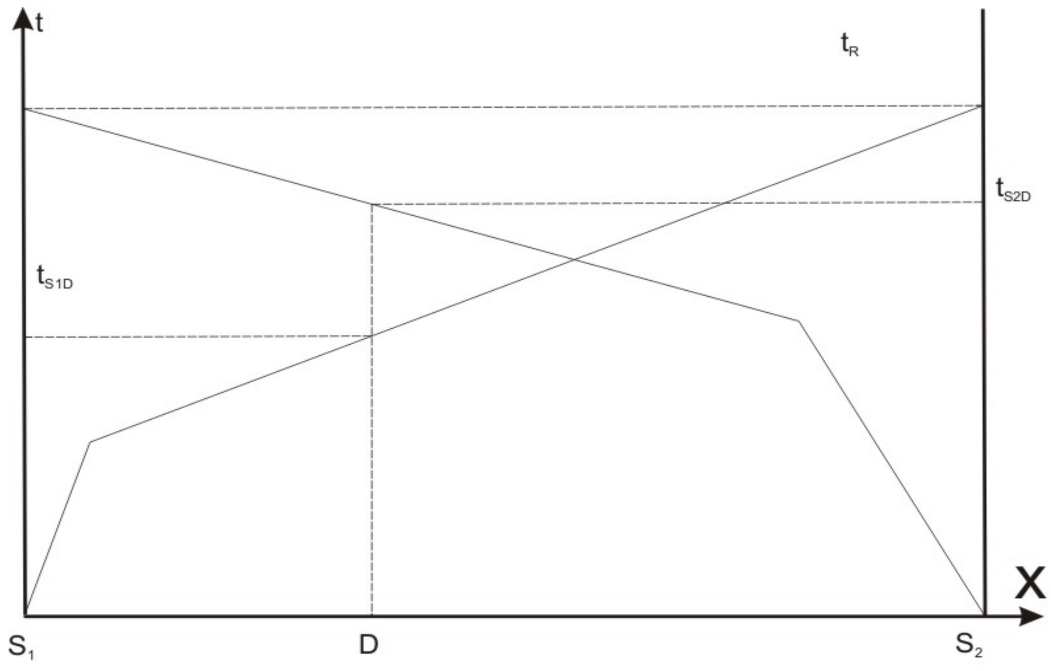


Figure 1.5: Travel times of head waves traveling in opposite directions as functions of the receiver location *x*. The reciprocal time t_R is the time the wave takes to reach from $S_1$to S_2 or vice versa.

For waves traveling from point S_1 to D, the delay-time formula (1.6) gives:

$$t_{S_1D} = t_{S_1} + t_D + \frac{x}{v_2}, \tag{1.7}$$

and for waves traveling in the opposite direction:

$$t_{S_2D} = t_{S_2} + t_D + \frac{S_1S_2}{v_2}. \tag{1.8}$$

By constructing the “plus” time, we see that it is represents a constant plus twice the delay-time at point *D*:

$$t_{PLUS} = t_{S_1D} + t_{S_2D} = \frac{S_1S_2}{v_2} + t_{S_1} + t_{S_2} + 2t_D = t_{S_1S_2} + 2t_D. \tag{1.9}$$

The time t_{S1S2} is the reciprocal time between the shot points S_1 and S_2, which can be readily measured from either of the two shot records. Hence:

$$t_D = \frac{1}{2}\left(t_{PLUS} - t_{S_1S_2}\right). \tag{1.10}$$

To transform t_D into the depth beneath the point D, we need to derive the critical angle i_c, for which we need to find v_2. This velocity can be obtained from the "Minus" travel-time:

$$t_{MINUS} = t_{S_1D} - t_{S_2D} = \frac{2x}{v_2} - \frac{S_1S_2}{v_2} + t_{S_1} - t_{S_2}, \tag{1.11}$$

and therefore:

$$Slope\left[t_{MINUS}(x)\right] = \frac{2}{v_2}. \tag{1.12}$$

This slope on the travel-time plot can be estimated visually or by mathematical methods, such as the Least Squares regression. We can choose the location of point D, calculate the Plus times, and from them determine the depths of weathering layer below any points between C and E in Figure 1.4. The Minus time is used to solve for the velocity of the bed rock. Finally, by varying the location of D, we can derive a depth profile and use it to calculate the statics at each station.

1.2.2 Generalized Reciprocal method

One limitation of the Plus-Minus method is in its averaging over the pre-critical region near point D (green ellipse in Figure 1.4). Also, the location of station D (Figure 1.5) might not always correspond to a receiver station. Both of these limitations are overcome by using the Generalized Reciprocal method (GRM; Palmer, 1981), which is an extension of the Plus-Minus method. In this method, the locations of the two points D_1 and D_2 on the surface are separated by a fixed distance D_1D_2, so that tighter "focusing" on the refractor can be achieved (Figure 1.6).

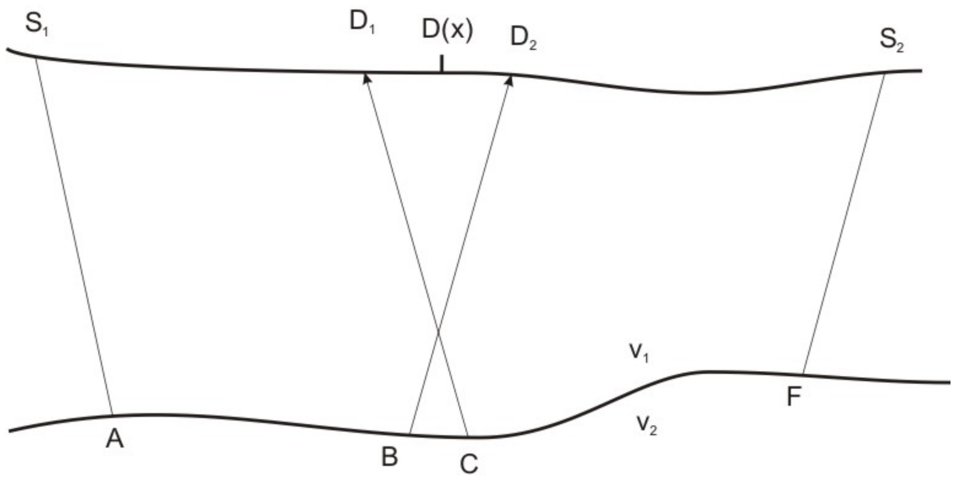


Figure 1.6: First arrivals at locations D_1 and D_2 are used to estimate the depth below the point D. Several combinations of D_1 and D_2 are used at each location.

Because segment D_1D_2, is covered by the head waves twice, the formulae for the Plus and Minus times are modified in the GRM method (Yilmaz, 2001):

$$t_{PLUS} = t_{S_1ABD_2} + t_{D_1CFS_2} - t_{S_1AFS_2} - \frac{D_1D_2}{v_2}, \tag{1.13}$$

and:

$$t_{MINUS} = t_{S_1ABD_2} - t_{D_1CFS_2} + t_{S_1AFS_2}. \tag{1.14}$$

The depths of the refractor and the velocity of the bedrock are estimated from equations (1.13) and (1.14). In this method, multiple estimates of the depth are made below each point D by using different separations between D_1 and D_2. The value of the D_1D_2 distance resulting in the most linear $t_{\text{MINUS}}(x)$ and the most detail in $t_{\text{PLUS}}(x)$ profile is considered to be the optimal. This selection of the optimal distance represents a subjective, difficult-to-quantify choice in this method. Like the Plus-Minus method, the GRM method can only be used with simple, single-layer models.

1.2.3 Generalized Linear Inverse method

The Generalized Linear Inverse (GLI) refraction statics method addresses both of the limitations above. The GLI approach is a broad group of multi-layer model-based techniques using accurate ray tracing and linear algebra methods to solve for detailed subsurface models. In particular, the method was implemented in the Hampson-Russell software package GLI3D, which is broadly used in both the industry and academia (Hampson and Russell, 1984).

The GLI method uses an iterative inversion approach. In this method, a starting model is chosen and ray tracing is done to estimate the first arrivals. The model is then iteratively updated until a match between the estimated and observed arrival times is achieved. In more detail, the existing GLI approach is discussed in Chapter 4.

The approach to refraction statics developed in this thesis also belongs to this GLI group and incorporates complex multi-layer models, accurate ray tracing, and an iterative inverse, However, it also reaches far beyond the traditional refraction statics inversion (like H-R GLI3D) and attempts contributing to a complete environment for first-arrival travel-time analysis and modeling, which incorporates a built-in 3D geometry, travel-time quality control, and also integrates waveform processing and travel-time picking tools.

1.3 Motivation

The general motivation for this research is to improve the existing approaches to refraction statics in three ways:

1. I use extensive pre-inversion data analysis and quality control (QC), which is not commonly performed in standard refraction inversion programs but could make great improvement in the quality of the inversion.
2. I employ model parameterization and inversion techniques that are different from the commonly used. These techniques allow great savings of processing times by allowing automatic construction of starting models and efficient iterations. In particular, the original procedure for constructing the starting model for inversion by using the Herglotz-Wiechert transform is likely to greatly improve the convenience, speed, and accuracy of the solution.

3. Finally, standard inversion QC operations (such as the resolution matrix and checkerboard resolution tests) are not commonly performed in the existing software. However, such tests yield quantitative measures of the quality and resolution of the model, and they allow selection and analysis of the model and algorithm parameters. Extensive QC tests are performed in the inversion of this study.

This research contributes to an integrated refraction statics analysis environment which is currently being developed in our group. Ultimately, this environment should provide significantly improved data QC and 3D visualization capabilities. It should also include provisions for automatic and manual picking of first arrivals in a surface-consistent way, with accounting for reciprocal travel-time relationships, as discussed in Chapter 2. However, this is a very broad goal, and the tasks of the present study are focused on the first-arrival travel-time modeling and inversion. In addition, I emphasize the analysis of algorithms and the resulting models. Therefore, the specific goals of this project include:

1) Use the ray-parameter based model parameterization that allow automatic construction of starting models, study and use such models;
2) Make an open-source inversion code that is easy to analyze and adapt to various related problems;
3) Test several ray-tracing and inversion approaches (especially the Simultaneous Iterative Reconstruction Technique, SIRT);
4) Analyze the inverted model resolution;
5) Compare the resulting model and statics to the solution by Hampson-Russell software (GLI3D);
6) Apply the resulting statics model to a subset of a large Beaver Ranch 3D seismic dataset in southern Saskatchewan.

For software development, Matlab programming environment was used, which allowed easy implementation and testing of complex matrix algorithms. The approach is thus intended as a prototype of the future large-scale code, which is intended for implementation in IGeoS system (http://seisweb.usask.ca/igeos). By using this system, the approaches described here, can be easily incorporated in full-scale, high-performance seismic data processing and inversion.

1.4 3D seismic dataset

The Beaver-Ranch 3D seismic dataset used for the study was donated to this project by Olympic Seismic. The dataset covers ~ 400 km^2 area in southern Saskatchewan (Figure 1.7). For this study, only the part of this dataset was used, including 255 shots and 12 lines of receivers at the western edge of the survey (red box in Figure 1.7). The detailed source-receiver layout used in this work is shown in Figure 1.8. The smaller dataset allowed us to increase the number of tests performed without having to wait for extended period of time while using relatively slow Matlab simulations. However, the selected subset is still large enough to give us meaningful information about the 3D structure of the study area and to produce a sample stack.

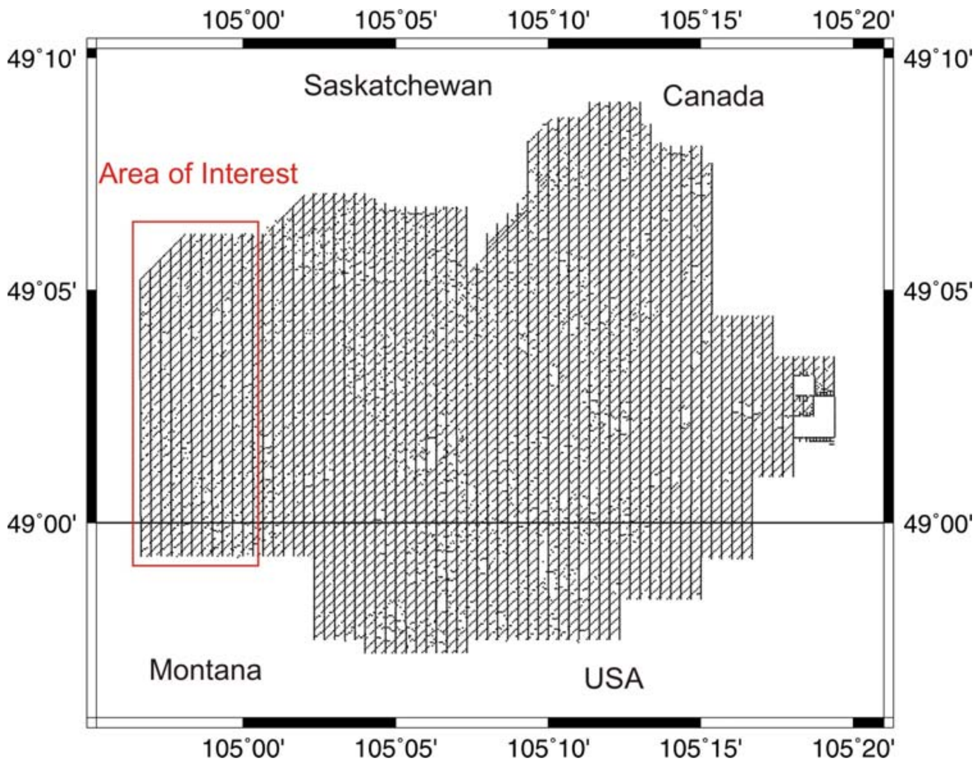


Figure 1.7: Location map of Beaver Ranch 3D Seismic dataset. The receiver lines extend north-south, and the source lines are oriented in the NE-SW direction. Red box indicates the data subset chosen for this study.

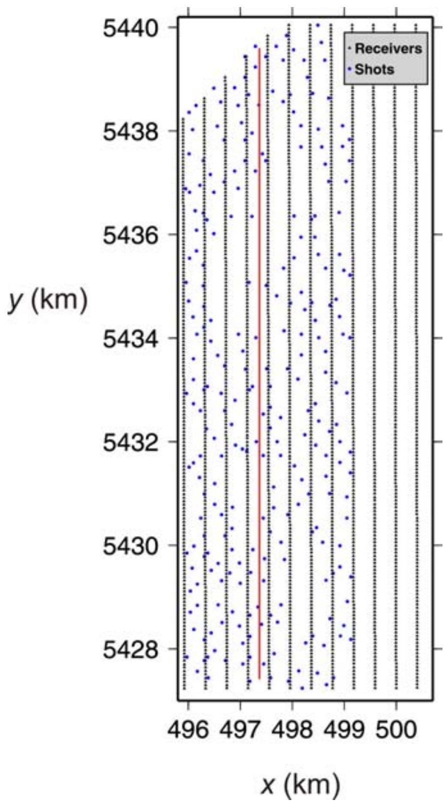


Figure 1.8: Locations of 255 shots (blue dots) and 12 receiver lines (densely spaced black dots) used in this thesis. Red line is the location of the resulting stacked section shown in Chapter 5. The total number of first-arrival travel-time picks is 169,667.

1.5 Structure of this Thesis

In this thesis, Chapter 2 discusses the relation of this study to the ongoing research on 3D refraction statics and seismic processing at Dr. Morozov's seismic group. Chapter 3 discusses the ray tracing methods, and Chapter 4 describes the inversion techniques and my prototype implementation using Matlab.

Further, Chapter 5 examines the resolution of the inversion methods by using a part of the Beaver Ranch seismic dataset and provides real field data examples. Chapter 6 concludes the research results and offers suggestions for further development and application of the new method.

2. First-Arrival Analysis Environment

The key idea of our refraction statics approach is in the introduction of extensive analysis of the travel times prior to their entering the model-based inversion. Inversion can only succeed when the input travel times are correct and consistent with the mathematical model of the first-arrival seismic wave propagation. Any travel-time errors caused, for example, by cycle skipping or errors in phase identification during picking would be impossible to identify during the inversion and will adversely affect the solution. Errors in source-receiver geometry could also have severe impact on the quality of the refraction solution. Therefore, such errors need to be identified and removed prior to the inversion.

The travel-time analysis procedure is based on the concept of the travel-time field, in which the first-arrival time readings are viewed as representing a continuous travel-time field function $t(x_S, x_R)$ sampled in a four-dimensional space formed by the positions of sources and receivers. For various purposes, common-shot, common-receiver, common-midpoint, or common offset-azimuth slices of this space can be created, in each of which the resultant travel-time field represents continuous two-dimensional surfaces. Such continuity is a very powerful criterion allowing establishing the internal consistency of the travel times.

Fortunately, the travel time field possesses several important properties that can be used to verify and establish the self-consistency of the travel-time field regardless of the subsurface model. These properties are: 1) travel-time reciprocity when using seismic sources and receivers located on a common surface, 2) similarity of the travel-time fields recorded from adjacent sources or receivers (i.e., travel-time field continuity); 3) great redundancy of travel-time sampling in 3D recording, and 4) generally regular variation of the refraction travel times with the source-receiver distance and azimuth. This allows inverting the travel times for an empirical “travel-time model” before defining an inverse problem that solves for a subsurface velocity structure. In defining such a model, only general properties of the 3D source-receiver coverage and the general character of the first-arrival travel-time inversion problem are utilized, as described below.

In the following, I summarize the main components of the new environment for integrated first-break travel-time analysis and refraction statics inversion developed in our group. Much of the data quality-control and visualization work was performed by my supervisor (Dr. I. Morozov), and I will only focus on the inversion components. However, the underlying travel-time field parameterization and the resulting corrections are also critical for this Thesis, and they are briefly presented here. These corrections utilize innovative travel-time decomposition for reliable and consistent manual and automatic travel-time picking, checking geometry, and constructing starting models for inversion.

Two- and three-dimensional interactive visualization is used at all stages of data analysis and inversion. The implementation is based on a large geophysical data processing system and allows broad customization of the refraction statics analysis and incorporation of other data.

2.1 Model-independent travel-time field decomposition

The travel-time model is constructed by explicitly separating the contributions from the receiver-, source, and offset/azimuth- related factors. For a source located beneath point $\mathbf{x}_\mathrm{S}$ at the free surface in 3D space, and a receiver at point $\mathbf{x}_\mathrm{R}$ at this surface, the observed travel time $t_{SR}(\mathbf{x}_R)$ can be expressed in the following way:

$$t_{SR}(\mathbf{x}_R) = t(\mathbf{x}_S, \mathbf{x}_R) - (t_u + t_S), \tag{2.1}$$

where the surface-consistent $t(\mathbf{x}_S, \mathbf{x}_R)$ travel-time is:

$$t(\mathbf{x}_S, \mathbf{x}_R) = t_d(\mathbf{d}_{SR}) + t_R(\mathbf{x}_R) + \delta t_{SR}(\mathbf{x}_R). \tag{2.2}$$

In these expressions, $\mathbf{d}_{SR} = \mathbf{x}_R - \mathbf{x}_S$ is the source-receiver offset vector, $t_R(\mathbf{x})$ is the receiver delay common to all sources, $t_u + t_S$ is the shot uphole time (shot time advance common to all receivers), and $\delta t_R(\mathbf{x})$ is the remaining travel-time delay of the particular travel-time reading relative to a combination of these terms. In eq. (2.1), an additional time shift t_S is added to the measured shot uphole time t_u to allow compensation for any errors in t_u or for inclusion of additional shot-time corrections.

