Surface-Source Downhole Seismic Analysis in R

By Eric M. Thompson

Open-File Report 2007–1124

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
Gale A. Norton, Secretary

U.S. Geological Survey
P. Patrick Leahy, Acting Director

U.S. Geological Survey, Reston, Virginia 2007

For product and ordering information:
World Wide Web: http://www.usgs.gov/pubprod
Telephone: 1-888-ASK-USGS

For more information on the USGS—the Federal source for science about the Earth, its natural and living resources, natural hazards, and the environment:
World Wide Web: http://www.usgs.gov
Telephone: 1-888-ASK-USGS

Contents

Figures

Tables

Surface-Source Downhole Seismic Analysis in R

By Eric M. Thompson[1]

Introduction

This report discusses a method for interpreting a layered slowness or velocity model from surface-source downhole seismic data originally presented by Boore (2003). I have implemented this method in the statistical computing language R (R Development Core Team, 2007), so that it is freely and easily available to researchers and practitioners that may find it useful. I originally applied an early version of these routines to seismic cone penetration test data (SCPT) to analyze the horizontal variability of shear-wave velocity within the sediments in the San Francisco Bay area (Thompson et al., 2006). A more recent version of these codes was used to analyze the influence of interface-selection and model assumptions on velocity/slowness estimates and the resulting differences in site amplification (Boore and Thompson, 2007). The R environment has many benefits for scientific and statistical computation; I have chosen R to disseminate these routines because it is versatile enough to program specialized routines, is highly interactive which aids in the analysis of data, and is freely and conveniently available to install on a wide variety of computer platforms.

These scripts are useful for the interpretation of layered velocity models from surface-source downhole seismic data such as deep boreholes and SCPT data. The inputs are the travel-time data and the offset of the source at the surface. The travel-time arrivals for the P- and S-waves must already be picked from the original data. An option in the inversion is to include estimates of the standard deviation of the travel-time picks for a weighted inversion of the velocity profile. The standard deviation of each travel-time pick is defined relative to the standard deviation of the best pick in a profile and is based on the accuracy with which the travel-time measurement could be determined from the seismogram.

The analysis of the travel-time data consists of two parts: the identification of layer-interfaces, and the inversion for the velocity of each layer. The analyst usually picks layer-interfaces by visual inspection of the travel-time data. I have also developed an algorithm that automatically finds boundaries which can save a significant amount of the time when analyzing a large number of sites. The results of the automatic routines should be reviewed to check that they are reasonable. The interactivity of these scripts allows the user to add and to remove layers quickly, thus allowing rapid feedback on how the residuals are affected by each additional parameter in the inversion. In addition, the script allows many models to be compared at the same time.

[1] eric.thompson@tufts.edu

Inversion formulation

In this section I follow the linear model formulation and notation in Faraway (2005) which has also helped in writing these scripts. Surface-source downhole-receiver seismic data with k travel-time measurements $tt = (tt_1, \ldots, tt_k)^T$ at depths $z = (z_1, \ldots, z_k)^T$ are predicted by the equation

$$tt = \frac{d_1}{v_1} + \frac{d_2}{v_2} + \ldots + \frac{d_n}{v_n} \tag{1}$$

where n is the number of layers in the profile, the velocity of each layer $v = (v_1, \ldots, v_n)^T$, and the distance traveled in each layer $d = (d_1, \ldots, d_n)^T$. Figure 1 illustrates these variables. Equation 1 can be rewritten as a summation

$$tt = \sum_{i=1}^{n} \frac{N_i h_i s_i}{\cos(\theta_i)} \tag{2}$$

where the layer thickness $h = (h_1, \ldots, h_n)^T$, the layer slowness $s = v^{-1}$, the angle of incidence of the ray path in each layer $\theta = (\theta_1, \ldots, \theta_n)^T$, and the number of transits in each layer $N = (N_1, \ldots, N_n)^T$. The array of transits is constructed as follows: $N_i = 1$ for layers above the layer containing the travel-time measurement at depth z_i; $N_i = 0$ for all layers below; and $N_i = (z - d2b_{i-1})/h_i$ for the layer that contains the travel-time measurement, where the depth to the bottom of each layer $d2b = (d2b_1, \ldots, d2b_n)^T$. The horizontal offset, H, constrains the ray path such that

