Travel Time Inversion in Seismic Tomography

S. Padina, D. Churchill, R. P. Bording

March 2, 2006

1 Introduction

The word *tomography* finds its origins in the two Greek words *tomos*, meaning section, and *graphy*, which translates as drawing. There are many different types of tomography used in many sciences such as medicine, biology, materials science and geology. The basic idea of tomography in all of its uses is to obtain a cross-section of an object which will then be used to infer various facts regarding the particular object's internal structure.

In processing seismic data the earth internal parameters of velocity and density play a critical role. Seismic imaging methods all require an accurate parameter estimation process. For imaging algorithms which are based on depth rather than time for the output, it is essential that the interval velocities be determined in a cellular type of model rather than the traditional layered model. The approach used in tomography is a natural for cellular seismic model building. In this paper we present a dipping layer, gridded cell, ray trace tomographic system built at Memorial University for determining reflection tomograms and well based tomograms. Based on Java it runs on both Windows and Linux machines and the code is open source which allows for modifications for research purposes. The second author is the primary developer of the code. More general systems are available but we feel the benefits of a graphical user interface and the open source code will allow for development of more sophisticated tools during the life of this project. Further, typically available codes are not written for parallel computation of the rays and the method used here is ideal for using hundreds, if not thousands of processors for the ray tracing and the linear algebra of the least squares problem. Both of these issues will be important when the seismic processing parameters are from three dimensional problems. We will illustrate the raytracing aspects of the three dimensional tomographic problem using the large screen visualization system in the Earth Sciences Department at Memorial University. As a part of the development effort we are build into the ray tracing methodology a mechanisms simulate converted waves, compute the relative amplitudes for compressional and shear waves in models that allow for acoustic, elastic, and density parameters. The research areas include allowing certain regions in the model to be anisotropic, highly fractured, or to be visco-acoustic in nature.

In the following sections we review the notion of seismic tomography, how and why travel times are computed, the forward and inverse modeling problem, how a cellular model is developed, how the raytracing works, the linear algebra of the travel time equation, solving the least squares problem using conjugate gradients, why the matrix is sparse, and some examples of ray tracing using the Java code and an example of the user interface.

1.1 Seismic Tomography

Just like radiation and magnetic fields are used in medical tomography to produce an image of the interior of the body, the same principles can be applied to imagine the interior of our planet. Seeing as sectional viewing of the subsurface of the earth is the ultimate goal of seismic surveying, it makes sense for tomography to be discussed in this context. Throughout the rest of this document, when used by itself, the term *tomography* will be taken to refer to seismic tomography and not its other applications in other fields.

Seismic tomography aids in the understanding of the planet's internal structure. However, due to the different scopes of investigations regarding said structure, most applications of seismic tomography find themselves divided between two different types:

Global In global tomography scientists apply tomographic methods to data obtained naturally from earthquakes (about 2 million of them) to understand the structure of the mantle—one of the deeper layers of earth.

Near-surface This second type of seismic tomography is concerned with the shallower subsurface, not going further than a few kilometers. Its major application is in exploration geophysics where the subsurface is investigated in a search for specific substances such as oil. As a major difference from global tomography, this method does not rely on naturally-occuring earthquakes, but rather artificial seismic-wave sources, such as explosive detonations.

It is the latter of the two types of tomography that this project is concerned with. Throughout the next sections we will further explain the basic principles of tomography along with computational problems and their solutions.

1.2 Travel Time Tomography

Travel time tomography is merely a more accurate way of referring to seismic tomography, one that puts the emphasis on the key aspect used to investigate the subsurface—*travel time*. The term refers to the time it takes a seismic wave to travel into the subsurface, reach a reflecting boundary and return to the surface. As mentioned previously, the seismic waves we are dealing with here are artificial and caused during experiments involving the detonation of certain explosives or other source mechanisms.