The different terms in the time-field decomposition (2.2) possess several properties making them useful in data quality control, travel-time inversion, and also in manual and automatic picking. The distance-dependent term $t_d(\mathbf{d}_{SR})$ is a relatively smooth function which is therefore close for the adjacent shots or receivers. This whole dependence can therefore be interpolated between the nearby shots to predict the travel-times in hitherto unpicked shots. The term $t_R(\mathbf{x}_R)$ is variable and comprises much of the elevation and "short-wavelength" receiver statics also common to all shots. By contrast, term $\delta t_{SR}(\mathbf{x})$ is highly variable, but it is also relatively limited in magnitude and represents a continuous 2D surface when viewed as function of $\mathbf{x}_R$. Performing a Delaunay triangulation of this surface allows interpolation of this term and predicting it at any receiver locations. Finally, the $t_u + t_S$ term is common to the entire shot, and adjusting the t_S parameter can be used to improve the average travel-time reciprocity, as explained below.

2.2 Using the travel-time reciprocity for correcting shot uphole times and QC

Regardless of the subsurface structure, the surface-consistent refracted travel times (2.2) between any two points must satisfy the reciprocity condition:

$$t(\mathbf{x}_S, \mathbf{x}_R) = t(\mathbf{x}_R, \mathbf{x}_S). \tag{2.3}$$

This condition can and should be tested and corrected prior to inversion. In our approach, we calculate the reciprocal time misfits between all pairs of shot locations *Si* and *Sj* with reciprocal (reversed) recording:

$$\delta t_{Si,Sj} = t(\mathbf{x}_{Si}, \mathbf{x}_{Sj}) - t(\mathbf{x}_{Sj}, \mathbf{x}_{Si}) = t_{ui} - t_{uj} + t_{Si} - t_{Sj}, \tag{2.4}$$

and therefore:

$$t_{Si} - t_{Sj} = \delta t_{Si,Sj} - t_{ui} + t_{uj}. \tag{2.5}$$

The travel times $\delta t_{SR}(\mathbf{x})$ at the reciprocal-shot locations are determined by linear interpolation based on a Delaunay triangulation, as outlined above.

The system of linear equation (2.5) is strongly over-determined for a typical 3D recording and can be solved for parameters t_S by using the Least Squares or SIRT methods described in Chapter 4 of this thesis. However, because of the simple form of the coefficients in this linear system (only equal -1, 0, and 1), a SIRT-type approximate solution can be used:

$$t_{Si} = -\frac{\lambda}{\left(N_{Ri}+1\right)} \sum_{j=1}^{N_{Ri}} \left(\delta t_{Si,Sj} - t_{ui} + t_{uj}\right), \tag{2.6}$$

which is applied iteratively until all $\delta t_{Si,Sj}$ become small. Here, N_{Ri} is the number of shots reciprocal to shot number i, and $\lambda \leq 1$ is a factor used for damping the iterations in order to prevent oscillations when the number of reciprocal shots is small. This correction reduces the residual average travel-time misfit of shot #i with all of its reciprocal shots.

As a result of iteratively applying the shot-time corrections (2.6), the average travel-time discrepancies between the shots become equal zero, and consequently any systematic travel-time errors related to shot timing or depth uncertainties are removed prior to the inversion.

2.3 Receiver static terms

Similarly to the shots, receivers may also have systematic travel-time variations caused, for example, by small-scale near-receiver heterogeneity or time picking problems. Such travel-time variations are incorporated in our model as the "receiver static" terms, which may be similar to the "short-wave" statics in GLI3D program by Hampson-Russell. The receiver static terms $t_R(\mathbf{x})$ in eq. (2.2) can be estimated by using an iterative procedure similar to the shot time correction (2.6):

$$t_{Ri} = \frac{1}{N_{Si}} \sum_{j=1}^{N_{Si}} \left[t\left(\mathbf{x}_S, \mathbf{x}_R\right) - \left(t_u + t_S\right) - t_d\left(\mathbf{d}_{SR}\right) \right]. \tag{2.7}$$

This sum, performed over all shots covering receiver #i, represents the average travel-time deviation associated with this receiver location. Such a correction would typically absorb the receiver elevation static. As a result of separating this term, the distribution of $\delta t_{SR}(\mathbf{x})$ values becomes centred and its variance reduced, and the offset-dependent trend $t_d(\mathbf{d}_{SR})$ can therefore be determined more accurately.

2.4 Travel-time data quality control and editing

Once the shot travel times are decomposed according to eq, (2.2), parameters of $\delta t_{SR}(\mathbf{x})$ can be used for identifying erroneous travel-time picks. This is particularly important if automatic pickers have been used, especially those done in other software, such as ProMAX, which does not recognise the geometrical and reciprocal relationships between the travel times from different source-receiver pairs.

Because the range of $\delta t_{SR}(\mathbf{x})$ values in eq, (2.2), should be moderate and centred at zero, its anomalous values can be easily identified. For example, we can measure the standard deviation:

$$S[\delta t_{SR}] = \frac{1}{(N_R - 1)} \sum_{j=1}^{N_R} \delta t_{SR}^2 \ . \tag{2.8}$$

This quantity represents the average width (dispersion) of the statistical distribution of $\delta t_{SR}(\mathbf{x})$. Values that are too large compared to this standard deviation, for example such that $|\delta t_{SR}(\mathbf{x})| > 3S[t_{SR}]$, can be considered outliers and removed from further analysis and inversion. Seismic traces containing such errors can also be examined for geometry errors or analysed interactively, as described below.

Similarly, the statistics of reciprocal travel-time misfits can be measured, and their standard deviation determined. A histogram of reciprocal travel-time errors defined as: $\Delta t_R = t(\mathbf{x}_S, \mathbf{x}_R) - t(\mathbf{x}_R, \mathbf{x}_S)$is shown in Figure 2.2. This histogram shows that the range of reciprocal travel-time mismatches is with in about ± 10 ms. Shots with significantl reciprocal-time mismatches can be re-examined or excluded from further travel-time analysis and reflection imaging. For example, in the Beaver Ranch data, such procedure helped to quickly identify shots with errors in source-receiver geometry patterns. This was the most common (and also widespread) source of travel-time errors in this dataset.

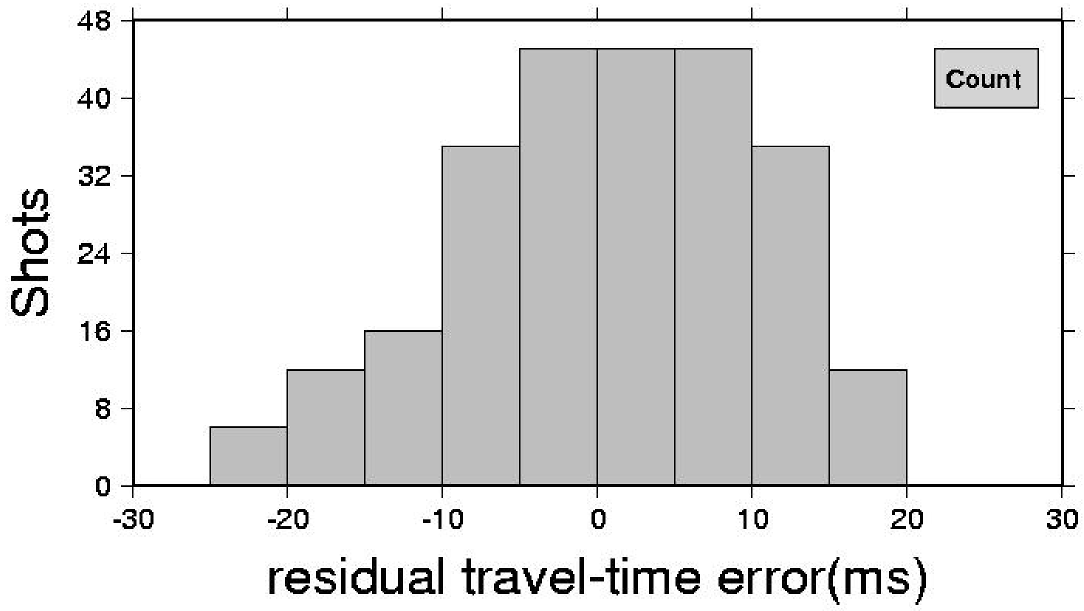


Figure 2.1: Histogram of residual reciprocal travel-time errors in the selected data subset.

2.5 Geometry pattern quality control

In cases with complex 3D layouts, such as the dataset of this study, geometry errors may present great difficulties in analysing the travel times. If present in the survey documentation, most geometry errors cannot be identified during data loading and binning, but they can be recognized upon examination of the first-break travel-time patterns. However, this is a difficult and extremely tedious procedure requiring repeated visual inspections and periodic re-binning of the entire dataset, which was impractical in this study which contained about 15000 shots. However, the statistics of the travel-time distribution [eq. (2.8)] can be used for detecting and correcting such pattern errors.

Automatic geometry pattern correction is a currently ongoing effort (Morozov, 2009, personal communication). In an experimental procedure, the patterns were randomly perturbed by shifting the receiver numbers up or down for each line of shot-receiver pattern. For each random modification of the pattern, shot travel-time field was decomposed by using equations. (2.1 - 2.2), the standard deviations (2.8) were measured, and the number of travel-time outliers were determined. By using Genetic Algorithms, several of such modifications were examined simultaneously, and those producing the least outliers and the smallest $S[t_{SR}]$ were intermixed and randomized again and kept in the analysis. After many (~1000) trials, the algorithm found the patterns providing the best-quality decomposition (2.1 - 2.2) and reported whether the originally specified pattern appeared to be correct.

2.6 Construction of the starting depth model

The efficiency of iterative inversion algorithms strongly depends on the proximity of the initial models to the true solution. With the chosen type of model parameterization (constant-velocity, variable-thickness layers) an efficient and fully automatic procedure exists for deriving a high-quality starting model. This procedure is based on the *τ-p* parameterization of travel times described below.

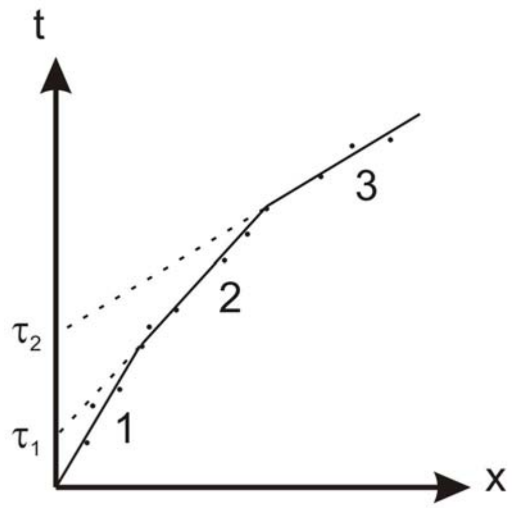


Figure 2.2: *t-x* decomposition of travel times. Picked first-break times are schematically shown by dots. Straight line segments approximately fit these travel times. These segments can be thought of as head-wave travel times. τ_1 and τ_2 are the intercept times of these head waves.

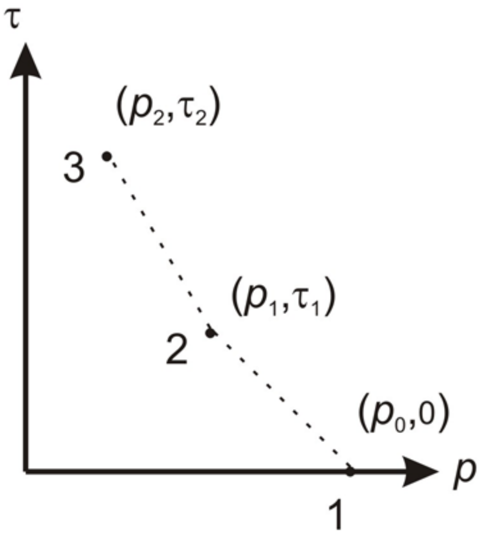


Figure 2.3: *τ-p* plot corresponding to *t-x* plot of Figure 2.2. Parameter *p* is the inverse of the velocity and is measured by the slopes of each of the three lines in Figure 2.2. *τ* values are the corresponding intercept times.

In order to define the τ-*p* parameterization, consider the typical first-break travel times schematically shown in Figure 2.2. These travel-time data (Figure 2.2) can be approximated by a series of head wave segments with increasing velocities (Figure 2.2). Any number of layers (lines in Figure 2.2) can be used to fit the same data in order to achieve sufficient accuracy. The intercepts and slopes of the straight lines labelled 1, 2, and 3 in Figure 2.2 represent the *τ-p* transform parameters, which are shown in Figure 2.3.

The *τ-p* formulation of the travel-time problem is convenient for inversion for the depths in a layered model. In a real dataset with dense first-break- recording, a nearly-continuous travel-time curve can be approximated by an infinite number of straight lines, which would correspond to a continuous (*τ,p*) curve (dashed line in Figure 2.3).Such a continuous curve could result from a continuous depth-velocity distribution, which can be obtained from the Herglotz-Wiechert transform (Aki and Richards, 2002):

$$z(v) = -\frac{1}{\pi}\int_{v_0^{-1}}^{v^{-1}} \frac{X(p)}{\sqrt{p^2 - v^{-2}}}\,dp\,, \tag{2.9}$$

Here, $z(v)$ is the depth as a function of wave velocity, p is the ray parameter (travel-time moveout slope), and $X(p)$ is the source-receiver distance at which this ray parameter is observed.

To construct a starting velocity-depth model for subsequent tomographic inversion, time-distance trends $t_d(\mathbf{d}_{SR})$ are extracted for each shot and "re-sorted" into common-midpoint (CMP) travel times. With moderate lateral variability, such CMP travel times are related to the velocity distributions beneath the corresponding midpoints. These velocity-depth distributions can then be obtained by the 1D τ-p inversion methods (Bessonova et al., 1974). Because this 1D inversion is performed at each (x, y) location (in our implementation, this is done at the receiver locations and further interpolated), the resulting model becomes spatially-variant. In 3D, a similar inversion was performed by Morozov et al. (2005) by using long-range seismic data.

Note that the starting-model inversion described here does not require interactive determination of any depths, velocities, or control points. The interpolated travel-time field (2.1 - 2.2) contains sufficient information for determining the near-surface velocity and layer depths automatically at all locations covered by the source-receiver spreads. For example, in the interactive program, the travel times and depth-velocity profiles can be examined interactively at any location (Figure 2.5d) and before performing the model-based inversion. The resulting 3D starting velocity model is derived at every point of the model grid and is quite detailed (Figure 2.4). It reproduces the observed travel-times well and needs to be only moderately adjusted by the inversion.

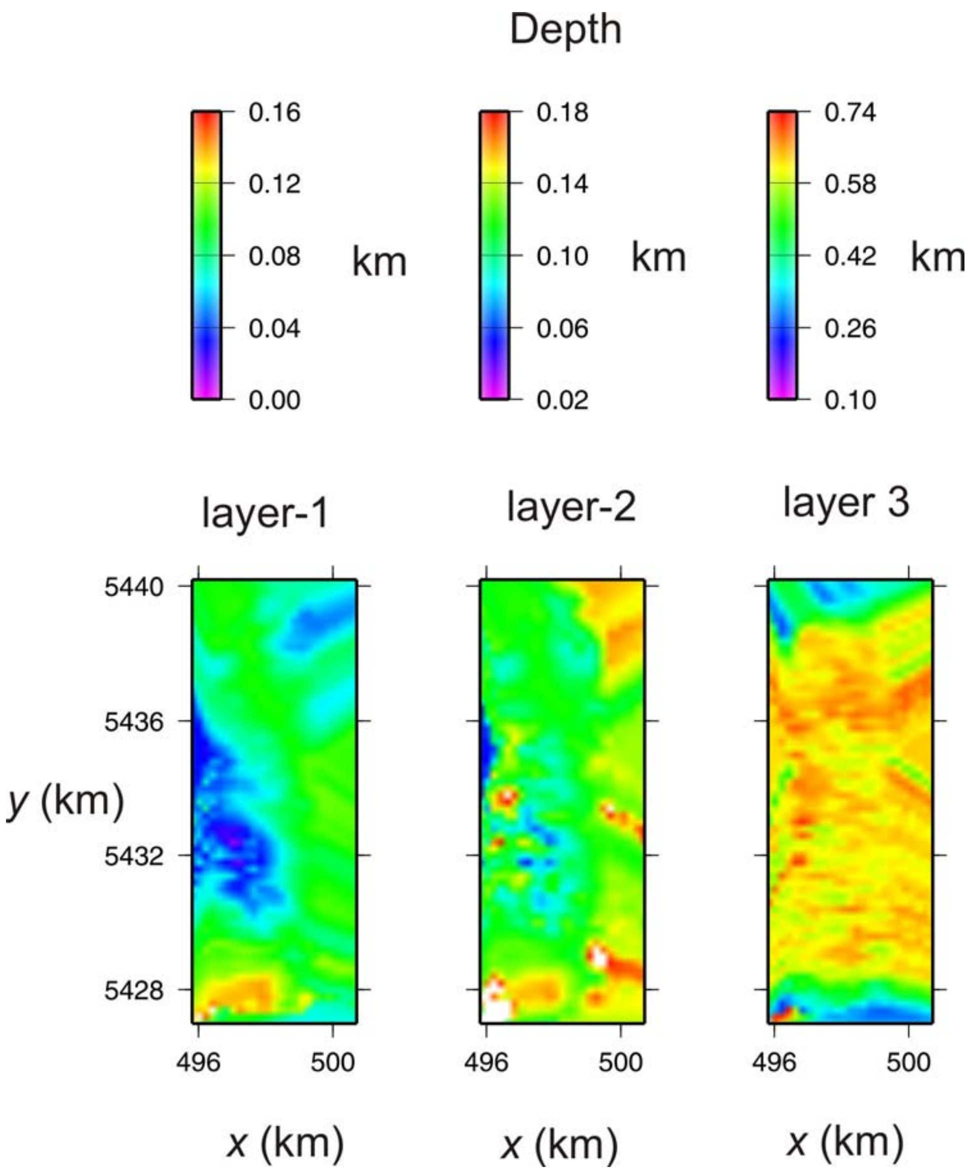


Figure 2.4: Depths to the refractors with velocities = 0.667 km/sec in layer-1, 1.6 km/sec in layer-2, 2.0 km/sec in layer-3, and 3.0 km/sec below, obtained from τ-p parameterization.

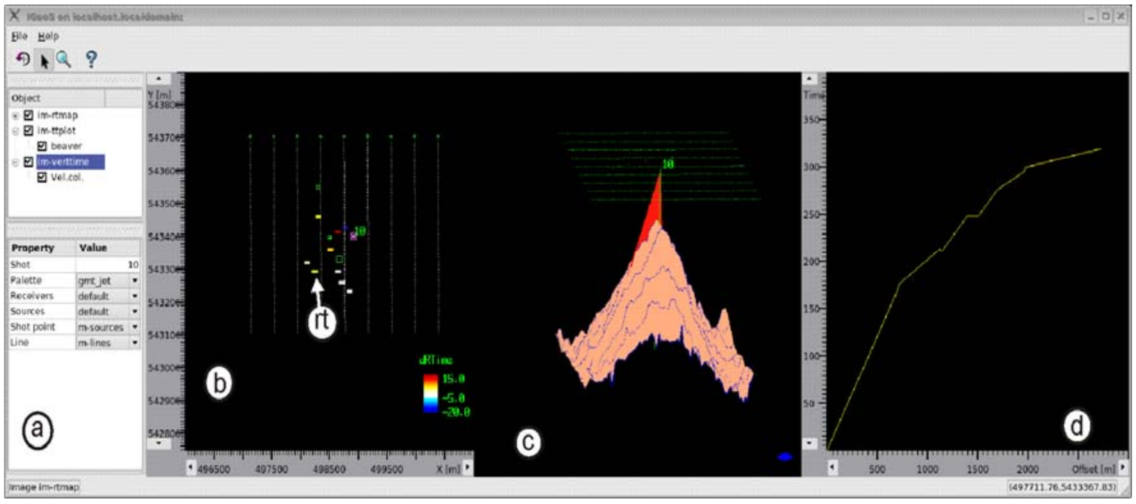
2.7 Manual and automatic first-break picking

The time-field decomposition equations (2.1 - 2.2) also allows improved manual and automatic picking of the travel times (Morozov and Jhajhria, 2008). Unlike how this is traditionally done (for example, in ProMAX), this decomposition explicitly takes into account the 3D geometry and the character of the wavefield corresponding to propagation in a layered velocity model. This ensures a significantly more intelligent and self-consistent picking of the entire dataset from only a few “seed” shots. This is also an ongoing work which is not included in this thesis and only outlined here.