$$H = \sum_{i=1}^{n} h_i \tan(\theta_i) \tag{3}$$

and the velocity model, v, constrains the ray path such that the ray parameter, p, is constant for all layers and is given by the equation

$$p = \frac{\sin(\theta_i)}{v_i} \qquad \text{for } i = 1, \ldots, n. \tag{4}$$

The ray path is a function of v, but the solution of the inversion for v is also a function of the assumed ray path. An iterative algorithm is necessary to find θ and v that are in agreement. A good initial model is the v for which all ray paths are assumed to be straight lines. The algorithm then solves for θ given this v, and inverts v again assuming the new θ. This is repeated until the maximum difference between the layers of two consecutive v is acceptably small (< 0.01 m/s).

The P- and S-wave travel-times must be manually picked from seismograms, and the confidence in each pick can be significantly different from measurement to measurement depending of the amount of noise in the signal. The standard deviation of the travel-time measurements normalized to the standard deviation of the best pick $sig = (sig_1, \ldots, sig_k)^T$, should

be recorded to document the confidence in each pick. The array of weights is defined as $w = sig^{-2}$. To estimate the *n*-parameters *s*, it is convenient to rewrite Equation 2 as

$$tt = Xs + \varepsilon \tag{5}$$

where the error in the representation $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_k)^T$ and the k by n predictor matrix X is given by

$$X_{i,j} = \sum_{j=1}^{n} \frac{N_j h_j}{\cos(\theta_j)} \qquad \text{for } i = 1, \ldots, k. \tag{6}$$

The weighted least squares solution for s is

$$\hat{s} = \left(X^T W X\right)^{-1} X^T W tt \tag{7}$$

where the k by k weighting matrix $W = diag(w)$. The standard errors of the parameters are given by

$$se(\hat{s}_i) = \hat{\sigma}^2 \sqrt{(X^T X)^{-1}_{i,i}} \tag{8}$$

and the estimated variance $\hat{\sigma}^2$ is given by

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{k} \left(tt_i - \hat{tt}_i\right)^2 w_i}{k - n} \tag{9}$$

and the travel-times predicted from the model are $\hat{tt} = X\hat{s}$.

Automatic picking of layer-interfaces

Traditionally the *d2b* are all picked manually or placed at the location of lithlogic changes in boring log descriptions. The *d2b* are usually easy to identify manually in the travel-time data because a change the 1D velocity profile will result in a change in the slope of the travel-time data. Here I present an algorithm that automatically finds layer-interfaces. The largest benefit is gained by using the automatic picking when analyzing a large number of sites. It should improve the consistency with which the downhole seismic data is interpreted, increase the efficiency of the interpretation process, reduce bias created by different analysts, and minimize over-fitting the data.

The process of picking boundaries can never be fully automated because the interpretation of all downhole seismic data requires judgment. Often there is information available to the analyst that is not reflected in the travel-time data such as boring log descriptions, and the use of the profile after it is calculated. This information can change the desired amount of complexity in the model.

An analyst should always carefully review and adjusted the resulting velocity model accordingly. Automatically picked *d2b* should be used as a tool to help guide the analyst and aid in the interpretation of a layered velocity model, but these algorithms are not a substitute for good judgment.

Given a set of travel times vs depth, I calculate the best-fit homogeneous model extending from the surface to the depth of the deepest travel-time measurement. The residual sum of squares is

$$RSS = \sum_{i=1}^{k} \left(tt_i^{obs} - tt_i^{pred} \right)^2 \tag{10}$$

where tt^{obs} is the observed travel-time, tt^{pred} is the predicted travel time. I define the residual sum of squares of the model as a function of a new interface as

$$RSS(Z^{new}) = \sum_{i=1}^{k} \left(tt_i^{obs} - tt_i^{pred} \right)^2 \Bigg|_{Z^{new}} \tag{11}$$