To retrieve the travel time, receivers are placed on the surface away from the source of the explosion at predefined distances, typically in a straight line. These receivers will record any sounds that reach them along with the time of the occurrence and by using a method called *time picking* we can obtain the exact travel times of the waves from these records. Also known as *seismic reflection times*, these travel times can be used to estimate the velocities at which the energy waves traverse the subsurface. Knowing that seismic waves are nothing more than acoustic sound waves—the travel velocity of which is known for a certain medium—we can use this information to determine the substances present in the subsurface and thus gain a better understanding of its structure. It should be at this point noted that for such a method to be successful, an elementary idea of the location of the reflecting boundaries is necessary. This issue will be mentioned again in later sections of the document.

The general procedure used in travel time tomography to determine velocities can be summarized as follows:

- In a primary step we perform the field experiment and we pick the time for rays that reach each receiver thus obtaining the seismic travel times. As a remark, it should be mentioned that between the source and any receiver there is usually more than one ray. The actual number depends on the number of reflecting boundaries.

- After retrieving the travel times we need to gain an understanding of the distances travelled to be able make any assumptions on velocities. This is where ray tracing comes to our aid. Ray tracing is a method that applies the theories of how seismic waves travel within various mediums, the result of which is a schematic that will allow us to approximate the distances travelled between reflection boundaries.
- In the concluding step we take the data produced by the previous two steps and construct so-called *travel time equations* that we then attempt to solve for velocity. This procedure is also called travel time *inversion* mainly because we start with experiment data and we infer a geophysical model that could have produced it.

In the sections to come we provide a more detailed explanation of the procedure described above.

1.3 Forward Modeling vs Inversion

In the previous section, the term travel time inversion was introduced and we provide here a section to explain the general differences between forward modeling and inversion for the reader to gain a better understanding of the opposing yet intertwining usage of the two methods in seismic tomography.

Forward modeling is a method the results of which are a prediction of the data an experiment would produce were to be performed. For this we need to have an accurate description of the geology of the subsurface we are studying. We then decide on what experiment we are concerned with and we can deduce the seismic parameters for data acquisition such an experiment would require. Based on theory and physical processes we can then use the geological model and the parameters to predict the data the the experiment would produce.

On the other hand, inversion concerns itself with how to infer a model after physically running an experiment. The known data here consists of the data acquisition parameters again and the data the experiment produced. We then apply a series of mathematic methods to obtain a description of the geology underlying the experiment.

While the two methods work at opposite ends (one's known data is the other one's aim and vice-versa) they can be combined with one working as a test for the other. This project is concerned with the inversion problem, but once we have obtained a model, we can use it as input for the forward modeling process to see if it the model would predict the data we know from our experiment to be true.

2 Previous Work

After we had a look at the basic ideas behind seismic tomography, in this section we will detail some of the work done in the past on this issue and that has enabled the inversion problem to be stated in mathematical terms. This will in turn facilitate the proposal of a solution and the discussion of its computational issues and overall feasibility.

2.1 Dividing the subsurface

In the initial stages of seismic tomography it was generally assumed that the subsurface is divided into different layers (Bording, 1987)—each with a constant velocity and density throughout the layer—and that these layers were separated by flat interfaces. While this scenario makes for very easy computations it is not in fact at all accurate. Later experiments have confirmed that velocity varies a lot within the subsurface and while there are severe differences present in some areas causing an apparent layer structure, the interfaces present between layers are not flat and one layer cannot be approximated by having constant parameters throughout. This means that the velocity with which a seismic wave would travel within a layer does not stay the same all throughout.