Travel-times predicted from reciprocal shots can be sufficiently close to allow their automatic refinement by locating the peak amplitudes in their vicinities. A still better approach consists in “training” the program by interactive selection of a waveform from one shot, which is further cross-correlated with the records in the vicinities of first breaks. In other shots, this “seed” waveform is selected automatically from receivers located near shots that have already been picked. The waveforms collected from each shot record can be saved and used later, for example, for deconvolution.

2.8 Visualization and integration with IGeoS processing system

Three-dimensional (3D) graphics using the OpenGL modeling language opens new possibilities for improved interaction with the data, resulting in an improved efficiency of the procedure and quality of the inversion. Travel-time surfaces from different shots can be viewed and examined for consistency. Selection of shots and seismic lines for viewing and travel-time picking is performed visually from an interactive base map (Figure 2.5). The images can be zoomed, panned, and rotated smoothly, as in most seismic interpretation programs. Many graphical options (colours, lines, fills, palettes) are selectable from context-sensitive goCad-like property menus (Figure 2.5). Drop-down menus, status lines and tool tips improve the interpreter’s experience.



Property	Value
Shot	10
Palette	gmt_jet
Receivers	default
Sources	default
Shot point	m-sources
Line	m-lines

Figure 2.5: Interactive travel-time analysis: a) tool Property menu, b) map of selected shot, rt) reciprocal-time mismatch indicators in eq 2.4. c) 3D display of shot (tan colour) and reciprocal (red) times, d) vertical travel-time at a midpoint selected in base map b). A 10-shot data subset is used for clarity of display.

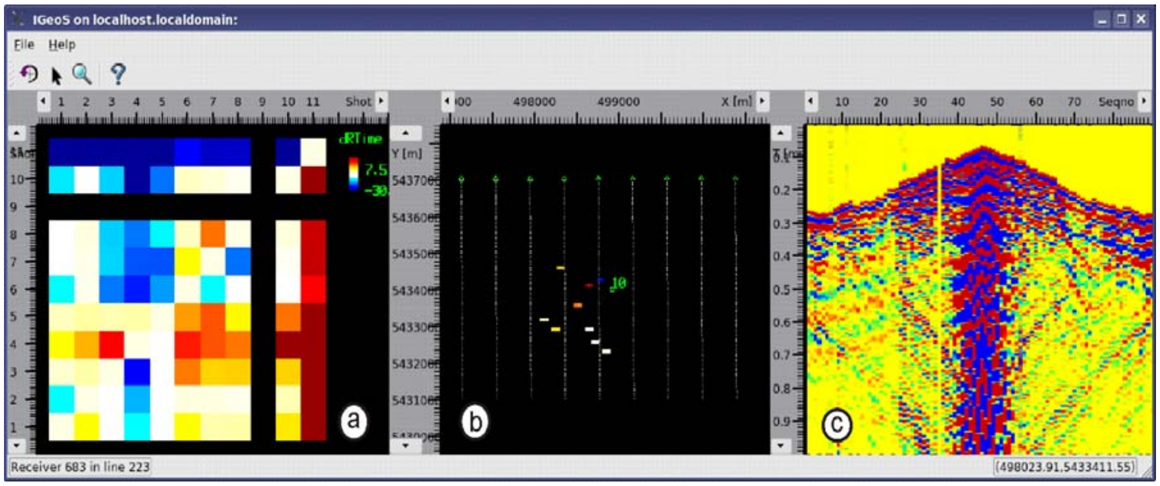


Figure 2.6: Interactive and automatic surface-consistent travel-time picking: a) reciprocal-time shot mismatch diagram. Colours represent the reciprocal-time misties in eq. 2.4 b) map of the selected shot with reciprocal-time mistie indicators as in Figure 2.5b; c) seismic section of the selected line for picking. Shots and lines can be selected from panels a) and b) and time reduction is applied. Reciprocal times from travel-time surfaces (Figure 2.5) can be used to guide picking.

The design of interactive displays is unusual and takes advantage of integration with a large data processing system (IGeoS; Chubak et al., 2007). The contents of the displays such as selection of images, objects, and their options in Figures (2.6) is performed entirely by the user, in the form of processing flows similar, for example, to those used in ProMAX. Other objects not directly related to the refraction static problem (e.g., base maps, gravity or magnetic models, wells, or seismic cross-sections) can also be included. Note that the images shown above were constructed without any "real" computer-language programming. The system's Graphical User Interface can be used for maintaining and executing the flows.

In addition to customisable graphics, integration with the processing system brings other significant advantages. Data input/output, visualization, PostScript plotting, seismic and potential-field data processing is performed "on the fly" by other (currently over 200) IGeoS tools. The resulting code has only to deal with the refraction statics problem and is therefore relatively compact. Software maintenance is also simplified by an automated code distribution system including tools for web-based collaboration (Morozov et al., 2007).

3. Forward Modeling

In general, forward modeling is the estimation of the observed property given some values of model parameters (Tarantola, 1987). In the refraction statics problem, forward modeling is the calculation of travel times using model parameters (layer velocities and depths), and is accomplished by ray tracing.

3.1 Model parameterization

The model used for ray tracing and inversion for statics consists of layers of constant velocities $V_l = 1/p_l$ where l is the number of the layer, V_l is its velocity, and p_l is the slowness. As it is typical in refraction cases using head waves, velocity normally increases from the top to the bottom of the model, corresponding to p_l decreasing with increasing l.

In such a parameterization, values of p_l are preset and kept constant during the inversion, and only the depths to the bottoms of the layers are modified. In principle, there is no limit on the number of layers and density of sampling in p_l, and therefore nearly continuously changing velocities can be considered. As shown below, such a parameterization scheme is convenient for ray tracing and inversion. Most importantly, as was explained in section 2.6, it also allows efficient and automatic construction of initial models. In complex situations with velocity inversions and low-velocity zones (such as caused by overturned layers), this model can be further improved by travel-time tomography.

As the independent model parameters derived by the inversion, we chose the depths of the bottoms of the layers sampled at regular spatial grids. For each layer #l, the depth to its bottom, $z_l(x,y)$ was discretized at a grid. Figure 3.1 shows the Cartesian grid covering the area of interest. Square cells are used, with the sizes of cells denoted as Dx_l. There are Nx_l and Ny_l grid points in X and Y directions, respectively.

By using the layer-bottom depths $z_{i,j}$ (where i and j are the grid numbers in the X and Y directions, respectively) we can define the depth at point (x,y) within the grid by using bilinear interpolation:

$$z(x,y) = \sum_{i,j} \phi_{i,j}(x,y) z_{i,j} \equiv \sum_k \phi_k(x,y) z_k \,, \tag{3.1}$$

where $z_{i,j}$ are the depths at the grid nodes. In order to formulate the linear inverse problem, all values of $z_{i,j}$ from all layers need to be combined in a single model vector. Therefore, in the following, we need to differentiate between the two-subscript notation $z_{\mathrm{i,j}}$ and a single-subscript one: z_k, in which k spans all layer nodes within the model. Mapping between these two notations (second half of eq. 3.1) is accomplished by appropriate counting of all grid nodes:

$$k = \sum_{m=1}^{l-1} Nx_m Ny_m + Ny_l \cdot i + j \,. \tag{3.2}$$

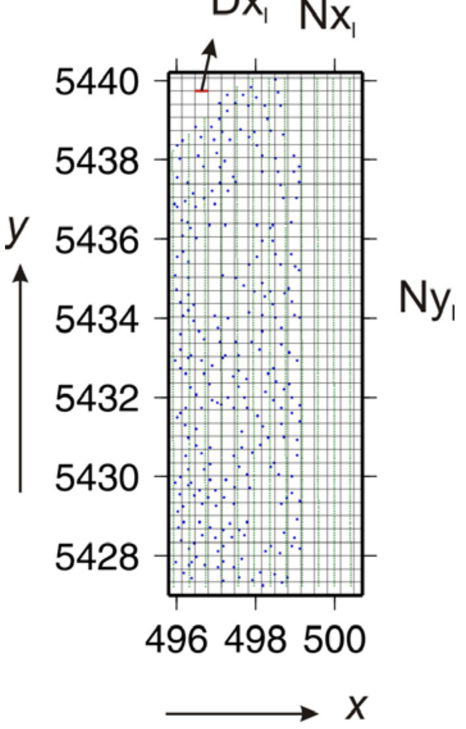


Figure 3.1: Inversion grid (black), seismic sources (blue) and receivers (green).

Explicitly, the depth at any point (x, y) in equation (3.1) can be written as:

$$z(x,y) = z_{i,j}\frac{(x_{i+1}-x)(y_{j+1}-y)}{(x_{i+1}-x_i)(y_{j+1}-y_j)} + z_{i+1,j}\frac{(x-x_i)(y_{j+1}-y)}{(x_{i+1}-x_i)(y_{j+1}-y_j)} + z_{i,j+1}\frac{(x_{i+1}-x)(y-y_j)}{(x_{i+1}-x_i)(y_{j+1}-y_j)} + z_{i+1,j+1}\frac{(x-x_i)(y-y_j)}{(x_{i+1}-x_i)(y_{j+1}-y_j)}, \quad (3.3)$$

This definition of *z(x,y)* completely defines a point inside the cell. Equation (3.3) represents a linear spline interpolation.

3.2 Linearization of the forward travel-time problem

Generally, the first-arrival time from source *S* recorded at receiver *R* is a function of all *N* model parameters:

$$t_{SR} = T_{SR}(z_1, z_2, z_3, ..., z_n). \quad (3.4)$$

Equations 3.4 for depths $z_1, z_2, ...z_n$ are non-linear and therefore they cannot be readily solved in terms of model parameters. Moreover, for an arbitrary layered model considered here, function T_{SR} can only be obtained by numerical modeling. Therefore, in order to find an inverse of equations (3.4), we need to use the perturbation theory to linearize the problem.

To linearize equation (3.4), consider a small perturbation in z causing the resultant perturbation in t_{SR}. By writing the Taylor series expansion in terms of δz, we have:

$$t_{SR}(\mathrm{z}+\delta\mathrm{z}) = t_{SR}(\mathrm{z}) + \sum_i \left.\frac{\partial T}{\partial z}\right|_{\delta z=0} \delta z_i + \mathrm{O}(\delta z^2), \quad (3.5)$$

where the summation is performed over all model depth nodes. For small perturbations, we ignore the higher-order terms in equation (3.5), leading to a linear approximation for the relation between the perturbations of layer depths and travel times.

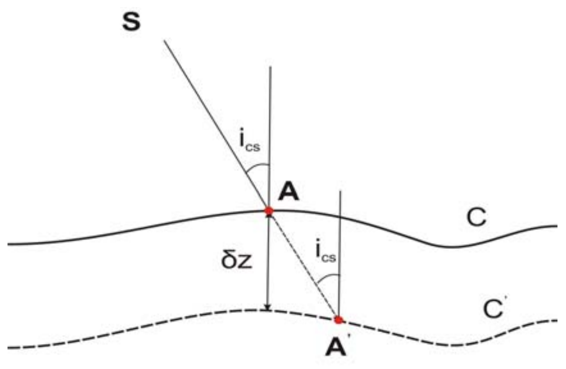


Figure 3.2: The ray strikes the interface at point **A** in unperturbed state. Perturbation of the interface shifts the refraction point to**A'**.

To express the partial derivatives in eq. (3.5), consider a perturbation of the interface shown in Figure 3.2. The entire interface is shifted downward by a small depth increment *δz*, causing the ray incident point to move from **A** to **A'**. Only the delay-time term changes in the resulting travel time (eqs. 1.4 and 1.5), and therefore:

$$\delta t_{SR} \approx \frac{\cos i_{cs}}{v_1} \delta z = p_1 \cos i_{cs}, \tag{3.6}$$

where $p_1 = 1/v_1$ is the slowness (ray parameter) of the layer.

Equation (3.6) is linear in δz and gives the change in the perturbed travel time due to the depth perturbation. Similar partial derivatives are accumulated for all nodes that belong to the grid cell penetrated by the ray.

3.3 Ray Tracing

Travel-time forward modeling methods fall into two broad categories: 1) wavefront propagation and 2) ray tracing. The first of this group is commonly used when travel times to large numbers of nodes within the model are required, such as in pre-stack Kirchhoff migration in 2D or 3D. The most popular solver of this type is based on solving the eikonal equation:

$$(\nabla\tau)^2 = p^2(x,y,z), \tag{3.7}$$

where τ is the travel time and p is the slowness of the medium at point(x, y, z). Surfaces $\tau =$ const in the solution $\tau(x,y,z)$ to this equation represent the first-arrival wavefront propagating within the medium described by the slowness distribution $p(x,y,z)$. Finite-differencing is the common method used for solving eq. (3.7), and such forward modeling can efficiently cover the whole receiver space. However, the accuracy of this method is limited by the use of discrete finite-difference modeling grids.

By contrast to eikonal time field propagation, ray tracing attempts to accurately reproduce the ray shapes and travel times between each pair of sources and receivers. This method is most often used in seismic tomography and also in the present approach. With the chosen layered model parameterization, we now need to develop the ray tracing algorithm.

In this study, I tested several ray-tracing schemes, all of which were based on the delay-time concept [eq. (1.5)] and approximation of the rays as traveling within the vertical cross-section plane between the source and receiver. This resulted in 2D ray propagation that could be solved efficiently by finding the delay-time terms in eq. (1.5) for the corresponding ray parameters.

The key difference in the ray tracers considered was in the ways for locating the positions of intersections of the downgoing rays with the $z(x,y)$ interfaces. Three methods using the quadratic approximation, numerical bisection, and simple delay-time formulae were studied.

In each of the three ray tracers, the key step is the propagation of a ray within a single layer and between two points on its upper surface. This procedure is further reduced to finding the travel-time from a point on the upper surface of the layer (**S** or **R** in Figure 1.3) to another point at its bottom (e.g., the source-receiver midpoint; point **X** in Figure 1.3). Also, it is convenient to normalize all coordinates by dividing them by the grid increment *Dx*. This transformation makes grid increments equal 1 in each direction and places the origin of the grid at point (0,0) for each layer.

The simplest ray tracing is based on the delay-time formula (eq. 1.1). Depth z_l is calculated below the ray entry point in the layer. This method does not involve iterations and results in a very fast calculation of travel times. However, this method is less accurate than the bisection and quadratic methods described below, because it does not account for horizontal variations of the delay times in the vicinities of the source and receivers. These accurate methods are discussed in more detail in the following sections.

3.3.1 Ray tracer using quadratic approximation for *z*(*x*,*y*)

Consider a point **P** (red dot, Figure 3.3) in a 3D space. The point is located inside the cell formed by four nodes with indices $(i,j),(i+1,j),(i,j+1),(i+1,j+1)$, which also equal their respective normalized (x,y) coordinates. The corresponding corner node depths are $z_{i,j}$, $z_{i+1,j}$, $z_{i,j+1}$, and $z_{i+1,j+1}$ (Figure 3.3). Let us denote the horizontal coordinates of point **P** as $x = i + u$ and $y = j + v$, respectively, where u and v are the relative coordinates lying within the (0, 1) range.

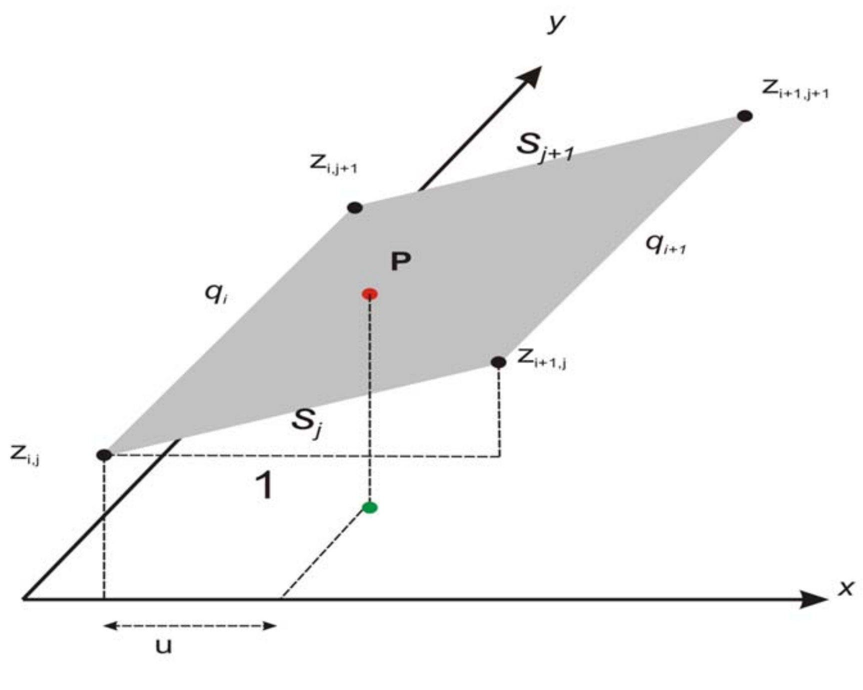


Figure 3.3: Point **P** is assumed to be the intersection of a incident ray and layer interface. In order to find the point **P**, we take the projection of point **P** onto the *xy* plane. S_j, S_{j+1}, q_i, q_{i+1} are the slopes along the *X* and *Y* directions for this cell.

Point P is projected onto the *xy* plane, and its offset from node (i, j) is denoted (u, v) With this parameterization, the slopes of the four sides, denoted q_i, q_{i+1}, s_j, s_{j+1} given by the following equations Figure (3.3):

$$q_i = \left.\frac{\partial z}{\partial y}\right|_{x=i,y=j} \quad , \tag{3.8}$$

$$s_j = \left.\frac{\partial z}{\partial x}\right|_{x=i,y=j} \quad . \tag{3.9}$$

The depth at any point inside the cell can be written as a linear combination of depth values at the four corners (equation (3.1)):

$$z(x,y) = z_{ij} + au + bv + cuv\,, \tag{3.10}$$

where:

$$a = s_j = z_{i+1,j} - z_{i,j}\,, \tag{3.11}$$

$$b = q_i = z_{i,j+1} - z_{i,j}\,, \tag{3.12}$$

$$c = q_{i+1} - q_i = \left(z_{i+1,j+1} - z_{i+1,j} - z_{i,j+1} + z_{i,j}\right), \tag{3.13}$$

We now calculate the position of the ray intersection point by using the following approach: we consider a horizontal distance $l_{\text{intersect}}$ between the ray origin (u_S, v_S) and the intersection point (Figure 3.4):

$$u(l_{\text{intersect}}) = u_s + e_u l_{\text{intersect}}\,, \tag{3.14}$$

$$v(l_{\text{intersect}}) = v_s + e_v l_{\text{intersect}}\,, \tag{3.15}$$

where e_u, and e_v are the directional cosines of the ray in the horizontal plane.

The depth to the layer boundary along the ray can then be written as:

$$z(l_{\text{intersect}}) = Dz\left(\frac{l_{\text{intersect}}}{\tan\alpha} + w_s\right) + z_{i,j}\,, \tag{3.16}$$

where α is the angle of incidence of the ray from the vertical and $w_s = z_s - z_{i,j}$.