where the independent variable Z^{new} is the depth of an additional interface. The new interface is added at the Z^{new} that minimizes $RSS(Z^{NEW})$. A near-perfect fit to the observations can be obtained when the number of interfaces in the model approaches the number of observed travel-times. I attempt to find a parsimonious model by only accepting new interfaces if the bias-adjusted Akaike's Information Criterion (Akaike, 1974; Sugiura, 1978; Burnham and Anderson, 2002), AIC_c, of the model with the additional interface is less than the model without the additional interface. AIC_c is a relative measurement of goodness-of-fit for comparing different models. It is a function of not only the model variance, given by the maximum likelihood estimator $\hat{\sigma}^2 = RSS / k$, but also the number of observations, k, and the number of estimated regression parameters $N = n + 1$ (number of interfaces, n, plus one for σ^2), as given by the equation

$$AIC_c = k \log\left(\hat{\sigma}^2\right) + \frac{2Nk}{k - N - 1}. \tag{12}$$

The $RSS(Z^{NEW})$ function becomes more complex while the number of layers in the model increases. Therefore, the solution is strongly dependent on the initial values and bounds in the algorithm because any optimization algorithm will converge on local minima of $RSS(Z^{NEW})$. To give the algorithm an initial value the vicinity of the best Z^{new}, I set the initial value equal to the midpoint of a selected layer and set the bounds equal to the depths of the top and bottom of that layer. Therefore, the order in which layers are selected as candidates for additional interfaces is important. Initially, I tried ordering the layers by the residual sum of squares of the data within each layer, defined as

$$RSS_j^L = \sum_i \left(tt_i^{obs} - tt_i^{pred} \right)^2 \forall \left\{ i : z_i > d2b_{j-1} \cap z_i \le d2b_j \right\} \qquad \text{for } j = 1, \ldots, n. \tag{13}$$

The results were unsatisfactory because too many interfaces were added to layers with few measurements but large residuals. Therefore, I weighted the RSS^L by the number of travel-time measurements within each layer, given by

$$WRSS_j^L = \frac{n_j^L RSS_j^L}{\sum_{i=1}^{n} n_j^L} \quad \text{for } j = 1,\ldots,n \tag{14}$$

where n_j^L is the number of measurements within the boundaries of the *j*-th layer. $WRSS^L$ is more effective than the RSS^L at finding the most appropriate layers as candidates for the addition of an interface. Additional interfaces are searched for within the layers of a model in order from the largest to smallest $WRSS^L$. If the AIC_c is increased by the addition of an interface, then an additional interface is searched for within the layer with the next largest $WRSS^L$. This continues until no additional interfaces can be added to any layers. To remove the additional iterations required to find the ray path in each inversion when calculating $RSS(Z^{NEW})$, the algorithm assumes that the ray paths are straight lines from source to receiver. After Z^{new} is found by minimizing $RSS(Z^{NEW})$, the AIC_c and $WRSS^L$ are calculated from the relevant statistics of the model fit to the data that accounts for layer refractions.

Implementing RvelsIant

In this section I attempt to outline the very basic commands that are needed to implement this package in recognition of the fact that many people who could benefit from this software are not likely to be familiar with R and will not want to invest the time to become experts in new software only to use this package. The amount of time needed to become familiar enough with R to be able run these routines should be relatively small. Those familiar with the software Matlab will find that learning R is particularly easy, because the syntax is very similar.

R is an open source language and environment for statistical computing which runs on Linux platforms, Mac OSX, and Microsoft Windows. It has tremendous capabilities, and only a relatively small percentage is discussed here. It is available online at *http://www.r-project.org/* (hyperlink). While this report cannot be an exhaustive manual on how to use R, extensive well written documentation is available at the R-project web site on how to install and get started using R. There are specific lists of answers to frequently asked questions (FAQs) for running R in Mac OSX and in Microsoft Windows. The "R Data Import/Export" manual is particularly helpful for learning how to read a specific type of data file into R. The "R Reference Index" can be somewhat intimidating at over 2000 pages of text with no figures, while the document "An Introduction to R" is much easier to read and is helpful for becoming familiar with the syntax of the R language. The reference cards in the "Contributed Documentation" are also very helpful for quickly finding the functions you are looking for. In fact, it is much less important to memorize functions than to become proficient at finding the functions you need. The quality and extent of the R documentation far surpasses any commercial software that I have ever used, so becoming familiar with it is key to successfully using the software. The "R-help" mailing list is archived online in a number of

locations, and searching through these archives can be very fruitful. R is used to teach many classes in statistics, and so often course websites have material designed to introduce R to students.