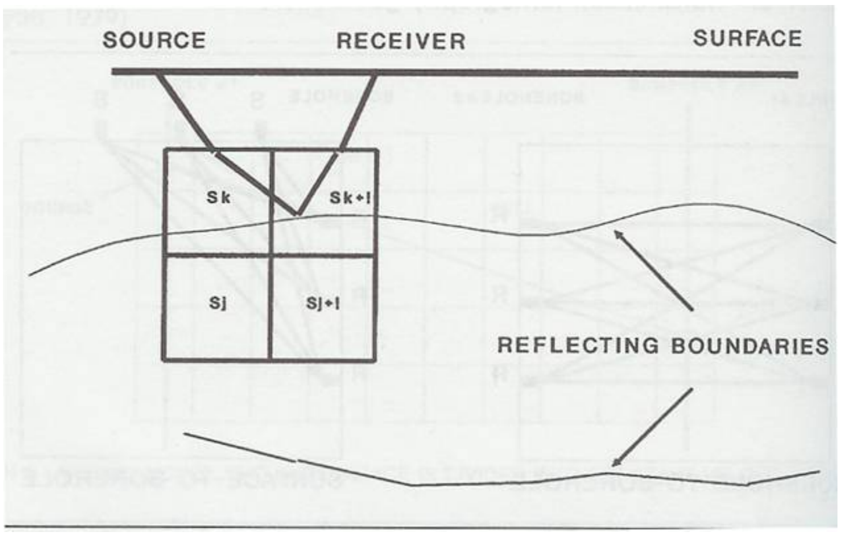


Figure 1: The division of the subsurface into cells (Gadallah, 1994)

In this context there is a need to both divide the subsurface into constant parameter units that provide for a higher resolution and to find a mathematical way to better approximate the non-flat shape of the interfaces which become our reflection boundaries. It is clear that if an interface is not flat, its shape—regardless of how oscillating it may be—can be approximated to a curve or a set of curves that connect. Mathematics provides us with a series of methods that can be used to take a set of connected curves and infer a function that will produce it. However, given the fact that we are dealing with real world structures here, their high degree of randomness renders this approach useless due to the complexity of the resulting functions which would make any computational effort unfeasible.

Consequently scientists have come forward with an idea of dividing the subsurface into square cells. The higher the number the higher the resolution and the better the accuracy. Interfaces would now be represented by a jump in velocity between adjacent cells. Figure 1 offers a better understanding of the underlying concept. In this figure we notice a seismic ray traveling from the source through two cells and reaching a receiver after reflecting of a boundary. In the picture one can notice the low resolution of the of the cells, seeing as they describe the interface rather poorly. This is a mere example; in reality cells offer a better approximation of reflection boundaries.

Please note that throughout this section we described the subsurface layers and interfaces in a two-dimensional manner. In reality of course the interfaces are curved planes and the cells are cubes given the three dimensions of our space. Nonetheless, since tomography is by definition concerned with cross-sections, it is safe to discuss the problem in two dimensions instead of three.

In Figure 2 we show the velocity and reflection surfaces of a typical but rather simple model using our JRAY system. This is used to illustrate the cellular nature of the tomographic model.

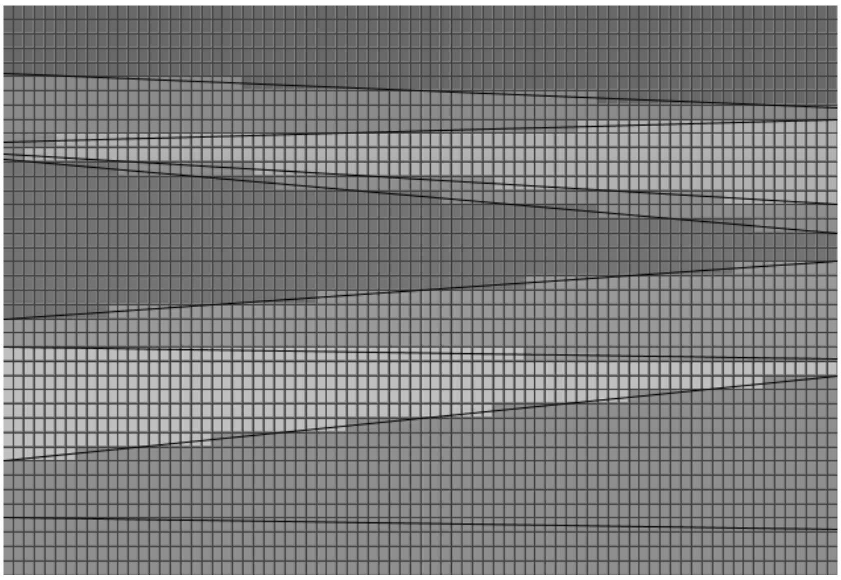


Figure 2:

By restriction to dipping layers the program is capable of tracing rays in a simple and iterative way, with few catastrophic failure modes. Methods based on local elements tend to have curved rays that wander out of the model, the layer assumption reduces these artifact rays to a minimum.