By equating the depths in equation (3.10) and (3.16), we find the equation for l corresponding to the intersection of the ray with the layer bottom:

$$\left(w_s + l_{\text{intersect}} \tan^{-1}\alpha\right)Dz + z_{i,j} = \left(z_{i,j} + a' u_s + b' v_s + c' u_s v_s\right) + (a' e_u + b' e_v + c' e_u v_s + c' e_v u_s)\, l_{\text{intersect}} + c' e_u e_v l_{\text{intersect}}^2, \tag{3.17}$$

Equation (3.17) is quadratic in $l_{\text{intersect}}$, and it can be written as:

$$A l_{\text{intersect}}^2 + B l_{\text{intersect}} + C = 0, \tag{3.18}$$

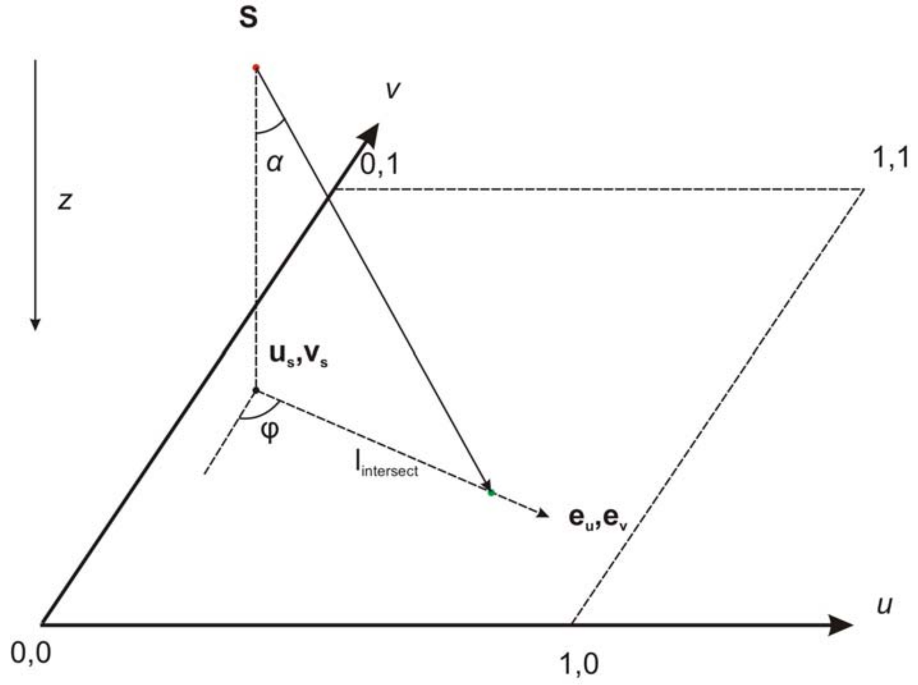


Figure 3.4: The ray strikes the boundary at the point in (u,v) plane shown in green dot. The angle of incidence is α and the angle ϕ is the azimuth of the incident ray. e_u and e_v are the direction cosines.

where:

$$A = c' e_u e_v, \tag{3.19}$$

$$B = a' e_u + b' e_v + c'\left(e_u v_s + e_v u_s\right) - \tan^{-1}\alpha, \tag{3.20}$$

$$C = z_{i,j} + a' u_s + b' v_s + c' u_s v_s - z_s, \tag{3.21}$$

When $A \neq 0$, the solution to the equation (3.18) is given by:

$$l_{\text{intersect}} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}, \tag{3.22}$$

and for $A \approx 0$, equation (3.22) reduces to:

$$Bl_{\text{intersect}} + C = 0, \tag{3.23}$$

with the following solution when $B \neq 0$:

$$l_{\text{intersect}} = \frac{-C}{B}. \tag{3.24}$$

However, if A is small and $B = 0$, the solution is given by:

$$l_{\text{intersect}} = \left(z_{i,j} - z_s\right)\tan\alpha . \tag{3.25}$$

The flowchart in Figure 3.5 lists the steps for finding the intersection point of the ray with the layer-bottom interface. The ray tracer returns the correct position of the intersection at the interface.

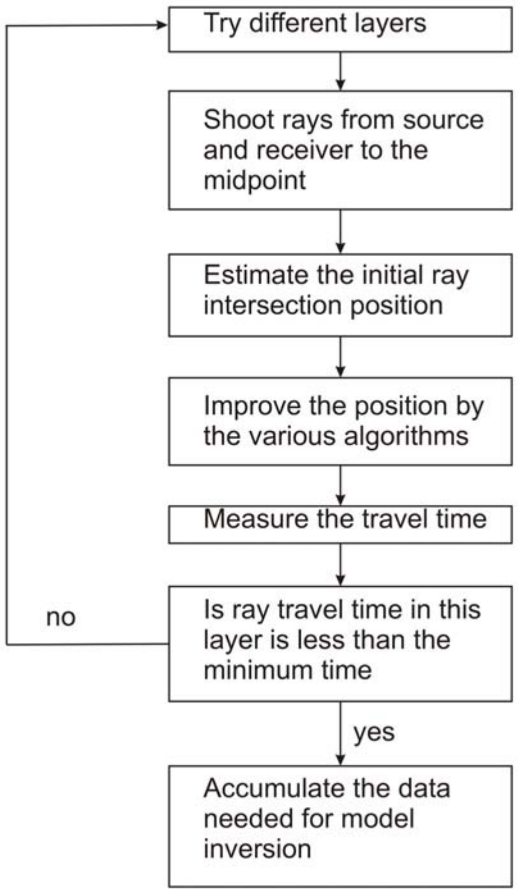

Try different layers
Shoot rays from source and receiver to the midpoint
Estimate the initial ray intersection position
Improve the position by the various algorithms
Measure the travel time
Is ray travel time in this layer is less than the minimum time
no
yes
Accumulate the data needed for model inversion

Figure 3.5: Schematic flowchart of ray tracer.

The ray tracer using the quadratic method is fast and accurate; however, it only works consistently when interface dips are relatively small. The reason for poor steep-dip handling is the second-order approximation of the surface which remains valid only when the ray endpoint lies within or close to the selected cell. For steep interface curvatures, the end point may fall in other cells, and the procedure has to be repeated with different (i, j) values. However, this still does not guarantee convergence, and infinite algorithm loops may result. The instability in respect to steep dips is difficult to control during inversion. Therefore, for inversion, I used the following bisection-based method, which is also quite accurate and unconditionally stable.

3.3.2 Bisection Method

In an alternate approach to finding the intersection of the ray with the bottom of the layer with variable-depth interface (Figure 3.6), I used an approach based on numerical equation solving by using an iterative bisection technique. Figure 3.6 shows the geometry of the problem. The ray is incident at the interface at angle α relative to the vertical. The ray parameter is p_{i-1} in the first layer and p_i in the second layer. The goal is to find the distance ℓ (Figure 3.6) measured from the position of the source-receiver midpoint, such that the angle of incidence equals α.

Using the Snell's law, the equation for the incident ray can be written as:

$$\sin[\alpha(\ell)] = \frac{p_i}{p_{i-1}} = \frac{L-\ell}{\sqrt{(L-\ell)^2 + z^2}}, \tag{3.26}$$

where z is the depth at point **T**. The algorithm starts from values $\ell = 0$ and $\ell = L$ and iteratively bisects this interval until it finds a value of ℓ which satisfies equation (3.26). Once the value of ℓ is found, travel-times from location point **S** to location point **T** can be accurately calculated. Similarly, the receiver-side travel times are also calculated. The total travel time is the sum of travel times for the source and receiver from the surface to the intersection point with the base of the layer (for example, point **T** for the source in Figure 3.6), and time taken by ray to travel horizontally.

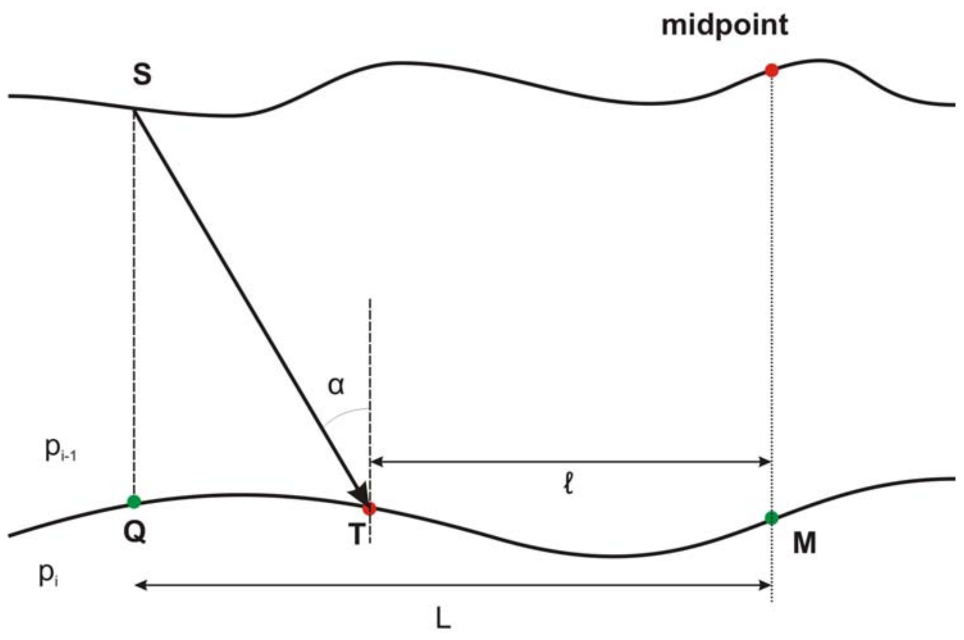


Figure 3.6: Notation in eq. (3.26) used in bisection ray-tracing method. ℓ is the distance of the ray point from the midpoint. p_{i-1}, p_i are the ray parameters in the corresponding layers.

3.3.2.1 Accuracy of the bisection ray tracer

The accuracy of the bisection ray tracer was established by comparing the calculated travel times with the theoretical first-break travel times. A flat one-layer model at the depth of 600 m with direct wave velocity of 0.667 km/sec and base layer velocity of 1.667 km/sec was used. Two sets of source and receiver spreads were used. The first spread had the source and receivers aligned along the ***x***-axis (black line in Figure 3.7a), and the second spread had the source and receivers at 45° to the ***x***-axis (red line in Figure 3.7a). A comparison of reduced travel time for the two spreads is shown in Figure 3.7a, and an error plot is shown in Figure 3.7b. The errors were calculated by subtracting the travel times obtained from theoretical calculation from bisection method ray tracing travel times. The errors in both cases were of the order of 10^{-11} (Figure 3.7b), showing that the bisection travel-time modeling is practically perfectly accurate for the selected type of depth models Similar tests were also conducted with theoretical models containing undulating interfaces and showed similar results.

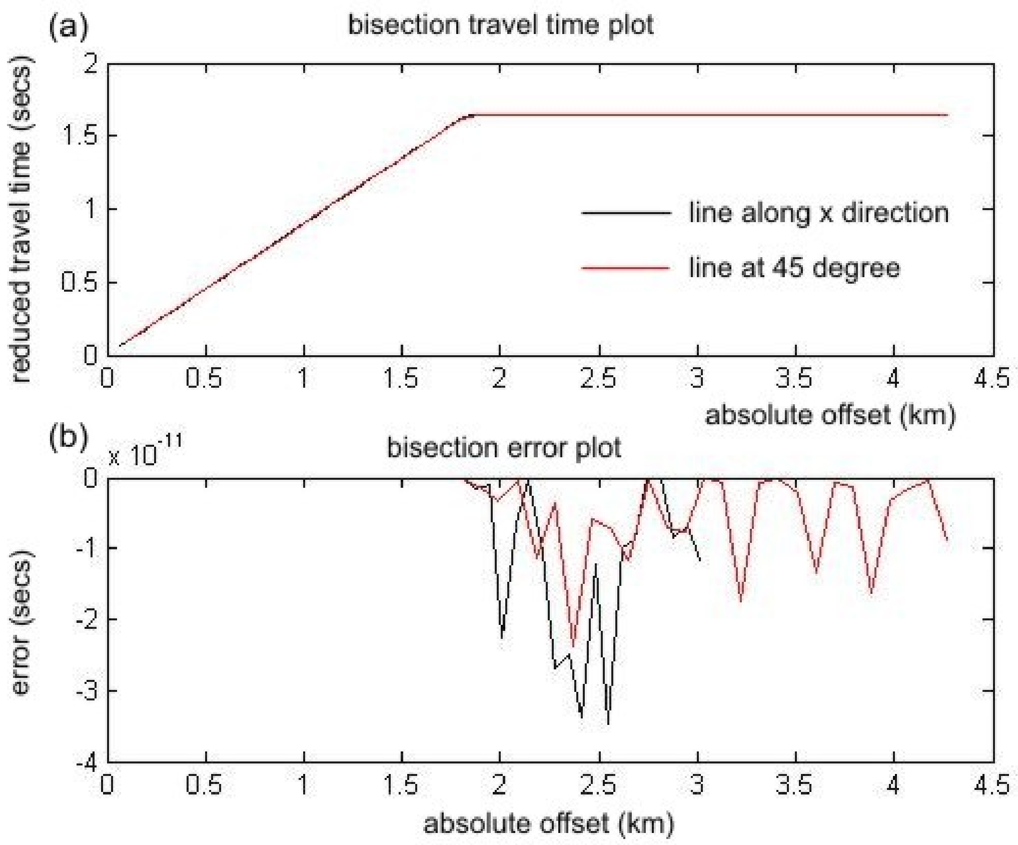


Figure 3.7: a) Travel times plotted with reduction velocity of 1.667 km/sec for one-layer horizontal model at 600-m depth. Black line corresponds to the source-receiver aligned along the *x*-axis, and red - for a line at an angle of 45°. b) Corresponding travel-time errors from the theoretical travel times.

4. Inversion

In order to solve the variations of first-arrival travel times for the depths of model layers, we need to express the inverse travel-time problems in a matrix forms and discuss the approaches for their solution. To define the inverse, the forward travel-time problem needs to be formulated as a system of linear equations:

$$\mathbf{d} = \mathbf{Lm} \,. \tag{4.1}$$

Here, **d** is the data (picked first-break travel times) vector of length N_d, **m** is the model vector of size N_m, and **L** is the forward kernel matrix of size $N_d \times N_m$ representing the linearized results of ray tracing, as discussed above. Inversion of equation (4.1) consists in solving for model vector **m**, given the data vector **d** matrix **L** are known. A formal solution to equation (4.1) is:

$$\mathbf{m} = \mathbf{L}^{-1}\mathbf{d} \,. \tag{4.2}$$

However, in most practical cases, the exact inverse of equation (4.1) does not exist, and the inverse given in equation (4.2) still needs to be constructed. Such an inverse is referred to as the Generalized Linear Inverse (GLI) and denoted $\mathbf{L}_g^{-1}$. The key property of such a solution is in **m** being sought as a linear function of **d**, with the inverse operator ($\mathbf{L}_g^{-1}$) dependent only on **L**. Many forms of such inverse solutions exist, each of them minimizing some form of the "data error" in equation (4.1) (Menke, 1984). The error is often defined by using various vector norms, which are functions that assign a single positive value ("norm") to the vector of data misfits **r**:

$$\mathbf{r} = \mathbf{d}_{\mathbf{obs}} - \mathbf{d}_{\mathbf{est}} \,, \tag{4.3}$$

The difference in the results using different norms, denoted by $||\mathbf{r}||$, is due to the amount of importance attributed to the various aspects of vector **r**. If outliers (large spurious values of r_i, such as caused by errors in travel-time picking) are present, the so-called L_1 norm is preferable:

$$||\,\mathbf{r}\,||_{L_1} = \sum_{i=1}^{N_d} |\,\mathbf{r}_i\,| \,. \tag{4.4}$$

However, for most practical purposes, the second-order, L_2 norm is most convenient:

$$||\,\mathbf{r}\,||_{L_2} = \sum_{i=1}^{N_d} |\,\mathbf{r}_i\,|^2 \,. \tag{4.5}$$

The L_2 norm is the most often used because it represents the distance between two points in a Euclidian space of data errors, and efficient matrix techniques for minimizing this norm exist. The mean data (travel-time) error associated with this norm is usually called the root-mean square (RMS) error:

$$\delta_{RMS}\mathbf{r} = \sqrt{\frac{||\,r\,||_{L_2}}{N_d}} \,. \tag{4.5a}$$

The generalized inverse (4.2) minimizing the L_2 error of data residuals (4.3) is called the Least Squares inverse. Its key properties will be discussed in the following sections in application to refraction statics. Note that strictly speaking, the broadly used program GLI3D, which took its name from the Generalized Linear Inverse, in fact does not use the GLI method. The Generalized Linear Inverse (Backus and Gilbert, 1968; Menke, 1984) was defined as deriving the model by a single matrix multiplication. The GLI3D program appears to use a non-linear, iterative Least-Squares inverse (Hampson and Russell, 1984); the same type of inversion approach is also used in this thesis.

Depending on the types of algorithms involved, inverse (4.2) may employ direct (single-step) or iterative (multiple-step) methods. In the statics problem, iterative methods appear most appropriate, because they are well-suited for large numbers of unknowns and sparse matrices **L**. In addition, in seismic ray-tracing, the forward travel-time problem (4.1) is non-linear and requires linearization that is also commonly achieved by iteration. In the following sections, I discuss the most commonly used iterative Least Squares methods: the Kaczmarz method, Simultaneous Iterative Reconstruction Technique (SIRT), Conjugate-Gradient, and the Least-Squares QR factorization (LSQR). From eqs. (3.1) and (3.6), the elements of matrix **L** are

$$L_{ij} = \phi_j(x,y)p_l\cos\theta_l(x,y), \tag{4.6}$$

where i is the ray number, j is the model cell number, l is the number of the model boundary containing cell j, (x, y) are the coordinates of the point at which ray i crosses thus boundary, p_l is the slowness and θ_l is the angle of incidence in lth layer.

4.1 Least-Squares Vector Norm and Sparseness

For a linear inverse problem, the Least Squares (LS) solution minimizing the L_2 vector norm is given, in matrix form, by (Menke, 1984):

$$\mathbf{m}^{\mathrm{est}} = \left[\mathbf{L}^{\mathrm{T}}\mathbf{L}\right]^{-1}\mathbf{L}^{\mathrm{T}}\mathbf{d}\,, \tag{4.7}$$

where T denotes the matrix transpose. This equation is valid when matrix $\mathbf{L}^{\mathrm{T}}\mathbf{L}$ is non-singular (has no zero eigenvalues), so that the inverse matrix $(\mathbf{L}^{\mathrm{T}}\mathbf{L})^{-1}$ exists. In the case of the travel-time problem discussed here, this requirement is satisfied when all cells are crossed by rays at (generally) varying angles.

However, in a travel-time problem, matrix $\mathbf{L}^{\mathrm{T}}\mathbf{L}$ may often be singular. No rays may pass through some of the cells, such as those lying outside of the area of data coverage. This means that we don't have any information to solve for the depths of those cells. The mathematical term for such a case is the "under-determined" inverse problem. In under-determined cases, we have more unknowns to solve for than the data allow. The problem may be "over-determined" when we have conflicting constraints for some elements of the model vector, so that all the data errors cannot be reduced to zero simultaneously. This typically happens when the number of data exceeds the number of unknowns, which is typical in the travel-time tomography or refraction statics problems. Finally, when both under-determined and over-determined conditions are encountered in parts of the model, the problem is called "mixed-determined". This is the typical case in the refraction statics problem discussed here.

The LS method (4.7) is suitable for over-determined problems, but for under-determined and mixed-determined cases, the inverse of the $\mathbf{L}^{\mathrm{T}}\mathbf{L}$ matrix does not exist, and the inverse requires regularization. Such regularization is achieved by adding a small term ε to each of the diagonal elements of matrix $\mathbf{L}^{\mathrm{T}}\mathbf{L}$ (van der Sluis, and van der Vorst, 1987):

$$\mathbf{m}^{\text{est}} = \left[\mathbf{L}^{\text{T}}\mathbf{L} + \varepsilon\mathbf{I}\right]^{-1}\mathbf{L}^{\text{T}}\mathbf{d}\,. \qquad (4.8)$$

The solution given by equation (4.8) no longer minimizes the L_2 norm, and it is called "damped LS" because it removes the uncertainty in the matrix inverse by slightly altering the solution.