In this section I outline how to install R and some examples of how to use the Rvelslant package. Commands intended to be entered at the R command line will be indicated by a different font and lines that begin with the "greater-than" symbol. Here is an example

```
> q()
```

which is the command to quit R. All functions in R have parentheses where arguments are usually placed. The function to quit does not require any arguments (although there are optional arguments), but the parentheses must be included so that R knows that you want to execute the function *q*.

Many scientists and engineers are intimidated by (or simply dislike) the thought of using the command line interface. There are many advantages to this, not the least of which is reproducibility of results. By keeping a record of each command entered at the command line, you have a record of the exact steps taken to reach your answer (or a particular graphic file). With graphical user interfaces, this is often not the case. Burns (2006) provides a thorough discussion of the advantages of using a programming language such as R to spreadsheet-style software for computations and data analyses. It is good practice to keep an R "diary" in a text file. In fact, many R users type their commands directly into a text editor, such as Emacs, and copy and paste them into the R command line. Another benefit to this is if there is a mistake in a complicated command, it can be edited rather than completely rewritten, which reduces a lot of frustration with the command line. Further, most high quality text editors, such as Emacs, provides syntax highlighting, auto-indentation, and parentheses checking which makes code much easier to read and write. Such files can act as a "cheat-sheet" for remembering the exact syntax of a command, or which commands are useful for a particular task.

Installation instructions

As with most tasks in R, there are multiple ways to install a package. The Rvelslant package can be installed from the files available on the website associated with this report. The file for Unix variants (including Mac OSX) is *Rvelslant_<version>.tar.gz* and the file needed for Microsoft Windows operating systems is *Rvelslant_<version>.zip*. Download the appropriate file to a directory on your computer, it is not necessary to expand the compressed file.

UNIX Variants

To install the package on a Unix based system at the R command prompt, type

```
> install.packages("DIR/Rvelslant_<version>.tar.gz", repos = NULL)
```

where DIR indicates the path to the directory where the ".tar.gz" file is located. The process to install a package on Mac OSX is different: from within the Mac R GUI, select the pull-down menu *Packages & Data* , and from this menu select *Package Installer*. Next, select *Local Source Package* option in the *Packages Repository* panel, the click on the *Install…* button in the *Install Location* panel. You will then be prompted to choose the file you want to install.

Microsoft Windows

To install the package on a Microsoft Windows system, select the pull down menu *Packages*, and from this menu select *Install package(s) from local .zip files*. You will then be prompted to locate the .zip file on your computer.

Loading Rvelslant (all operating systems)

Once the package is installed, the commands will be the same for all operating systems. To access the scripts, the package must be loaded with the command

```
> library(Rvelslant)
```

and the command to get help is

```
> ?Rvelslant
```

or

```
> help(Rvelslant)
```

This will display a short description of the function, its usage, the definition of each argument for the function, a description of the value returned by the function, and a series of examples for how to use the function with the data that is included in the package.

An example session

Assuming that the Rvelslant package is properly installed and loaded, we can load some example data with the command:

```
> data(tt.s)
```

now if you list the objects in your workspace:

```
> ls()
```

you should see the object “tt.s” listed. To extract an example site from the data

```
> f <- tt.s$hole.code == 293
> example <- tt.s[f, ]
> hc293 <- list(tt.slant = example$tt.slant,
>               hoffset = example$hoffset[1],
>               z = example$z,
>               sig = example$sig,
>               hole.code = 293)
```

if at anytime you want to see what something is, just type the object name, and its contents will be printed to the terminal. For example, the first object created was `f` and it may not be immediately

obvious what this is; it is a vector of logical values (TRUE or FALSE) indicating which values of the column named `hole.code` in the `tt.s` dataframe are equal to 293. Typing the name of the object in the terminal

```
> f
```

usually helps illustrate what it contains. Also, keep in mind that the "=" operator can be used in place of the "<-" for assignment if you prefer. Next, we used this logical object to extract only the rows of the dataframe with hole code 293 into a new object `example`. Then we created a list `hc293` that is properly formatted so that the function `Rvelslant` will be able to use it. Note that R uses brackets for subsetting/indexing. In this case we used a logical vector to subset, but vectors of index values can also be used within brackets.

