Preliminary results of Refrapy: an open-source program for seismic refraction data analysis

Victor José Cavalcanti Bezerra Guedes[1], Marcelo Peres Rocha[1] and Susanne Taina Ramalho Maciel[2]
[1]Instituto de Geociências, Universidade de Brasília; [2]Faculdade UnB Planaltina, Universidade de Brasília

Abstract

Refrapy is a multiplatform package of open-source programs developed with Python 3.4 for seismic refraction data processing. The algorithm presented allows the processing of seismic sections, picking of first arrivals of refracted waves and inversion by least squares method to generate a model of velocities up to two layers. The objective of the work was to create an open-source program for refraction data processing. The program was applied for the processing of synthetic and real data. A well established seismic data processing software was used as reference to compare the calculated velocity models. The results presented by our open-source program were compatible with the results of the commercial package used as reference, both for real and synthetic data analysis.

Introduction

Geophysical data processing often requires the use of computer programs due to the large number of samples and mathematical routines that need to be applied. The purchase and maintenance of licenses of commercial programs generate huge investments for research institutions, such as universities. Also, many software packages are sold with several modules that are often not used, configuring a waste of resources.

Most programs that are used on geophysical data processing come from international companies. The development of programs for geophysical applications in Brazil is limited and usually not available to the general community. This project aims to develop programs for refraction seismic data processing that obtain results compatible with commercial software. Validation of the results produced by the software is made by a set of inversion calculations on synthetic and real field data.

The Refrapy package (Fig. 1), developed here, is an open-source software with a graphical interface for reading, processing, picking first breaks on SEG2 and SEGY seismic sections and computing a time-terms inversion to create a velocity model of the subsurface up to two layers. All code is written in Python. The libraries Numpy (Walt *et al*., 2011), Scipy (Jones *et al*., 2001), Matplotlib (Hunter, 2007) and Obspy (Beyreuther *et al*., 2010) are required prior to using Refrapy. The graphic interface is based on the Tkinter module, which already comes with Python installation on Windows and needs to be installed separately like the other dependencies on Unix based systems. Full source code and documentation can be obtained under the terms of the Gnu Public License (version 3 or later) from https://github.com/viictorjs/Refrapy.

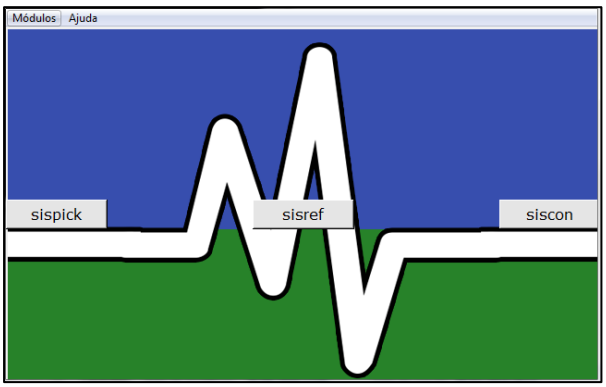

Figure 1 *– Launcher interface of the Refrapy package.*

Theory and Method

The program was written in Python 3, making use of some external libraries. Reading of SEG2 and SEGY waveform files is achieved by the reading algorithm of the Obspy framework. The plot of the read seismic sections and all other plots of the software, including amplitudes shading, relies on the extense plot functions of Matplotlib. All mathematical operations involved on the processing of the seismic traces, such as normalization of traces, apply/removal of gain on amplitudes, frequency filters and the inversion itself use the Numpy arrays architecture. The Tkinter GUI is designed to simplify the use of the program, also providing interactive tools along with Matplotlib in a way that it is possible for users to click on seismograms to pick first arrival times, preview live travel-time curves, draw apparent velocity lines and identify layers in travel-time curves.

The Sisref program is the interpretation module, which is based on the time-terms analysis suggested by Scheidegger & Willmore (1957) and Willmore & Bancroft (1960), where a linear system is assembled from the direct and refracted wave travel time vectors. The model parameters were the velocities of each layer and the thicknesses below each geophones. The inversion process used relies on a least squares approach, with a regularization term based on a

smoothness matrix included in the objective function, to stabilize the problem (Meju, 1994).

The time-term approach has been widely applied for decades all around the world to interpret the geometry of irregular refractors on shallow seismic refraction surveys. Scheidegger and Willmore (1957) suggest that the travel time of a refracted wave is directly affected by delay times, or time-terms. Given sufficiently dense coverage, it is possible to determine the two-dimensional distribution of propagation velocity for the materials immediately below the boundary, and all of the delay times which describe the transmission of the waves through the upper layer. A conventional time-term solution assumes that the material below the boundary has a uniform propagation velocity, and that the delay time for any given survey point is the same for all the ray paths which begin and end at that point. When applying the time-term method, one is likely to start without prior knowledge of the structure, and must, therefore, make a trial solution which ignores the effects of possible inclinations of the refracting boundary and other structural anomalies. Even the identification of onsets can be in doubt until trial values of time-terms are available for insertion on time-distance plots (Willmore, 1969). As detailed by Willmore & Bancroft (1960), the use of the least squares approach might be applied to solve the inverse problem of the time-terms linear equations.