Thus, a non-linear inverse problem which is linear for small perturbation can be iteratively solved if we are able to compute matrix $\mathbf{L}^{\text{T}}\mathbf{L}$ and vector $\mathbf{L}^{\text{T}}\mathbf{d}$ during ray-tracing iteration. Note that for solution (4.8), neither matrix $\mathbf{L}$ nor the complete data vector $\mathbf{d}$ need to be kept in computer memory in order to compute the inverse. Instead, only matrices $\mathbf{L}^{\text{T}}\mathbf{L}$ and $\mathbf{L}^{\text{T}}\mathbf{d}$ need to be evaluated during ray tracing. This is particularly advantageous in over-determined problems with large data redundancy over model parameters, such as the head-wave travel-time problem of this study. After each ray is traced, the corresponding model parameters (layer-cell depths) affected by the ray are identified, and the contribution to $\mathbf{L}^{\text{T}}\mathbf{L}$ and $\mathbf{L}^{\text{T}}\mathbf{d}$ are accumulated.

After tracing all rays, we can solve for the inverse problem by inverting the $\mathbf{L}^{\text{T}}\mathbf{L}$ matrix and iteratively moving toward the solution. The iterative Least-Squares method is a popular method for finding the inverse (Hampson and Russell, 1984). Model vector is updated after each iteration by using one of the approximations of equation (4.8), as discussed below. Most of the elements in $\mathbf{L}^{\text{T}}\mathbf{L}$ matrix are zero, and therefore this matrix is sparse. For sparse linear systems, there is a range of approximate methods available which make computation of the matrix $\mathbf{L}^{\text{T}}\mathbf{L}$ simple and efficient. Examples of such methods considered here include the Kaczmarz method, SIRT, Conjugate-Gradient method, and LSQR. Any of these methods can be used to solve the travel-time inverse problem. I used the SIRT method, whose definition with respect to other methods is given below.

4.2 Kaczmarz method

In numerical linear algebra, Kaczmarz's (1937) method solves systems of linear equations $\mathbf{d} = \mathbf{Lm}$. This approach is also known as one of the Algebraic Reconstruction Technique (ART), or "back-projection" methods. For efficient computation of the inverse matrix $(\mathbf{L}^{\mathrm{T}}\mathbf{L})^{-1}$, Kaczmarz's method replaces the matrix $(\mathbf{L}^{\mathrm{T}}\mathbf{L}+\boldsymbol{\varepsilon}\mathbf{I})$ in equation (4.8) with its diagonal:

$$\mathbf{D} = \mathrm{diag}(\mathbf{L}^{\mathrm{T}}\mathbf{L} + \varepsilon\mathbf{I}), \tag{4.9}$$

whose inverse is also a diagonal matrix. Because the inverse formula (4.8) is modified, the solution needs to be sought iteratively. Starting from a selected initial model, successive model vector updates **m** are obtained by "back-projecting" the data residual onto the model space. Thus, for q^{th} iteration, the model estimate is (van der Sluis and van der Vorst, 1987):

$$m_{\mathrm{j}}^{\mathrm{q}} = m_{\mathrm{j}}^{\mathrm{q-1}} + \sum_{\mathrm{i}} \frac{L_{\mathrm{ij}}\, r_{\mathrm{i}}^{\mathrm{q-1}}}{D_{\mathrm{i}}}. \tag{4.10}$$

This iteration is repeated until the model updates become negligibly small. Note that as in any over-determined or mixed-determined inverse problem, the convergence does not mean that data error vector **r** vanishes, but it becomes orthogonal to the model vector in the sense of equation (4.10).

Kaczmarz's method is one of the early iterative methods which served as the basis for other, more refined methods like the Simultaneous Iterative Reconstruction Technique (van der Sluis and van der Vorst, 1987) used in this thesis. Such methods are useful when the data volume becomes large. The main advantage of iterative methods is in their simplicity, stability, and the ability to combine iteration process with the iteration required for linearization of the problem. In addition, these methods greatly reduce the computer memory requirements, which result from the need to storing only the diagonal of the $\mathbf{L}^{\mathrm{T}}\mathbf{L}$ matrix and the "back-projected" data residuals $\mathbf{L}^{\mathrm{T}}\mathbf{d}$. Therefore, such methods can work with very large models ($N_{\mathrm{m}} \approx 10^7$, as in pre-stack reflection depth migration) and with unlimited data volumes. Model updates (4.10) can be accumulated during the iterations, which correspond, for example, to one-pass of ray tracing through a complex structure.

4.3 Simultaneous Iterative Reconstruction Technique

The SIRT-type (Simultaneous Iterative Reconstruction Technique) techniques are commonly used in earthquake seismology and medical tomography. In earthquake seismology, SIRT was used to calculate the location of an earthquake "simultaneously" with the velocity inside the Earth, and hence the name of the technique. The word "reconstruction" is used most often in medical tomography, where images are reconstructed from 2D slices by using the "centre slice theorem" (Radon, 1917). In earthquake seismology and refraction statics calculations, the term "reconstruction" can be thought as reconstruction of the velocity model by its iterative updating during repeated ray tracing.

The SIRT is a modification of Kaczmarz's method (van der Sluis and van der Vorst, 1987). In this method, the model vector is updated by using the following equation:

$$m_j^q = m_j^{q-1} + \frac{\alpha}{S_j} \sum_{\mathrm{i}} \frac{L_{ij}\, r_i^{q-1}}{D_i}, \tag{4.11}$$

Here, S_j is the normalizing factor which is the number of rays contributing to model parameter #j, q is the iteration number, and i is the ray number. I also added a factor α which can be used to control the convergence; this constant is found to be in the range of 0.2-0.3 for the models considered. Iterations are performed until the average step size becomes negligibly small. At this point, the residual error becomes related to the statistical data error or insufficient model assumptions. Once the SIRT code establishes the best model, further iterations may cause fluctuation of the error before converging back to the minimum-error level. At this point, our code terminates the ray-tracing/inversion loop.

The SIRT method is simple and stable, but it can be slow to converge to the solution if the initial model is far from the solution. There are other methods which can be used to calculate for faster convergence to the solution, for example, the Conjugate Gradients method and LSQR (Least-Squares QR factorization) methods. The following section explains these two approaches.

4.4 Conjugate Gradients and LSQR

The scheme for conjugate Gradients (Golub and van Loan, 1996) iteratively modifies the model in the direction of auxiliary vectors **p** (called conjugate gradients). Starting from the following initial values:

$$\mathbf{m_0} = \text{initial model}, \tag{4.12}$$

$$\mathbf{s}_0 = \mathbf{d} - \mathbf{L}\mathbf{m}_0, \tag{4.13}$$

$$\mathbf{r}_0 = \mathbf{p}_0 = \mathbf{L}^{\mathrm{T}}(\mathbf{d} - \mathbf{L}\mathbf{m}_0) = \mathbf{L}^{\mathrm{T}}\mathbf{s}_0, \tag{4.14}$$

$$\mathbf{q}_0 = \mathbf{L}\mathbf{p}_0, \tag{4.15}$$

these vectors are updated iteratively for i= 0, 1, 2…:

$$\alpha_{\mathrm{i}+1} = \frac{(\mathbf{r}_i, \mathbf{r}_{\mathrm{i}})}{(\mathbf{q}_i, \mathbf{q}_i)}, \tag{4.16}$$

$$\mathbf{m}_{i+1} = \mathbf{m}_i + \alpha_{\mathrm{i}+1}\mathbf{p}_i, \tag{4.17}$$

$$\mathbf{s}_{i+1} = \mathbf{s}_{\mathbf{i}} - \alpha_{\mathrm{i}+1}\mathbf{q}_i, \tag{4.18}$$

$$\mathbf{r}_{i+1} = \mathbf{L}^{\mathrm{T}}\mathbf{s}_{i+1}, \tag{4.19}$$

$$\beta_{\mathrm{i}+1} = \frac{(\mathbf{r}_{i+1}, \mathbf{r}_{i+1})}{(\mathbf{r}_i, \mathbf{r}_i)}, \tag{4.20}$$

$$\mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_{\mathrm{i}+1}\mathbf{p}_i, \tag{4.21}$$

$$\mathbf{q}_{i+1} = \mathbf{L}\mathbf{p}_{i+1}. \tag{4.22}$$

When the solution converges, both of the error vectors **s** and **q** become zero. For a linear inverse problem in an N_m-dimensional space, this scheme achieves convergence in exactly N_m iterations.

Another popular iterative method for solving equation (4.1) is the least-squares solution by QR factorization (LSQR). QR factorization is used to represent the matrix **L** as **L**= **QR**, where **Q** is an orthogonal matrix and **R** is an upper-triangular matrix. The factorization can be obtained by the classical Gram-Schmidt approach (Golub, Van Loan, 1996). In pseudo-code form, this decomposition is:

```
$R(1,1) = \|\mathbf{L(:,1)}\|_2$,                                   (4.23)
$\mathbf{Q}(:,1) = \mathbf{L}(:,1)/R(1,1)$,                          (4.24)
for i = 2:n
        $\mathbf{R}(1:i-1,i) = \mathbf{Q}(1:m,1:i-1)^T \mathbf{L}(1:m,i)$,     (4.25)
        $\mathbf{z} = \mathbf{L}(1:m,i) - \mathbf{Q}(1:m,1:i-1)\mathbf{R}(1:i-1,i)$,   (4.26)
        $R(i,i) = \|\mathbf{z}\|_2$,                                   (4.27)
        $\mathbf{Q}(1:m,i) = \mathbf{z}/R(i,i)$.                       (4.28)
end
```

At the ith step of this iteration, the ith columns of matrices **Q** and **R** are generated. After matrix **L** is factorized, it is easy to compute the inverse of **Q** and **R** matrices separately, and the solution for model vector **m** can be found as: $\mathbf{m} = \mathbf{R}^{-1}\mathbf{Q}^{-1}\mathbf{d}$.

4.5 Implementation

The prototype refraction statics inversion using the SIRT algorithm was implemented in Matlab (Figure 4.1 and Appendix A). The code was subdivided into a number of smaller subroutines which were developed and tested separately. The main subroutine was called `Inversion` and was the central part of the program (Appendix A).

In the initialization of the code, the (x, y) coordinates of the sources and receivers, and the corresponding first-break time values are read in one by one. The initial model is obtained from the Herglotz-Wiechert transform and read in through subroutine called `init_model`. The ray tracing subroutine is called `trace_ray`, and it uses the subroutine called `descend_all`, whose purpose is in tracing a ray that bottoms at a specified refracting boundary Subroutine `descend_all` takes the source and receiver coordinates together with the slowness of each layer and traces one ray from the surface to the bottommost layer by using a subroutine called `descend`. For any source-receiver pair, refractions along each of the model boundaries are tried, and the ray with the smallest time is selected (Appendix A). Subroutine `descend` traces the ray through a single layer and uses locating of the ray intersection with the refractor (section 3.3), which is found by a subroutine called `bisection`. Subroutine `bisection` estimates the intersection point of the ray with the base of the layer, and it can be replaced by subroutine implementing other ray intersection methods described in section 3.3.

Once the minimum time is calculated for a source-receiver pair, the program stores the contribution of that particular source-receiver pair into the $diag(\mathbf{L}^T\mathbf{L})$ and $\mathbf{L}^T\mathbf{d}$ vectors, after which the next available source-receiver pair is considered. After all of the input source-receiver pairs are modelled, the $diag(\mathbf{L}^T\mathbf{L})$ and $\mathbf{L}^T\mathbf{d}$ are complete, and the resulting residual vector $\mathbf{r}$ is calculated and projected back into the model space. The model vector is updated by using equation (4.11), and the ray-tracing iteration loop is started again (Figure 4.1).

Such iterations are performed until code reaches the stopping criteria. The stopping criterion I used was the minimal step size of the model update. When this step size becomes too small to have a significant effect the model, the code outputs the model depths and terminates.

The performance of the inversion code mainly depends on the ray-tracing technique used. The code using bisection ray-tracing for three-layer case, takes about an hour for single iteration. The performance of this code could be greatly improved by using a compiled computer language, such as C/C++. However this was not the goal of this study, and I focused on studying the properties of the inversion scheme and the resulting solution

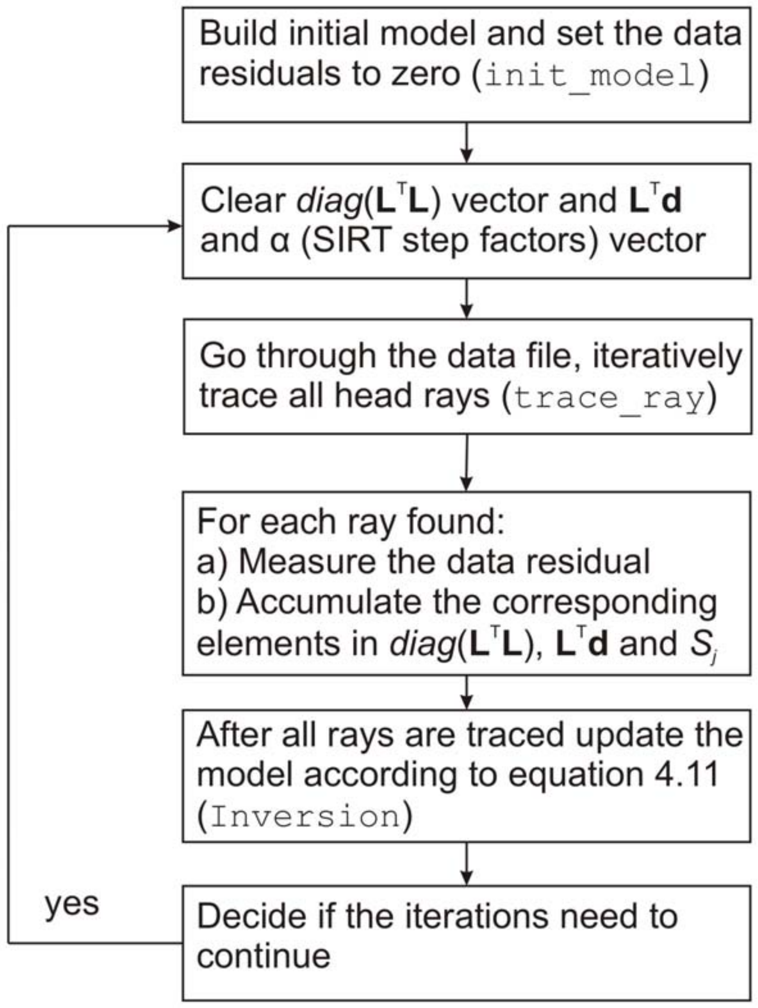

Build initial model and set the data residuals to zero (init_model)
Clear $diag(\mathbf{L}^T\mathbf{L})$ vector and $\mathbf{L}^T\mathbf{d}$ and α (SIRT step factors) vector
Go through the data file, iteratively trace all head rays (trace_ray)
For each ray found: a) Measure the data residual b) Accumulate the corresponding elements in $diag(\mathbf{L}^T\mathbf{L})$, $\mathbf{L}^T\mathbf{d}$ and S_j
After all rays are traced update the model according to equation 4.11 (Inversion)
Decide if the iterations need to continue
yes

Figure 4.1: Flowchart of the inversion program.

5. Results

In this Chapter, I apply the ray-tracing and inversion algorithm to the inversion for surface-consistent statics in the study area (Figure 1.8). First-break travel-time data are inverted for a depth-velocity model, from which the surface consistent statics are measured and applied to the data.

By examining the first-arrival travel-times in the Beaver Ranch dataset (Figure 1.8), I determined that a three-layer model was sufficient to approximating all the travel-time curves in a *τ-p* form. The velocities of three layers were 0.667 km/sec, 1.5 km/sec, and 2.0 km/sec, and the velocity of the medium below the model was 3.0 km/sec. These velocities were fixed, and only the depths of the three refracting interfaces were varied during the inversion.

Before applying the inversion to real data, it is important to examine the ability of the algorithm to invert for the various types of detail of the model. Such testing is known as model resolution analysis (Menke, 1984) and is the basis for selecting the optimal inversion grid size. I studied the properties of the ray coverage and their effects on the quality of the inverse by using several synthetics tests. In addition to the three-layer model above, I started these tests with a simple one-layer inverse problem, and examined its resolution.

5.1 Model resolution analysis

In model resolution analysis, only the geometrical properties of the source-receiver distribution (but not the travel times) are used, leading to estimates of optimal inversion grid sizes. Resolution analysis also provides understanding of the areas and features of the model which are better or poorer constrained by the available travel-time data.

Model resolution analysis is commonly performed by using two techniques. First, if the model can be derived from the data by some form of formal generalized linear inverse $\mathbf{m} = \mathbf{L}_g^{-1}\mathbf{d}$ (equation 4.2), then we can use it to invert the synthetic data predicted in some trial model $\mathbf{m}_0$: $\mathbf{d} = \mathbf{L}\mathbf{m}_0$. Therefore, the inverted model becomes linearly related to the input model:

$$\mathbf{m} = \mathbf{R}_m \mathbf{m}_0 , \tag{5.1}$$

where the model resolution matrix $\mathbf{R}_m$ is:

$$\mathbf{R}_m = \mathbf{L}_g^{-1}\mathbf{L} . \tag{5.2}$$

For perturbation of any model node #*i*, the *i*-th column of matrix $\mathbf{R}_m$ shows how this perturbation is recovered by the forward modeling and inversion. Similarly, for perturbation of model node #*i*, the *i*-th row of matrix $\mathbf{R}_\mathrm{m}$ can be interpreted as averaging vector centered at model parameter m_i (Menke, 1984) and shows how this perturbation is affected by neighbouring model parameters. In a perfectly resolved model, the resolution matrix should equal the unit matrix. Note that for the undamped Least Squares inversion, the resolution is perfect: $\mathbf{R}_\mathrm{m} = (\mathbf{L}^\mathrm{T}\mathbf{L})^{-1}\mathbf{L}^\mathrm{T}\mathbf{L} = \mathbf{I}$.

In our case, computation of the full matrices $\mathbf{L}$, $\mathbf{L}_g$, and therefore, $\mathbf{R}_m$, is impractical. However, perturbation of a selected model node can still be performed in model $\mathbf{m}_0$, and synthetic data generated and inverted as in eq. (4.11). This type of testing is performed in Section 5.1. Such tests provide great amount of detail showing interaction between the different parts of the model during the forward modeling and inversion.

Another type of resolution test often employed in tomography (Humphreys and Clayton, 1988) is the so-called "checkerboard" test. In this method, a regular alternating spatial pattern is generated in the model, and the inversion is tested for its ability to recover this pattern. The advantage of this approach is in its ability to examine the entire model at once, although with less detail of the trade-off between the different model nodes. Checkerboard testing is an effective way to measure resolution, and its results are immediately interpretable. Such testing is performed in Section 5.2.

All of the tests described below used either the 3-layer initial model with interfaces at 200-m, 400-m, and 600- m depths or a simple one-layer model with variable interface depth. The tested inversion grid sizes were 67, 134, 201, and 335 m in both the *x* and *y* dimensions (Figure 5.1). These grid sizes are multiples of the nominal receivers spacing in the Beaver Ranch seismic dataset, which was set equal 67 m. The number of the first-break travel-time data points was 169,667 in all tests, and corresponded to the actual source-receiver coverage in the selected data subset.