While the above example of reading in data is good to use for becoming familiar with how to use the commands in this package, it is important to be able to read in your own data for analysis. This is typically very easy for files with a special character as a separator (such as the comma or tab). In these cases the functions `scan`, `read.table`, and `file` are all included in the R base package. Again, the "R Data Import/Export" manual is the authoritative document on this topic and is far more thorough than I can be in this document. Because data files often come in some kind of fixed-width format, as illustrated in Table 1 (each column has a width of 10 characters) I will go through an example set of commands that are able read the contents of such files. Assuming that the contents of Table 1 were in a file named "hc293.txt" in your current working directory we can use the `read.fwf` function

```
> data <- read.fwf(file = 'hc293.txt', widths = rep(10, 3), skip = 3)
```

where `widths` is a vector of column widths, and `skip` tells the function to ignore the first two lines. Also, sometimes the current working directory is not immediately apparent when working in Microsoft Windows, so the functions `getwd` and `setwd` can be helpful. There are any number of ways that the information on the first two lines can be read in. One example is

```
> header <- scan(file = 'hc293.txt', what = character(), nlines = 2)
```

where what tells the function to read in the values as characters. Now the object header is a vector with 6 values, and the important information is in the third and sixth elements. Now we can create a list that is properly formatted so that `Rvelslant` will be able to use it

```
> hc293 <- list(tt.slant = data[ , 1],
>               hoffset = as.numeric(header[6]),
>               z = data[ , 2],
>               sig = data[ , 3],
>               hole.code = header[3])
```

Once you have the `hc293` object in your workspace using either of the examples above, we can start to look at the data. To run the Rvelslant script with all the defaults, simply type

```
> mod1 <- Rvelslant(data)
```

and you will then see a graphics device appear. It is usually helpful to resize this so that you can see the plots more clearly. Also, instructions will be printed to the terminal to remind you you're your

options are (to see the instructions while running Microsoft Windows, it is necessary to uncheck the *Buffered output* option in the *Misc* pull-down menu). Within the graphics device, you will see three plots: travel-time, residuals, and the layered velocity or slowness profile with depth. The default is to begin by fitting a single layered model, extending from the surface to the depth of the last measurement. Figure 2 shows that these plots should look like. At this point you can interactively add and remove layers until you are satisfied with the model. To add a layer boundary, left-click on the travel-time or residual plots at the depth where you want the layer to be added. To remove a layer boundary, left-click near the layer to be removed on the slowness or velocity profile plot. Once you are finished adding and removing layers, right-click on any of the plots. After right-clicking in Windows, you are given the option to "stop" or "continue," so you would want to select "stop." If you do not have a second mouse button on a Mac, you can press the ESC key instead. Alternatively, holding the CTRL button while clicking the mouse is usually equivalent to a right-click. Figure 3 shows the plots if layer boundaries were located at depths of 2.7, 9, 14, 24, 32, and 89.5 meters.

Creating graphics varies slightly on different operating systems. The functions `postscript`, `pdf`, `jpeg`, `png`, and others are all effective at creating high quality graphics. In Mircosoft Windows there is also the option of selecting the *Save As* option from the *File* pull down menu. Here is an example series of commands that will create a PDF named "hc293-mod1.pdf" in the current working directory containing the travel-time, residuals, and slowness plots for the final model returned from `Rvelslant` above

```
> pdf(file = "hc293-mod1.pdf", height = 5, width = 7)
> par(mfrow = c(1, 3), las = 1, xaxs = "r", yaxs = "r",cex = 0.6,
>     mar = c(1, 4, 4, 1))
> plotmod(mod1)
> plotresid(mod1)
> slowprofile(mod1, new = FALSE)
> dev.off()
```

the final command `dev.off()` is needed to tell R that you are finished writing to the file.