2.2 Ray Tracing

The next aspect that needs to be addressed in order to understand the seismic tomography problem is ray tracing. Looking back at the travel time procedure explained in Section 1.2, we notice that, even though the experiment will give us the travel times, we need an estimate of the distances travelled before we can calculate velocities. Ray tracing handles this part of the process and by superimposing our cell structure on the ray trace we can measure the distance each ray travels inside each cell.

Using the general approximation that energy travels from source to receiver along a ray path, ray tracing provides a set of rules that govern the way a path takes to reach a source. This includes methods of calculating the angle of reflection and also how the angle changes for the part of the ray that travels through the interface. Because there are many interfaces within small distances the rays can appear to curve the deeper they travel and the more boundaries they traverse. This can be seen in Figure 3, which represents two sources (one at point 0 and one at point 100) and the travel paths of the rays to the receivers placed at equal distances between them.

For this method to work properly we need to know where the reflection boundaries are located. Otherwise we have no way to say for sure how long the rays have travelled. This is to say we cannot use the method described by this document to deduce the structure of the earth from scratch, but rather to confirm and increase the accuracy of previous less exact findings.

In Figure 4 we show a single shot with multiple receivers, the shot is located in the upper left hand corner of the graphic, the receivers are located across the model as indicated by the triangles. Each ray that reaches the receiver was computed by an iterative process developed by

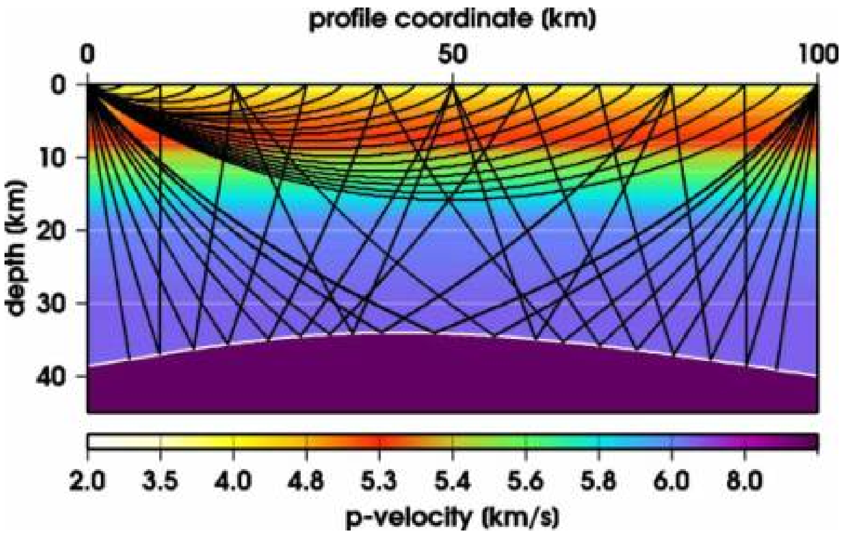


Figure 3: Travel paths of ray as they travel from two sources into the subsurface, reflect and reach several receivers

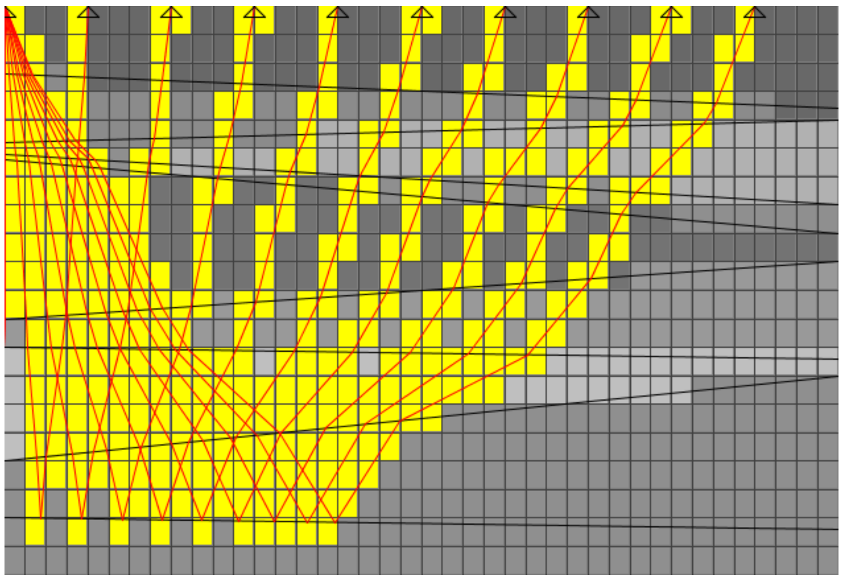


Figure 4:

the second author. As these rays are from a shot location and each shot location is independent we could implement a parallel processing method of distribution of the shots. In this way each processor would have a large collection of rays from the set of shots. As each shot would generate a part of the large sparse matrix, this would also be a useful decomposition of the work for the conjugate gradient solver. A natural parallelism for the tomographic system.

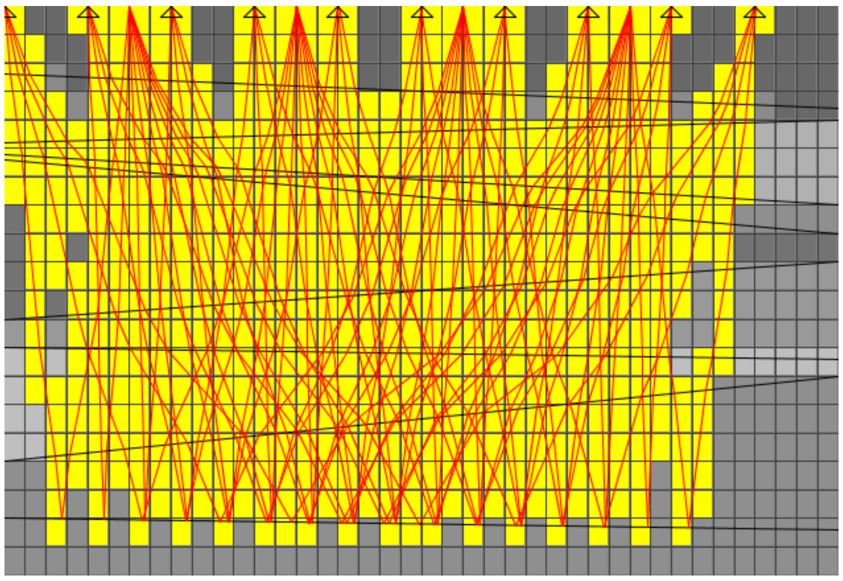


Figure 5:

In Figure 5 a typical source/receiver configuration is used with just a few of each to illustrate the ray path bending (using Snells law) and the ray coverage within the model space. In a real seismic tomogram we would cover the subsurface with rays to the point that the cells would be lost in the many rays. Hence, by using just a few we can demonstrate this process. In the tomogram we use only a few of the layers as reflector because those represent the main reflections in the real data that provided the travel time picks. The real earth causes reflections at impedence changes in the earth. Some of the reflections have significant energy, more than others, and are dominate in the seismogram. By using a few layers for the travel time picks we reduce that effort. These layers must encompass the breadth and depth of the model as illustrated in this figure.

2.3 Travel Time Equations

In order to reach the final goal of the inversion problem, to calculate the velocities of the rays within the cells based on the ray traces and the travel times, we need to describe the relationships present between the different values in a mathematical way. After the ray tracing, for every ray we will be able to write the following equation. Based on basic Newtonian mechanics we have

$$t_i = \frac{d_{i,1}}{v_1} + \frac{d_{i,2}}{v_2} + \frac{d_{i,3}}{v_3} + ... + \frac{d_{i,m}}{v_m} \tag{1}$$

where:

t_i is the time it took ray i to travel from source to receiver

$d_{i,j}$ is the distance travelled by ray i in cell j

v_j is the velocity corresponding to cell j (and constant throughout the cell)