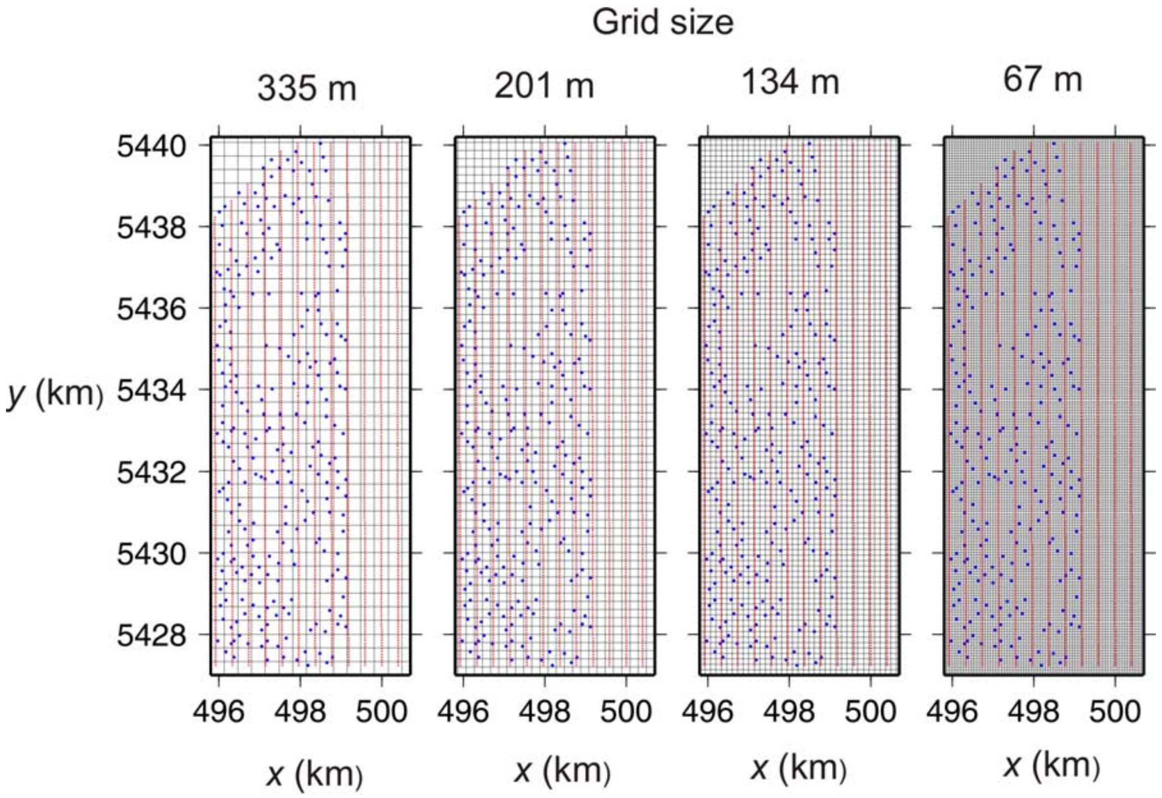


Figure 5.1: Tested inversion grid sizes, shown in combination with the source-receiver layouts (Figure 1.8).

5.1.1 Perturbation test

In order to study the sensitivity of the inversion to perturbation of a single model node, a one-layer model was used first. The initial thickness of the layer was chosen 600 m. Then I increased the depth at a single node of the model by 10% (i.e., to 660 m). Ray tracing was further performed to predict the travel times for all source-receiver pairs within this model. The output of this forward modeling then replaced the observed travel-time data and was used as an input in the inversion algorithm. The grid size for the inversion was chosen 335 m and the starting model had depth a uniform depth of value 600 m. Starting from this model, the model was updated iteratively and arrived at the model with the recovered perturbation close to the desired 10% (Figure 5.2).

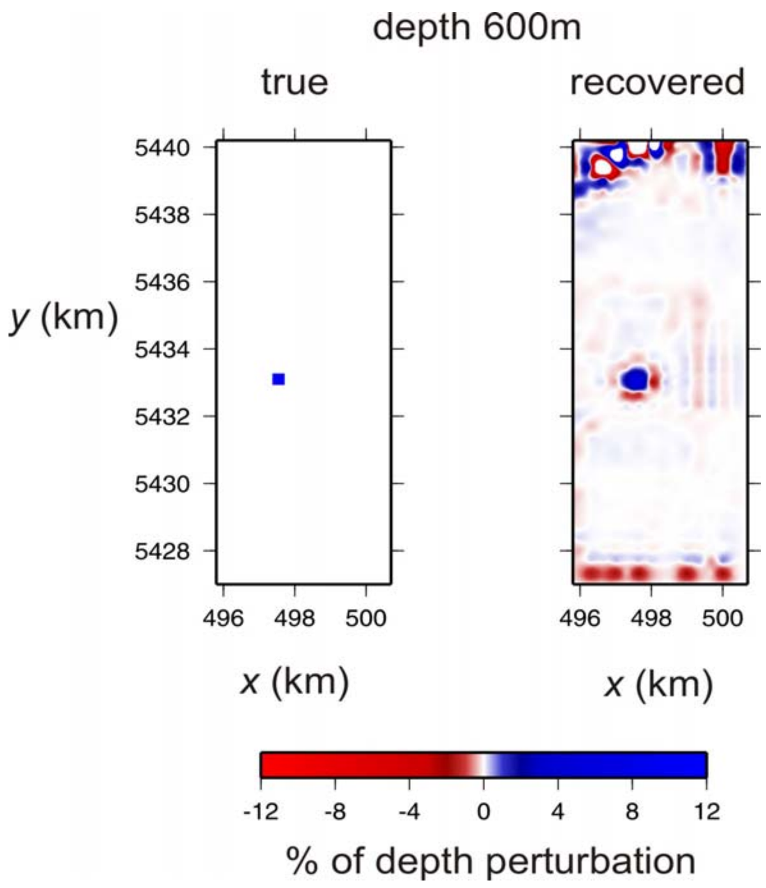


Figure 5.2: Perturbed model (left) and inverted from it (right) using a 335-m inversion grid. Note the side-lobes in the recovered model.

The recovered model reproduced the position of the perturbation accurately (Figure 5.2), but it was also smeared, and a pattern of side-lobes surrounding it could also be seen. These side lobes correspond to non-zero non-diagonal values in the resolution matrix. Also, very large spurious perturbations were recovered along the edges of the grid, where the model is poorly constrained by the data points. In Figure 5.2, the side lobes were enhanced by choosing an appropriate colour palette.

A similar resolution test was performed in a three-layer model with interface depths of 200, 400, 600 m (Figure5.3). As seen in Figure 5.3, a perturbation in layer #2 also caused a slight perturbation in the other two layers.

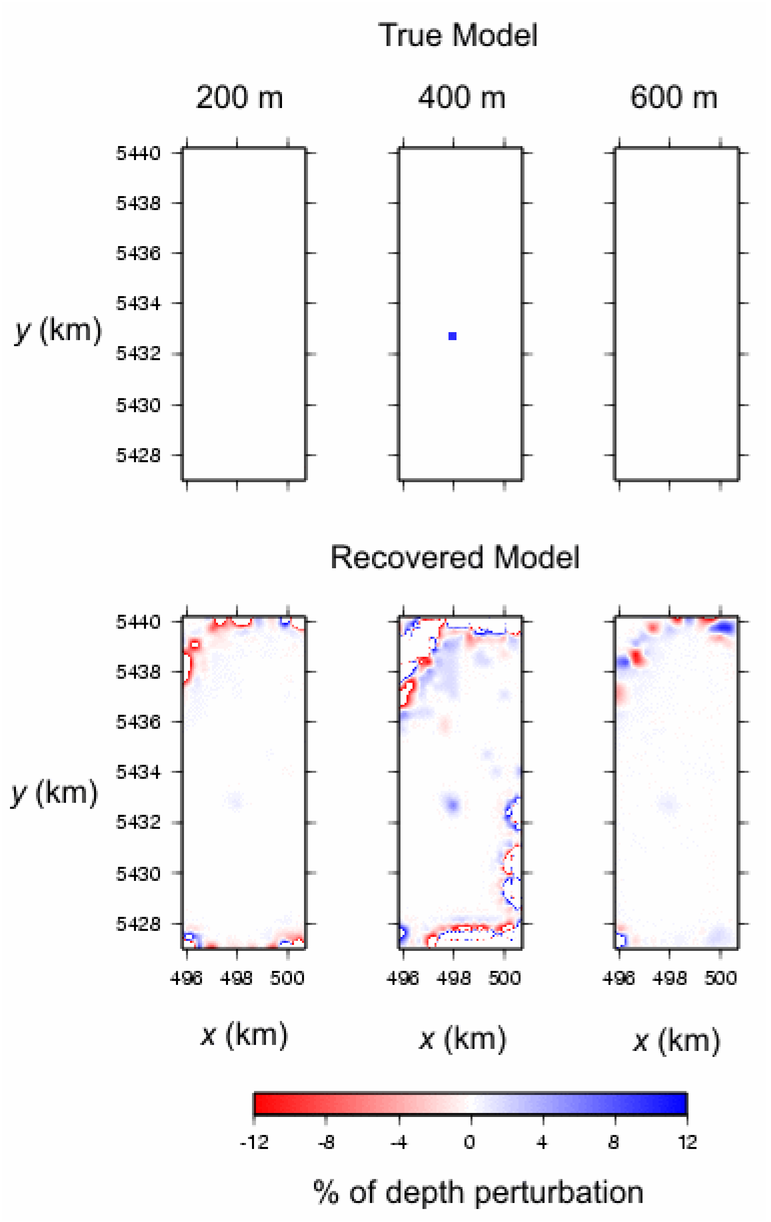


Figure 5.3: Perturbation test in a three-layer model with 200-m, 400-m and 600-m interface depths. The second interface is perturbed at a single node in the middle. The recovered model shows ghosts of this perturbation in layers #1 and #3.

5.1.2 Checkerboard resolution tests

Checkerboard tests were performed by constructing alternating positive and negative interface depth perturbations, predicting the travel-times, and inverting them (Figure 5.4). The checkerboard models were obtained by varying the depth by alternating the 10% perturbations of the model depths within the model.

The first checkerboard test was performed in a one-layer, 600-m depth earth model. The best-resolved models appear to be somewhere between 134-m to 201-m grid sizes. The 67-m grid is unable to recover the checkerboard model. The 335-m grid size seems to have recovered the checkerboard pattern well. A close-up view of the checkerboard test results is shown in Figure 5.5.

Another test was performed for the one-layer earth model by using grid size of 201 and 335 m but placed at shallower depth of 200 m (Figure 5.6). The model was generated by using a sine function with a period of about 2 km. Note that although the 201-m grid has correctly predicted the checkerboard pattern, it has also produced some spurious linear features along it (Figure 5.6). These linear features are aligned along the receiver lines. However, at the same depth of 200 m, the 335-m grid size leads to almost no linear features in the recovered model (Figure 5.6). Hence, although the resolution of the dataset could be better than 201 m, we do not use these grid sizes because of the footprint of the receiver spread. Grid size of 335 m seems appropriate for recovering the subsurface features at these depths.

Finally, in the one-layer case, the convergence speed of SIRT algorithm was measured in the first (one-layer) checkerboard test with 335-m inversion grid. The error in the inversion was calculated by using the root-mean square error of the travel-times (eq. 4.5a). The error reduction was measured in inverting for the checkerboard model with grid size of 335 m (Figure 5.7). The error decreased rapidly with the number of iterations, with some oscillations starting after about 60 iterations.

For the three-layer earth model, the checkerboard test was done by using only a single grid size of 335 m. All three layers were perturbed by ±10% in depths and alternating in space. The result of this test is shown in Figure 5.8. Note that the 335-m grid size was sufficient for recovering the checkerboard pattern for the three-layer case. It is thus established that the grid size of 335 m should be used to perform the inversion on the real dataset of this study.

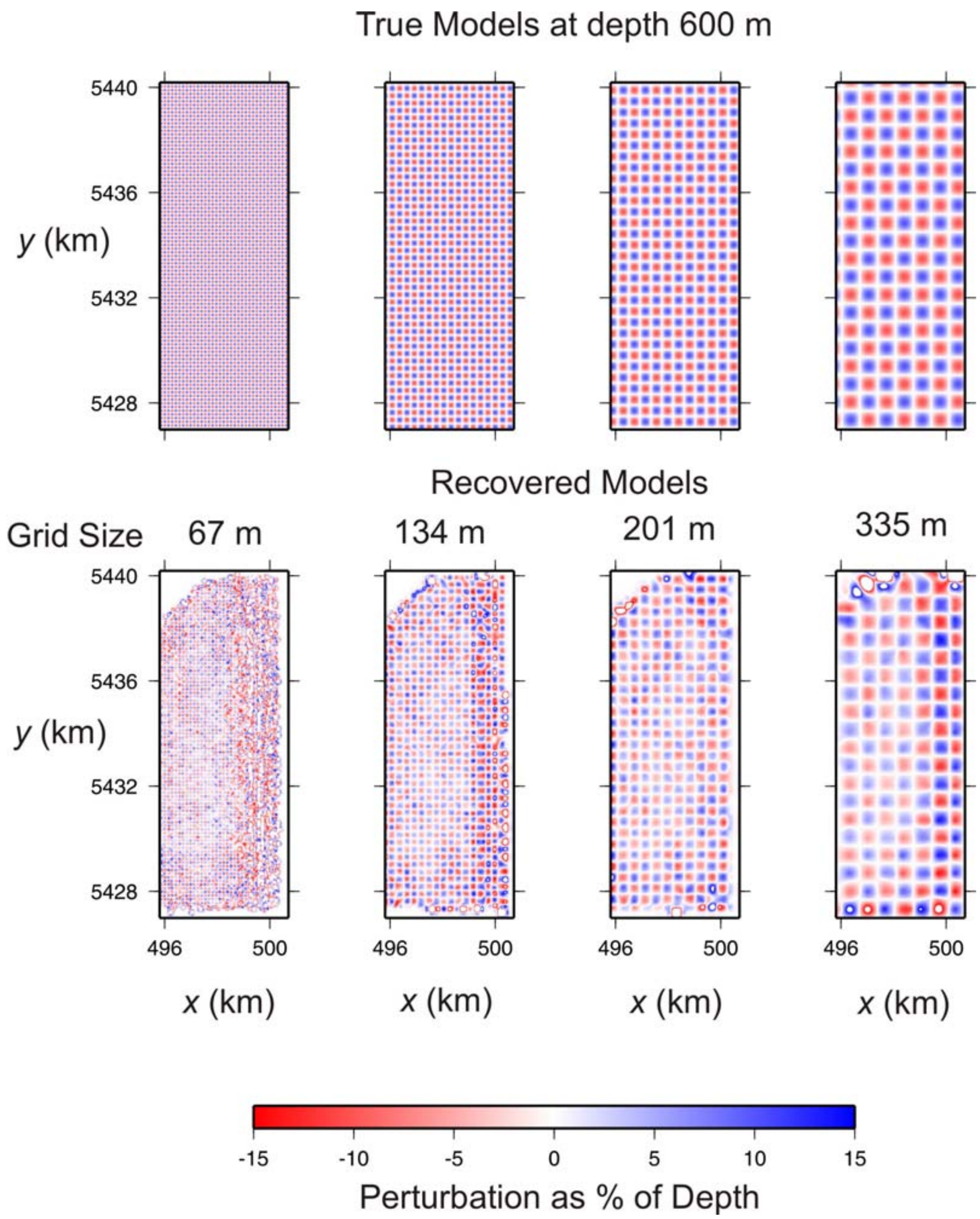


Figure 5.4: Results of single-layer checkerboard model Tests. Gird sizes used in the algorithm were 67, 134, 201, and 335 m (labelled).

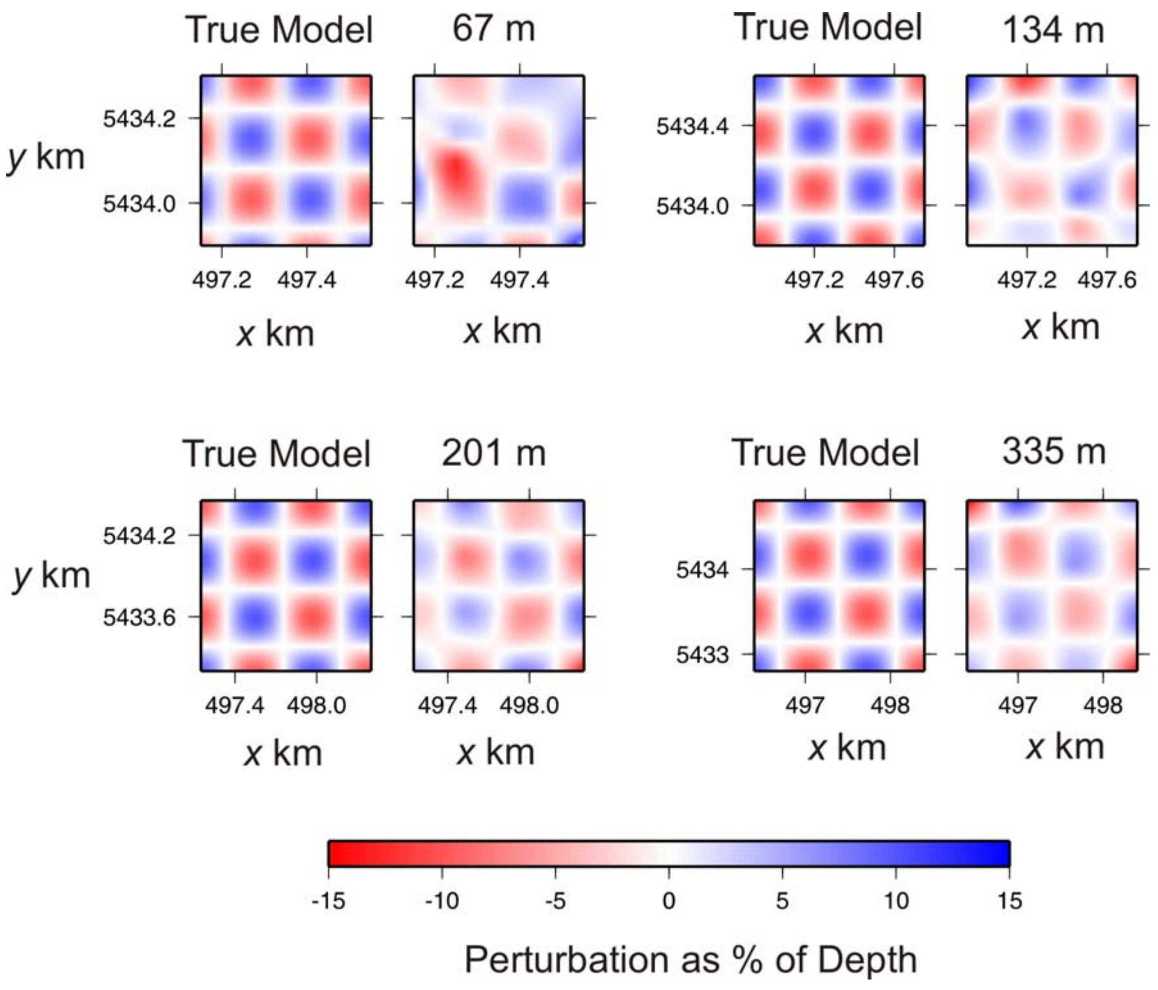


Figure 5.5: Zoom-in at checkerboard patterns in Figure 5.4. Note that patterns at 134-m and 201-m grids are recovered well.