There are also many ways to save the data depending on how it is to be used. The function `save` writes an R-object to a file, and can be read back into the workspace with the `load` command. This is a binary format, however, so it is not easy for someone else to view the data if they don't have R. The functions `write`, `write.table`, and `file` all can write to text files. I have also written a function, `write.mod` which quickly and easily writes all the important information in the object returned by `Rvelslant` to two text files. The first contains the observed travel-times, the predicted travel-times, and other pertinent information and has the extension ".mod." The other file is formatted as a stair-step function of the velocity model, and has the extension ".ss." This file is intended to make plotting the model easier if using other software.

```
> writemod(mod1, prefix = "hc293")
```

For further details about the different options available in the package, see the help files within R. I have also provided a manual for the Rvelslant package that reproduces the content of the help files, but this format might be preferred to the text-based format of the help files.

Acknowledgements

I thank Dave Boore for making his FORTRAN codes available, and for patiently cross-checking the results of my codes with his FORTRAN programs. He also gave me extensive feedback while writing the codes that has improved the functionality of the codes. Rob Kayen and Dave Boore provided useful reviews that have improved this report.

References Cited

Akaike, H. (1974). A new look at the statistical model identification, *IEEE Transactions on Automatic Control*, 19 (6), 716.

Boore, D. M. (2003). A compendium of P- and S-wave velocities from surface- to-borehole logging: Summary and reanalysis of previously published data and analysis of unpublished data, U.S. Geological Survey Open-File Report OFR 03-191.

Boore, D. M. and E. M. Thompson (in press). On using surface-source downhole-receiver logging to determine seismic slownesses, *Soil Dyn. Earthquake Eng*.

Burnham, K. P., and D. R. Anderson (2002). Model Selection and Multimodel Inference: a practical information-theoretic approach, 2nd edition. Springer-Verlag, New York.

Burns, P. (2006). Spreadsheet addiction, url: http://www.burns-stat.com/pages/Tutor/spreadsheet_addiction.html

Faraway, J. J. (2005). *Linear Models with R*, Chapman & Hall/CRC.

R Development Core Team (2007). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Sugiura, N. (1978). Further analysis of the data by Akaike's information criterion and the finite corrections. *Communications in Statistics, Theory and Methods* A7, 13-26.

Thompson, E. M., L. G. Baise, R. E. Kayen, (2007). Spatial correlation of shear-wave velocity in the San Francisco Bay Area sediments, *Soil Dyn. Earthquake Eng*, vol 27, pp 144-152.

Table 1: Contents of an example data file.

```
'hole.code = 293
hoffset = 2.5
tt.slant            z        sig
0.0146           2.0          1
0.0232           4.5          1
0.0368           7.0          2
0.0520           9.5          1
0.0626          12.0          1
0.0734          14.5          1
0.0786          17.0          1
0.0864          19.5          1
0.0916          22.0          1
0.1028          27.0          1
0.1084          29.5          1
0.1126          32.0          1
0.1178          34.5          1
0.1214          37.0          1
0.1252          39.5          1
0.1288          42.0          1
0.1330          44.5          1
0.1362          47.0          1
0.1416          49.5          1
0.1456          52.0          1
0.1514          54.5          1
0.1542          57.0          1
0.1578          59.5          1
0.1634          62.0          2
0.1664          64.5          1
0.1704          67.0          1
0.1752          69.5          1
0.1790          72.0          3
0.1818          74.5          3
0.1878          77.0          3
0.1932          79.5          3
0.1962          82.0          2
0.2012          84.5          1
0.2050          87.0          2
0.2082          89.5          2
```

Figure 1. Ray diagram illustrates the variables in the inversion. The model of *n*-horizontal layers with thickness is $h = (h_1, \ldots, h_n)^T$, the depth-to-bottom of each layer is $d2b = (d2b_1, \ldots, d2b_n)^T$, the slowness is $s = (s_1, \ldots, s_n)^T$, and the velocity is $v = s^{-1}$. The ray travels a distance $d = (d_1, \ldots, d_n)^T$ in each layer with an angle of incidence of $\theta = (\theta_1, \ldots, \theta_n)^T$. The travel-times $tt = (tt_1, \ldots, tt_k)^T$ are measured at depths $z = (z_1, \ldots, z_k)^T$ in the borehole.