To simplify the notation, we introduce a new unit S defining the slowness of a cell. S is defined to be the inverse of the velocity v in each cell, so we have

$$S_j = \frac{1}{v_j}$$

By replacing velocity with slowness equation (1) now becomes

$$t_i = d_{i,1}S_1 + d_{i,2}S_2 + d_{i,3}S_3 + ... + d_{i,m}S_m \tag{2}$$

Throughout the course of an experiment we deal with a very high number of rays, thus we have a multitude of equations of the form of equation (2). To present the information in a more clear fashion we can rewrite all travel time equations in the form of a matrix-vector product which will give us

$$T_n = D_{n \times m} \times S_m$$

where:

n is the number of rays

m is the number of velocity/slowness cells

T is the vector of travel times with as many rows as there are rays

D is the distance matrix with a row for every ray and a column for every cell

S is the slowness vector with as many columns as there are cells

2.4 Problem Statement

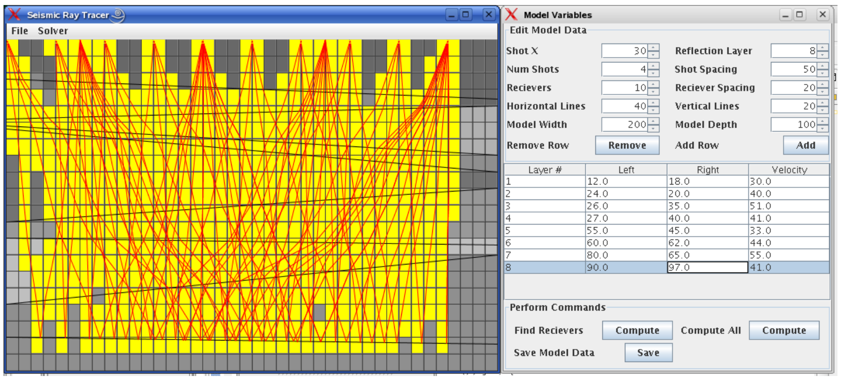
Looking at equation (4) from the previous section it might seem that knowing T—from the experiment—and D—from the model—is enough to solve for S. This is not the case as T is known to be accurate while D is an estimate obtained from ray trace which was done based mainly on assumptions. The problem can now be stated in a clear way.

Given an accurate set of travel times for the rays and estimated distance travelled within each cell—which has been calculated based on assumed slowness values—we want to calculate an accurate slowness vector.

2.5 Implementation

The graphical user interface of JRAY shown in Figure 6, is written in Java and is designed to allow the user to interact in a meaningful way, and to give a dynamic interaction with the tomographic system. The user can specify the input files, control a number of seismic parameters, the shot/receiver geometry and quantity, and in general operate the system. There are three key elements to be noted about JRAY:

Intuitive GUI The user interface provides a simple and intuitive method for manipulating all options and input values for the ray tracer. Each value is clearly labelled, and by simply saving these values, the results are automatically displayed in the ray plotting screen. Ray paths, cells, and shot/reciever points are clearly marked and visible to even first time users.



Shot X	30	Reflection Layer	8
Num Shots	4	Shot Spacing	50
Recievers	10	Reciever Spacing	20
Horizontal Lines	40	Vertical Lines	20
Model Width	200	Model Depth	100
Remove Row		Add Row	

Layer #	Left	Right	Velocity
1	12.0	18.0	30.0
2	24.0	20.0	40.0
3	26.0	35.0	51.0
4	27.0	40.0	41.0
5	55.0	45.0	33.0
6	60.0	62.0	44.0
7	80.0	65.0	55.0
8	90.0	97.0	41.0

Find Recievers
Compute All
Save Model Data

Figure 6:

Simple Output The menu driven output system enables users to output any data stored within JRAY's memory about mathematical calculations to a file. Whether it be the distance matrix, ray paths, cell velocities, travel times just to name a few. With a single click these are stored in a file of the user's choice in a clear, readable format.

Behind the scene mathematics Since all computations are done behind the scenes, the user interface is not cluttered by program output. All calculations are performed, with the display updating automatically. This makes it much easier to view results and analyze data without being distracted by calculations. If the user does with to see these calculations, they can print any of the output to file using the file menu after the calculations have been performed.

3 Solution

Although the problem can be summed up in just a few lines, the solution, as always, is slightly more complex. Before we go into the details of the algorithm and the various steps involved, we want to point out the two sets of data that we are facing—the model versus the experiment.