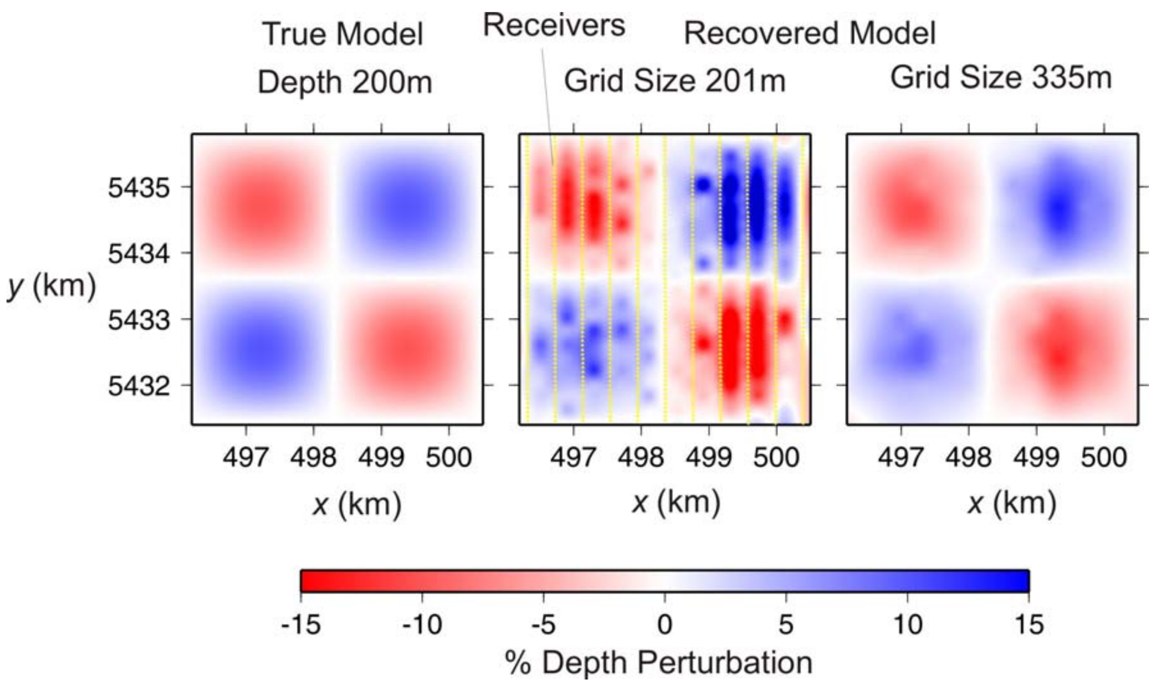


Figure 5.6: Checkerboard test using grid sizes of 201 m and 335 m. Linear features (acquisition footprint) appear on the model recovered by using grid size of 201 m.

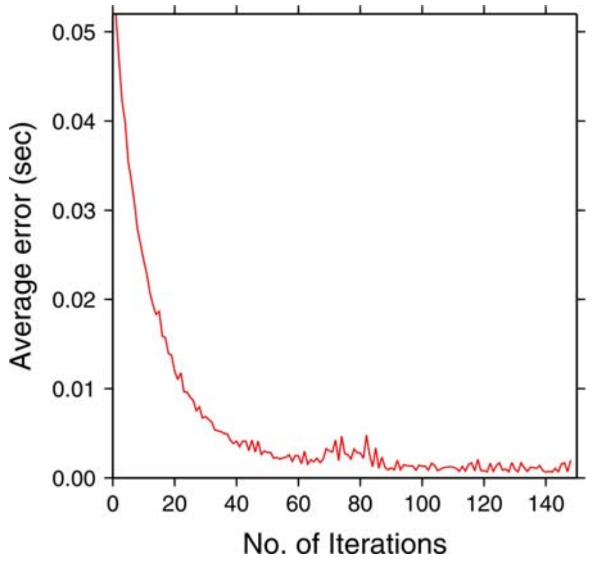


Figure 5.7: Error reduction as a function of the number of iterations during a checkerboard test.

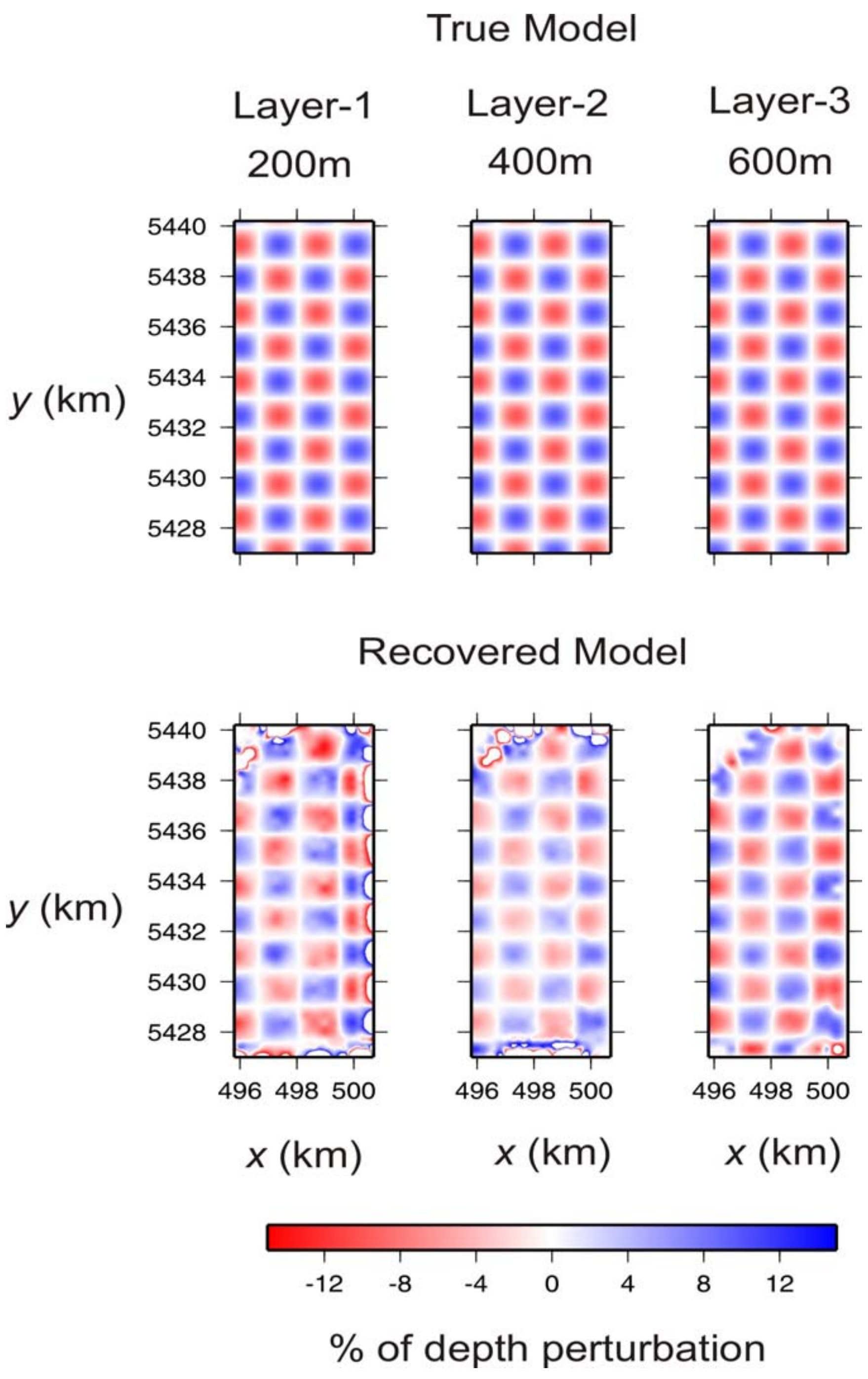


Figure 5.8: Results of three-layer checkerboard resolution test. Top: true model, bottom: recovered model. Grid size of 335 m was used.

5.2 Resulting model and surface-consistent statics

The initial model for the inversion was estimated by using IGeoS module `rsgli` (Chapter 2), which produced a three-layer earth model shown in Figure 2.4. This model was used as an input for the travel-time inversion algorithm described above. Based on the resolution analysis described above, the grid size for the inversion was chosen to be 335 m for all three layers.

The final depth model resulting from SIRT iterations is shown in Figure 5.9. In order to suppress the inversion edge-effect artefacts resulting in excessive perturbations along the northern, eastern, and southern edges of the models (as in Figure 5.2), the solution was smoothened in these areas.

All three layers in the resulting depth model show significant structures, with a greater amount of detail recovered at the shallower depths (Figure 5.9). These variations should be generally related to the variations of the subsurface. However, it is also important to realize that because of the inherent ambiguity of first-arrival inversion in general, details of this structure in depth may not be accurately related to the physical parameters, such as velocities and depths of the discontinuities. In particular, low-velocity and thin high-velocity layers can be introduced without altering the first-arrival travel times (Telford et al., 1990). The result of this inversion (as well as of all similar first-arrival tomographic approaches) represents the "optimal" solution without low-velocity zones, similarly to the Herglotz-Wiechert transform (eq. 2.9). With increasing number of layers considered, this solution can also be viewed as the smoothest vertically.

In addition to the uncertainty of the solution, note that the depths along the three out of four edges of the model are not well resolved (Figure 5.9). The depths take spurious values because these areas are not covered by the rays, as the resolution tests also showed (Figures 5.4 and 5.8). Also, as expected, in the deeper layers, the pre-critical region is broader than for shallower layer, and therefore the corresponding band of unconstrained model nodes is broader. The topographic map of the region is shown in Figure 5.10. The topography is closely related to the modeled layer #1 in Figure 5.9.

However, despite the potential uncertainties present in the inverted depth model, these uncertainties of under-constrained inversion have significantly smaller effects on the predicted data, i.e., on the travel times within the model (Menke, 1984). Therefore, the model should describe the travel-time statics well, which is the ultimate goal of my inversion. Surface-consistent statics were calculated in the resulting model by using equation 1.2 (Figure 5.11a). The static values range from about -60 to -110 ms, and the variations correlate to the surface topography and shallow subsurface structure (Figure 5.10).

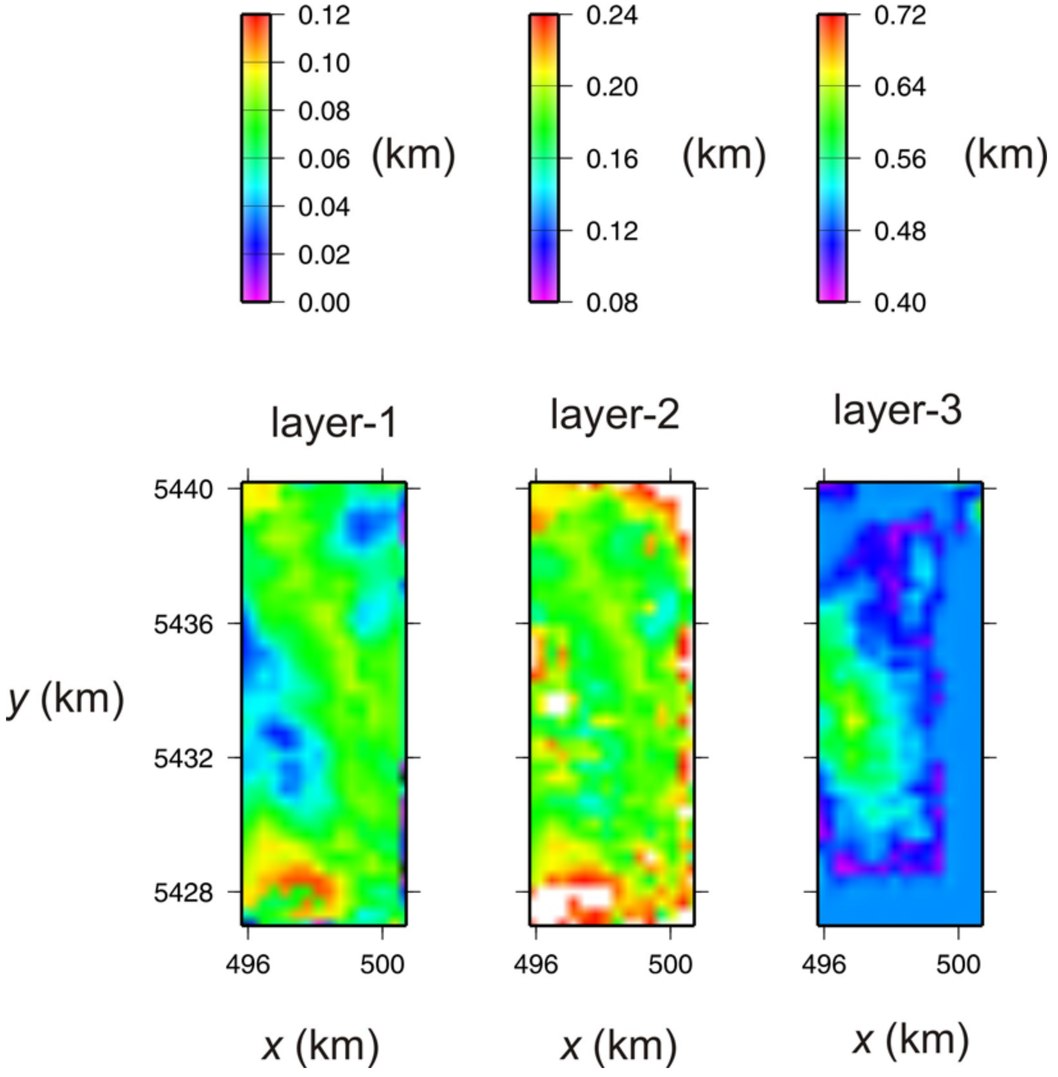


Figure 5.9: Depth to three model interfaces obtained after the inversion.

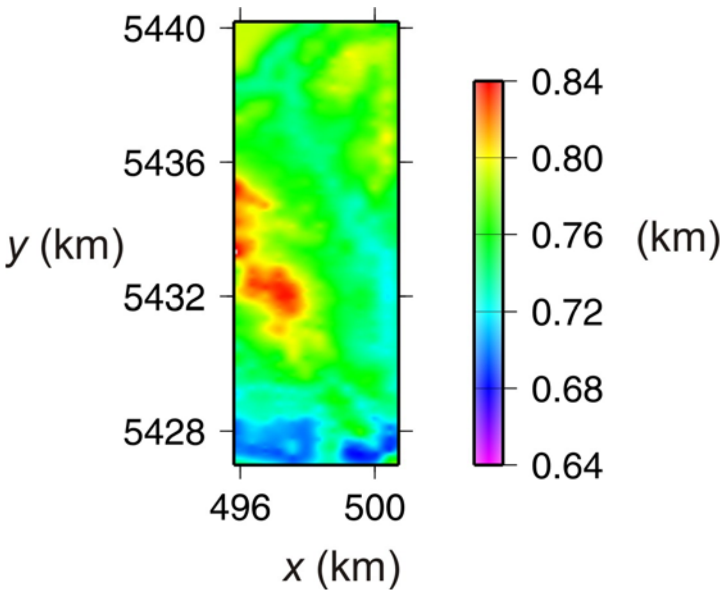


Figure 5.10: Elevations of the source-receiver locations.

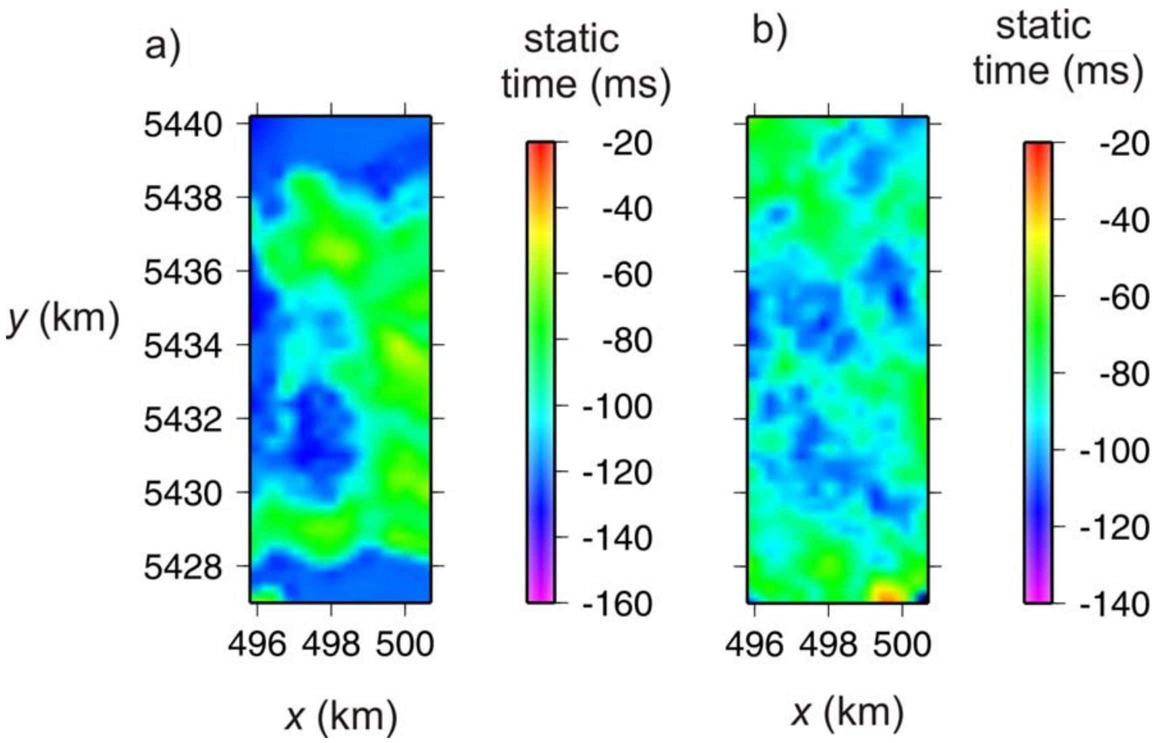


Figure 5.11: a) Predicted surface-consistent statics derived from the depth model (Figure 5.9) and b) statics derived by using Hampson-Russell software.

Further, I compared the derived surface-consistent statics model (Figure 5.11a) was to the statics obtained from Hampson-Russell software (Figure 5.11b). Both solutions show similar range of values and similar structural patterns, particularly in the middle of the model where the ray coverage is the best (Figure 5.11). As expected, both solutions correlate with surface topography, because topography represents the strongest contributions to statics (Chapter 1). My solution appears to correlate with topography somewhat stronger, likely because of the thinner layer 1 and thicker layer 3 beneath the area of increased elevation near the western edge of the area (Figures 5.9 and 5.10). However, along the under-constrained edges of the model, the solutions differ significantly. Further comparison of the two solutions is difficult, because of the differences in the algorithms and particularly in regularizing the inversion near the edges of data coverage.

Finally, the statics shown in Figure 5.11 (a) were applied to the reflection seismic dataset. Figure 5.12 shows a sample reflection shot line in the middle of the spread, in which the data coverage was good and the resulting statics solution is of high quality. Application of static corrections removes irregularities from the reflection events (Figure 5.12a) and makes them nearly hyperbolic (Figure 5.12b). Smaller variations of travel times still remaining in the records can be further removed by non-surface consistent and residual (waveform cross-correlation based) statics corrections which are not considered in this thesis. However, closer to the edges, there are other areas where the static solution was less successful. Broader data coverage and potentially a more sophisticated regularization algorithm is required in order to overcome these problems.

Comparison of surface consistent statics obtained using our program (Figure 5.12b) was made with standard industry software (Figure 5.12c). The statics obtained using Hampson-Russell is more successful in removing the near surface irregularities. This may be due to the difference in the inversion grid size. In our case the inversion grid size is constrained by model parameters with emphasis to obtain model which close to the reality, for ex. topography of the surface.

A 8.41-km long (Figure 1.8) portion of the stacked section was obtained in the middle of the area of data coverage where the surface-consistent statics worked well (Figure 5.13). As this Figure shows, the coherency and continuity of stacked events in raw stacked section (Figure 5.13a) is improved dramatically by application of the surface-consistent refraction statics correction (Figure 5.13b). For comparison, the same stacked section obtained by using the Hampson-Russell (H-R) statics and following same processing sequence is shown in Figure 5.13c. In both cases, statics lead to a substantial improvement in the quality of the stack. This comparison of the two statics methods shows the effects of the different algorithms and model parameterizations on the quality of the result.

Notably, the H-R statics appear to be somewhat better in the shot gathers (Figure 5.12). This could probably be explained by effectively finer gridding employed by the H-R model. The finer scale of H-R model gridding can also be seen in Figure 5.11b. It is known that if larger number of parameters is used in the inversion, the model may become less well constrained and the inversion - less stable, yet the data (travel time) fit would typically improve (Menke, 1984). However, my primary objective in this study was inversion for a best-resolved subsurface velocity structure. Interestingly, in the application to the stack, my solution appears somewhat better than the H-R one (Figure 5.13). This observation could again be interpreted as caused by the relative roughness of the H-R model, which could be slightly less coherent between the different shots.