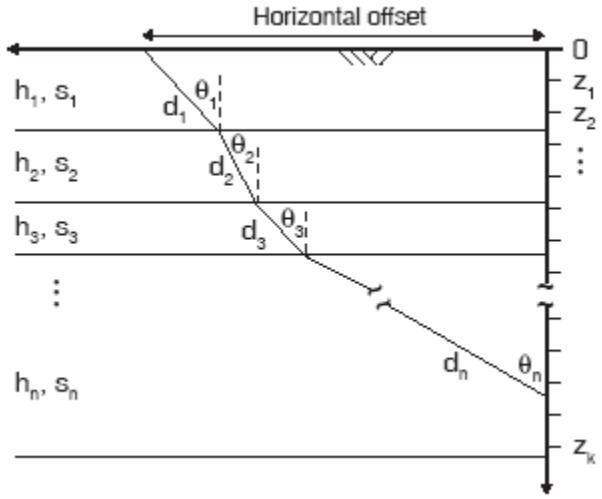

Figure 2. Example graphics output for a homogeneous model extending from the surface to the depth of the last measurement for hole code = 293.

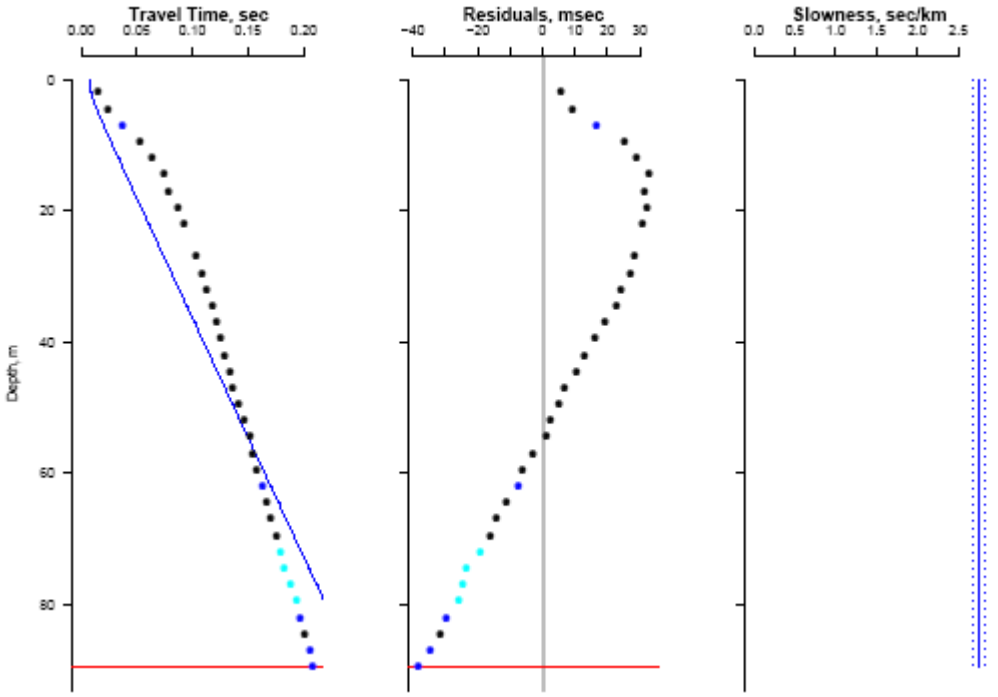

Figure 3. Example graphics output for a layered model for the surface to the depth of the last measurement for hole code = 293 with depth to layer interfaces at 2.7, 9, 14, 24, 32, and 89.5 meters.