3.1 Experiment Data vs. Model Data

For both the real world situation and our seismic model we will have a set of three values

$$T, D, S$$

which we can write as

$$T_{real} = D_{real} \times S_{real}$$

and

$$T_{model} = D_{model} \times S_{model}$$

A quick look at what we are given will reveal that we only know T_{real}, as far as experiment data goes and D_{model} where the model is concerned. Realizing this problem which has been pointed out in the problem statement as well is key to understanding the certain assumptions that need to be made and without which we would be doomed for failure because of too many unknowns.

The major assumption we need to make here is that our ray tracing has been accurate enough in terms of the distances it has provided us with that we can consider the two distance matrices to be approximately equal. Thus we would have

$$D_{real} = D_{model} = D$$

This is however not the only assumption we need to make as we will soon see in the first step of the proposed algorithm.

3.2 Algorithm

In order to calculate the value of S_{real}, or a value close enough to it to be within an acceptable error, the following algorithm has been proposed:

1. Estimate an initial value for S_{model}. This new assumption can be based on the values used in the ray tracing or can be chosen entirely at random, although the former would help for a faster convergence.

2. We can now calculate a provisional $T_{model} = D \times S_{model}$.

3. Having a value for T_{model} and the original T_{real} we can now compute the variation in travel times given by
$$\Delta T = T_{real} - T_{model}$$

4. After having calculated the variation in travel time we can subtract
$$T_{model} = D \times S_{model}$$
from
$$T_{real} = D \times S_{real}$$
which will give us
$$T_{real} - T_{model} = D \times (S_{real} - S_{model})$$
$$\Delta T = D \times \Delta S$$

5. Solve $\Delta T = D \times \Delta S$ for ΔS and compute S_{real} from it.

6. Re-evaluate $S_{model} = S_{real}$. The $=$ in this case stands for assignment rather than its arithmetic equality.

7. Reiterate Steps 2–6 until little or no variation can be observed between S_{model} and S_{real}. We will then have reached the most accurate values for the real slowness. The acceptable error is given by ΔT as soon as its value drops under 0.001 milliseconds.

As part of this project, the above algorithm was implemented in the C programming language. The choice for C over a more convenient, mathematically-oriented programming language is due to compatibility issues with OpenGL, the ability to create programming platform-independent binary executables and a tighter integration with other applications.

3.3 The Least Square Problem

A quick scan over the algorithm presented in the previous section will reveal that the most important step, both computationally but also as the key to the entire algorithm, is Step 5 where a system of equations is solved. It should be noted at this point that in practice the rays far outnumber the cells. While we have a number of rays of the order of 10^4, the number of cells is usually of the order of 10^2. This leaves the system $\Delta T = D \times \Delta S$ in a state of being over-determined. Since there is no exact solution this becomes a least square problem.

If we try to imagine the travel path of a ray through the cells it become fairly clear that the number of cells traversed by one individual cell is very low compared to the total number. This can lead us to conclude that every row in D has a very small number of elements that are non-zero. By generalizing to the entire matrix we can conclude that D is a very sparse matrix. Practice confirms with approximately 99% of the elements in a distance matrix being zero. This situation allows us to rule out any attempt of solving the system with a QR decomposition. Instead we will employ the conjugate gradient method.

3.3.1 Conjugate Gradient

The method of conjugate gradient changes the problem of solving $Ax = b$ into an equivalent one. For this purpose we consider the function $\phi(x)$ defined as

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b$$

where $x \in R^n$ and $A \in R^{n\times n}$. The minimum value of $\phi(x)$ is $-b^T A^{-1}b/2$ which is reached at $x = A^{-1}b$. Thus the problem become finding the x for which $\phi(x)$ reaches its minimum. The easiest way to do this is to employ the *steepest descent* method.