In summary of these comparisons, I conclude that my model-based travel-time inversion approach works quite similarly to the industry-standard H-R GLI3D program and produces usable, and potentially even better-quality statics. Because of the additional tools for data quality control, starting model generation, and resolution analysis, this approach should provide at least a useful complement to the industry refraction statics software.

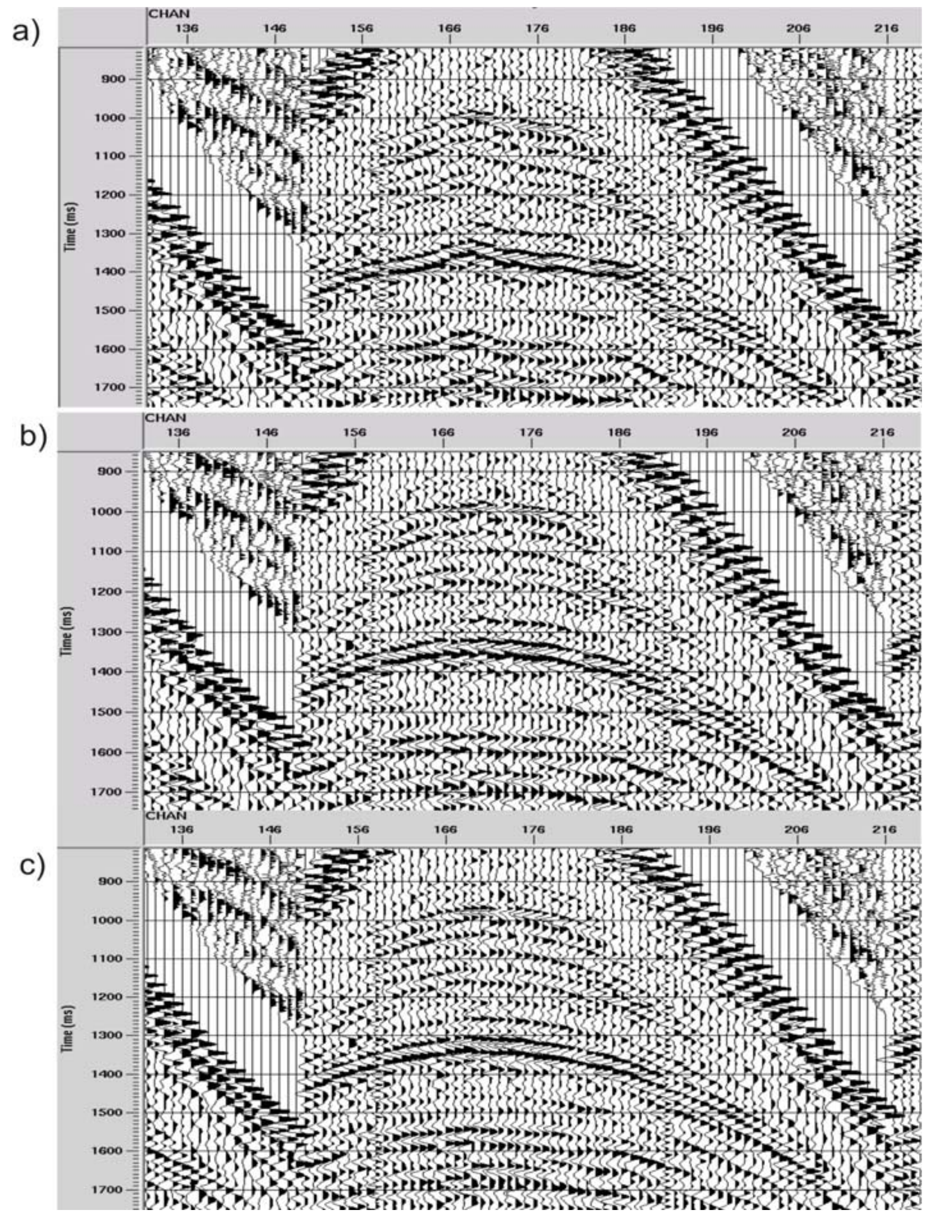


Figure 5.12: Seismic reflection shot gather: a) before application of statics, b) after application of my surface-consistent statics, c) after application of Hampson-Russell statics.

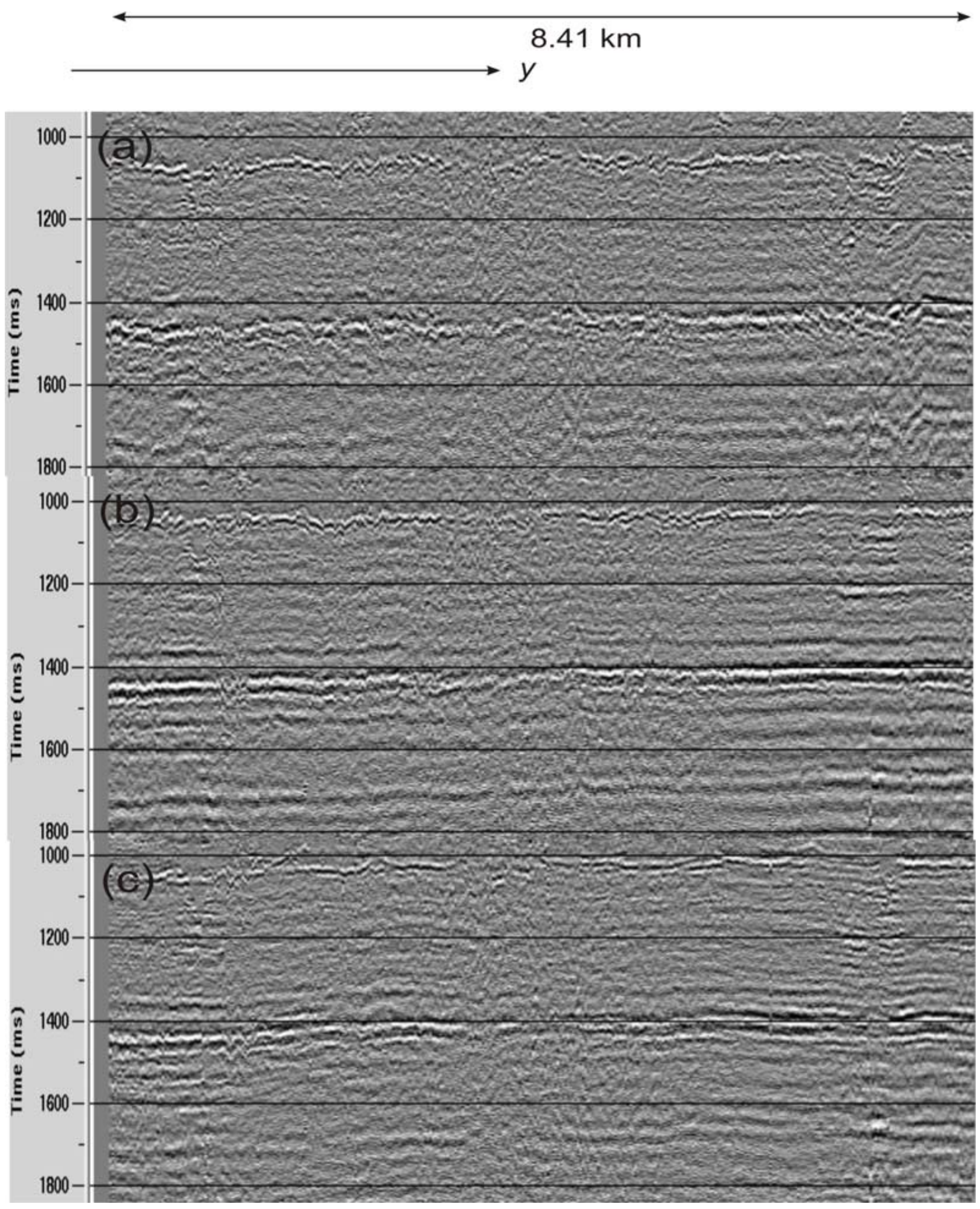


Figure 5.13: 8.41-km long segment of stacked section location (indicated by red line in Figure 1.8): (a) without statics; (b) with my surface-consistent statics applied; and (c) with Hampson-Russell statics applied.

6. Conclusions and Suggestions for Further Research

In this Thesis, the refraction statics problem was studied in 3D and in application to several synthetic and a real dataset. The two principal contributions of this study are in A) recognition of the importance of travel-time data quality control (QC) prior to inversion and the use of new techniques for this QC, and B) development, testing, and application of a new inversion technique for a refraction-statics model. The specific results in these two areas are given below.

Travel-time QC: The redundancy of travel-time data in 3D seismics can be exploited to check for errors in geometry and first-break picking. Various QC tools can be used to verify the accuracy of the travel-time data before they enter travel-time inversion. Integration in a seismic data processing system and the use of 3D visualization improves the versatility of the approach. Pre-inversion travel-time QC includes the following components:

1) Treatment of the first-arrival travel times as surfaces in the (x, y, *time*) space allows obtaining numerous constraints on the consistency of the travel-time data that help in manual, and potentially in automatic picking;

2) Travel-time reciprocity can be used to check for any errors in geometry and first-break picking. Travel-time reciprocity check is a powerful way for identifying and correcting travel-time picking and geometry pattern errors;

3) Statistics of the reciprocal travel-time misfits were measured and used for estimating the average data error (~10 ms in this dataset) and identifying outliers;

4) τ-p parameterization is convenient for describing the time-distance dependences of the travel times and leads to robust and efficient inversion.

Inversion: In this study, an innovative model parameterization using constant-velocity layers was used. This parameterization allowed a direct link to τ-p travel-time parameterization above and allowed creating an efficient inversion algorithm. Least-Squares inversion was performed by using a modified SIRT scheme. A software implementation using Matlab was created. The results of this development are:

5) Initial models were obtained directly from common-midpoint τ-*p* parameterized travel times. Such initial models are obtained completely automatically, without any model-based inversion, they closely approximate the input travel times, facilitating the subsequent tomographic inversion.

6) The subsurface models were further improved by using accurate ray-tracing and SIRT-based iterative Least-Squares inversion.

7) Surface-consistent statics were derived from the resulting model and successfully applied to the dataset. The resulting statics were also compared to the results derived by using commercial software.

8) Model resolution analysis was performed in order to evaluate the optimal grid sizes for the inversion. In particular, checkerboard resolution tests were useful for assessment of data coverage and the ability of the travel-time data to resolve the structure of the subsurface. In the dataset of this study, the best-resolved grid size was 335 m.

An overall result of this study is in demonstration that the fully 3D, τ-*p* based travel-time inversion method works, is applicable to large seismic datasets, and results in detailed shallow subsurface models and reliable statics solutions.

This study represented a part of a much broader program for creating a unified first-arrival analysis and inversion environment. In particular, the implementation of the inversion code in this thesis represented a prototype intended for testing the various aspects of the technique and different ray-tracing algorithms. In consequence, several suggestions for further improvement of the results also arise from this work:

1) More tests on real seismic datasets are needed in order to further verify the effectiveness of the algorithm and optimize its performance.

2) Because by far the time consuming component of the algorithm is the ray-tracing, an approximate but more efficient method could greatly accelerate the inversion code. However, it is likely that implementation in a compiled computer language, such as C/C++ would also improve the efficiency of the algorithm dramatically.

3) To accelerate convergence, faster-converging iterative algorithms could be used, such as, for example, the LSQR method.

4) In this research, only a crude scheme for smoothing the solution at the edges of data coverage was used. However, in order to improve the results, smoothening or another regularization of the model in the areas of poorer data coverage is required. Since travel-time errors propagate within the model, such regularization would reduce the effects of these errors on the results and also accelerate convergence.

5) Some additional advantages of the 3D *τ-p* inversion scheme described in Chapter 2 still have not been exploited in this Thesis. In particular, better accuracy and detail in utilizing the direct-wave (near offset) travel times could be obtained from the automatically-derived starting model. Recent test show that the *τ-p* inversion allows not only estimating the interface depths, but also the variable velocities in the uppermost layer. These velocities are close to the velocity of 0.667 km/s used in my inversion presented here; however, incorporation of its variability in the inversion code would likely improve its performance and quality of the solution.

7. REFERENCES

Aki, K., Richards, P.G., 1980. Quantitative Seismology, Freeman and Co.

Bessonova, E. N., V. M. Fishman, V. Z. Ryaboy, and G. A. Sitnikova (1974). The tau method for inversion of travel times, 1, Deep seismic sounding data, *Geophys. J. R. Astron. Soc*, 36, 377-398.

Chubak, G., Morozov, I., and Blyth, S. 2007. Towards a comprehensive open-source system for geophysical data processing and interpretation, *CSEG Recorder*, 32, Sept. 2007, 52-61.

Golub, G. H, and Loan, C F. V. Matrix Computations., 1996. The Johns Hopkins University Press.

Hagedoorn, J. G., 1959. The plus-minus method of interpreting seismic refraction sections, *Geophys. Prosp.*, 7, 158-182.

Hampson, D., and Russell, B., 1984, First-break interpretation using generalized inversion, *J. Can. Soc. Explor. Geophys.*, 20, 40-54.

Humphreys, E. and Clayton, R.W. (1988) Adaptation of back-projection tomography of seismic travel time problems. *J. Geophys. Res.*, 93, 1073-85.

Kaczmarz, S., 1937. Angenaherte Auflosung von Systemen linearer Gleichungen. *Bull. Acad. Polon. Sci. Lett. A.*, 35, 355-357.

Menke, W., 1984, Geophysical data analysis: Discrete inverse theory, Academic Press.

Morozov, I. B., and A. Jhajhria, 2008. 3D Refraction Statics Integrated with Surface-Consistent First-Break Picking, Iterative Inversion, and 3D Visualization, 2008 CSPG/CSEG/CWLS Convention.

Morozov I. B., V. I. Khalturin, L. N. Solodilov, E. A. Morozova, S. B. Smithson, and P. G. Richards (2005). 3-D first-arrival regional calibration model of northern Eurasia, *Bull. Seism. Soc. Am.* 95, 951-964.

Palmer, D., 1981, The generalized reciprocal method of refraction static interpretation: *Geophysics*, 36, 1508-1518.

Radon, J., 1917. Uber die Bestimmung von Funktionen durch ihre Integralwerte langs gewisser mannigfaltigkeiten, Ber. Verh. Sachs. Akad. Wiss., 69, 262-267.

Tarantola, A., 1987, Inverse problem theory. Methods for data fitting and model parameter estimation, Elsevier.

Telford, W. M. L. P. Geldart, and R. E. Sheriff, 1990. Applied Geophysics, Second Edition, Cambridge University Press.

Van der Sluis, A. and Van der Vorst, H.A. (1987) Numerical solution of large, sparse linear algebraic systems arising from tomographic problems, in Seismic Tomography, (ed. G. Nolet), Reidel, Hingham, MA, pp. 49-84.

Yilmaz, O., 2001, Seismic Data Processing: Doherty, S.M. (Ed.), Investigations in Geophysics, vol. 1, Soc. Expl. Geophys., Tulsa, OK

8. APPENDIX A: MATLAB Inversion Code

The full MATLAB algorithm is available at http://seisweb.usask.ca/students/atul. The following table lists its components and summarizes their functionalities:

`inversion.m`	Main inversion code; loads the travel-time data and performs the inversion
`init_model.m`	General model initialization
`init_tpmodel.m`	Model initialization for τ-p model or checkerboard model
`trace_ray.m`	Traces one ray through the model
`descend_all.m`	Traces one ray for 1-layer, 2-layer, and 3-layer earth model
`descend.m`	Traces one ray through single layer
`bisection.m`	Finds the intersection point of the ray with the bottom of layer
`interpolate.m`	Gives weight and cell indices for the (x,y) location based on its four corner nodes, x and y coordinates and the number of the layer
`twoindices.m`	Converts the one index notation to the two indices along x and y directions
`inv_updatecell.m`	Updates the $diag(\mathbf{L}^T\mathbf{L})$ and $\mathbf{L}^T\mathbf{d}$ and stores these values temporarily
`inv_closeray.m`	Updates and stores $diag(\mathbf{L}^T\mathbf{L})$ and $\mathbf{L}^T\mathbf{d}$ for the Inversion.
`depth.m`	Gives depth at any point (x,y) using weight and indices obtained from module `interpolate.m`
`thickness.m`	Calculates thickness at any point using `depth.m` module

Below, I give Matlab source codes of the two key modules, `inversion.m` and `descend.m` discussed in Section 4.5.

inversion.m

```
clear all;

%%% Initialize global variables accessed by most subroutines.
%%% These variables contain the layer grids and vectors accumulated
during ray tracing

global Ltd
global LtL
global model
global Ncell
global No_of_rays
global No_of_rays_r
global No_of_rays_temp

% Layer grid parameters

global Xmax
global Ymax

global Dxlayer
global Dylayer

global Nxlayer
global Nylayer

% The following vectors contain coordinates of the grid points in X and
Y directions
% needed for interpolation

global X1
global Y1

global X2
global Y2

global X3
global Y3

init_model;

global p
p = [1.5, 1/1.5, 0.5, 1/3];

%%% Ns = Number of sources
%%% Nr = Number of receivers
Ns = 255;
Nr = 1529;
Lambda = .25;
M_average = (Ns * Nr)/Ncell;

iteration = 1;
errorsum = 50;

while errorsum >= .00001
```

```
    errorsum = 0;
    i = 1;
fid_1 = fopen('beaversirt.txt');

    while 1
        tline = fgetl(fid_1);
        if ~ischar(tline),   break,   end

          x1 = str2num(tline(1,1:8));
           y1 = str2num(tline(1,11:20));

          x2 = str2num(tline(1,23:32));
           y2 = str2num(tline(1,34:44));

          tinput = str2num(tline(1,46:52));

          x3(i) = x2;
          y3(i) = y2;

          [t] = trace_ray(x1, y1, x2, y2);

          inv_closeray(tinput - t);
          error = t - tinput;
          errorsum = errorsum + error*error;
          fbtime(i) = t;

          i = i + 1;
    end
      errorsum
      epsilon = .01*.00006;
      dm = Ltd./(LtL + epsilon*ones(1,length(LtL)));

      model   =   model   +   Lambda*(M_average)   *   (dm./(No_of_rays   +
epsilon*ones(1,length(LtL))));

           fclose(fid_1);

           iteration = iteration + 1;
           fid_2 = fopen('200-400-600-z1.txt', 'wt');
           for i=1:32
               for k=1:79
                   fprintf(fid_2,  '%8.3f%12.4f%12.4f\n',  X1(i),  Y1(k),
depth_1(i,k));
               end
           end

           fid_3 = fopen('200-400-600-z2.txt', 'wt');
           for i=1:32
               for k=1:79
                   fprintf(fid_3,  '%8.3f%12.4f%12.4f\n',  X1(i),  Y1(k),
depth_2(i,k));
               end
           end

           fid_4 = fopen('200-400-600-z3.txt', 'wt');
           for i=1:32
```

```
               for k=1:79
                   fprintf(fid_4, '%8.3f%12.4f%12.4f\n', X1(i), Y1(k),
depth_3(i,k));
               end
           end
end
```

descend.m

```
% This subroutine descends the ray by one layer only.
% clayer is the current layer number
function [x1, y1, z1, ind_corner, w_weight, alpha, flag] = descend(xr,
yr, xo, yo, zr, px, py, clayer, reflayer)
ex = px/(sqrt(px^2 + py^2));
ey = py/(sqrt(px^2 + py^2));

% Find length l = distance from midpoint to point of intersection
[l, alpha] = bisection(xr, yr, xo, yo, ex, ey, clayer, reflayer);

if (l ~= 0)
    xm = (xr + xo)/2;
    ym = (yr + yo)/2;


    x1 = xm - ex * l;
    y1 = ym - ey * l;

    [w, ind] = interpolate(x1, y1, clayer);
    z1 = depth(w, ind);
    w_weight = w;
    ind_corner = ind;
    flag = 1;
end

if l == 0
    x1 = xr;
    y1 = yr;
    [w, ind] = interpolate(x1, y1, clayer);
    z1 = zr;
    w_weight = w;
    ind_corner = ind;
    flag = 0;
end
```