The method of steepest descent starts from the statement that at a current point x_c the function $\phi(x)$ decreases most rapidly in the direction of the negative gradient $-\nabla\phi(x_c) = b - Ax_c$. By calculating the *residual* of x_c as $r_c = b - Ax_c$ we can pick values of x to further minimize $\phi(x)$ until the residual falls under a predetermined value. That is, if r_c is not zero, we can conclude that there exists a positive α such that $\phi(x_c + \alpha r_c) < \phi(x_c)$. When performing steepest descent we set $\alpha = r_c^T r_c / r_c^T A r_c$ thus minimizing $\phi(x_c + \alpha r_c)$. A C routine to perform this algorithm can be quickly programmed. The value we have chosen as a requirement for the residual before ending the algorithm is 10^{-3}.

As a remark we would like to point out that to apply a conjugate gradient algorithm, the matrix A is required to be positive definite. The matrix D is not positive definite but we can replace it with

$$M = D^T D$$

which is a non-negative definite matrix, albeit possibly having a zero eingenvalue.

3.4 Computational Issues

A very obvious computational issue results from a fact stated in a previous section regarding the low number of cells any given ray will traverse. Since D is very sparse and made up of many zeros, a lot of processor time can be wasted in Step 2 of the algorithm where $D \times S_{model}$ is computed. The processor will waste a flop for every zero element of D which can be catastrophic for a high number of rays and cells. To avoid this, instead of storing D it as classical matrix, we have decided to store it as tuples (d, i, j) where the d is the actual value that would reside in $D[i.j]$. We now only need to store non-zero elements.

To calculate T_{model} we now just compute every element as $T_{model}[i] = \Sigma_j dS_{model}[j]$. Although the formula might look misleading it should be remarked that d differs between tuple and the symbol d only refers to the distance value in a tuple in general and not to a specific value. To perform the calculation we initialize $T_{model} = 0$. We then process the tuples one at a time and in each iteration we add the product $dS_{model}[j]$ to the corresponding element of $T_{model}[i]$ given by the i in the tuple. The same technique can be applied in Step 5 when calculating $D^T \times \Delta T$. In this case though, by transposing, tuple (d, i, j) becomes (d, j, i).

Another computational issue that needs to be taken into account is to remember that Step 5 is a least square problem. This means that the solution we will get in the end is one possible solution but by no means the only one. Least squares problems can have several solutions that will satisfy the original system. In general the solution we do get is close enough to the real value for the purposes of seismic tomography that this does not play a major role. Nonetheless, care should be applied to ray tracing. Minor errors here could ill-condition Step 5 and produce wrong results.

4 Conclusions and Future Work

As we have pointed out earlier in this document, to be able to run the proposed algorithm and reach a correct result we need to have certain previous knowledge of where the reflection boundaries are as well as have a general idea of the velocity parameters of the cells. This can be quite limiting and will only allow the algorithm to be used in order to further refine the slowness values for each cell. However, it seems that in reality we sometimes misplace reflection interfaces and that causes us to err significantly in determining the structure of the subsurface.

Imagine we think there is a boundary about 1-ft closer to the surface than it actually is. This means that in a time t the a ray reflecting on the respective boundary will travel a distance approximately 2-ft longer than what we traced. This would seem to suggest that since more distance was covered in the same time the cells that were traversed have a lower aggregate slowness than we anticipated with our model. The algorithm will run in this case as well, and it will produce results. However, they will not be refining the value of the real slowness vector but rather an imaginary situation.

To void being dependent on previous information regarding the subsurface in the future, the mathematical treatment of the problem can be refined, perhaps even introducing a concept of probability regarding the location of a reflection boundary. This way, while running the algorithm based on a more or less accurate assumption we would be able to start confirming or disproving the placement of reflection boundaries by the effective use of probabilities. This could also allows us to introduce new boundaries that were not taken into account before but which the data seem to indicate as possible. The end result will be a much more accurate approximation of the structure of the subsurface.

5 References

- Gadallah, M.R. *Reservoir Seismology: Geophysics in Nontechnical Language.* Tulsa, Oklahoma: PennWell Books, 1994.
- Golub, G.H. & Loan, Van, C. F. *Matrix Computations.* Baltimore, Maryland: The Johns Hopkins University Press, 1996.
- Tarantola, A. *Inverse Problem Theory and Methods for Model Parameter Estimation.* Philadelphia, Pennsylvania: SIAM, 2005.