The problem is to determine the constants in systems of equations for direct (1) and refracted (2) arrivals, of the types

$$t_{ij} = \frac{\Delta_{ij}}{v_1} \text{ (1) } t_{ij} = \alpha_i + \beta_j + \frac{\Delta_{ij}}{v_k} \text{ (2)}$$

where α_i and β_j are the delay times or time-terms (not present in the direct waves equations) of the i[th] shotpoint and j[th] receiver position, Δ_{ij} is the distance between the i[th] shot point and the j[th] receiver, t_{ij} is the time of the wave propagation from the i[th] source position until the j[th] receiver, v_1 is the velocity of propagation of the direct wave at the surface and v_k is the seismic velocity of a refracted wave in the k[th] underlying marker layer (Scheidegger & Willmore, 1957).

The thickness of a k[th] layer below the j[th] receiver (H_{kj}) can be calculated as a function of the mean velocity above the refractor ($\underline{v}$), the velocity in the k[th] layer (v_{k+1}) and the time-term of the j[th] receiver (β_j) (Willmore & Bancroft, 1960; Smith *et al*., 1966), yielding the relation

$$H_{kj} = \frac{\beta_j . \overline{v} . v_{k+1}}{\sqrt{{v_{k+1}}^2 - \overline{v}^2}} \text{ (3)}$$

Assuming that m receivers observe waves from a total of n sources, up to nm equations of forms (1) and (2) might be obtained. Apart from v**,** the problem contains n+m unknowns. The search for physical parameters that generate a theoretical model as close as possible to the actual geological model in situations such as this is known as the inverse problem (Menke, 1984).

Rewriting these linear systems into a stochastic expression, we get

$$\vec{d} = G\vec{m} \text{ (4)}$$

where $\vec{d}$ is a vector of travel times, G is a coeficiente matrix and $\vec{m}$ is the unknowns parameters vector. Rewriting (1) and (2) into the (4) matrix form, we get

$$[t_{ij}] = \left[\frac{\partial t_{ij}}{\partial \frac{1}{v_1}}\right]\left[\frac{1}{v_1}\right] \text{ (5) } \quad [t_{ij}] = \left[\frac{\partial t_{ij}}{\partial \alpha_i} \; \frac{\partial t_{ij}}{\partial \beta_j} \; \frac{\partial t_{ij}}{\partial \frac{1}{v_k}}\right]\begin{bmatrix}\alpha_j \\ \beta_j \\ \frac{1}{v_k}\end{bmatrix} \text{ (6)}$$

for i = 1,2,3,...,,m and j = 1,2,3,...,n. Once $\vec{d}$ and G are defined, it's possible to calculate a solution for the $\vec{m}$ vector using an ordinary least squares approach (Willmore & Bancroft (1960); Menke, 1984), of form

$$\vec{m} = (G^T G)^{-1} G^T \vec{d} \text{ (7)}$$

Once the first arrival times of each opened seismic section is picked on the software, the interpretation of the travel-time curves needs to be done. At this point, the user should identify, by clicking, if a specific arrival time represents layer 1 (direct wave with a v_1 velocity) or layer 2 (refracted wave with $v_2 > v_1$). All clicked arrival times are then included in a specific numpy array object for each existing layer ($\vec{d}$ vectors). After all layers are identified by the user, the algorithm builds a G matrix for all existing layer (a n dimensional numpy array object), technically being ready to compute the inversion described in (7) to estimate m (if the layer selections can be invertible).

The direct solution calculated by (7) solves $1/v_1$, for the direct wave problem, and $1/v_k$, for the refracted layers, due to the velocity parameter usually being a well resolved unknown (Hatton *et al*., 1986; Meju, 1994), but might not yield the best possible time-terms for the real model of the surface refractors (6), as result of the geological formation of the survey area or even an unstable $G^T G$ inversion.

Inversion problems in geophysics often deal with overdetermined systems of equations, where an undetermined number of solutions are possible due to a high amount of data (Menke, 1984; Meju, 1994). A priori information of at least one of the unknowns is the best possible way to constraint the linear system into computing a unique solution for the problem (Hatton *et al*., 1986). When a priori information is not available, the best approach one could hope for to solve the non-uniqueness of the inversion is to add smoothness constraints in the system of equations (i.e. adding smoothness relations between the unknown parameters) (Meju, 1994). An often applied estimative when creating such smoothness constraints is that the refractors have small angulations, which, along with a small spacing between the survey receivers, may suggest that the time-term parameters of the refractor vary a small amount in relation to each other. This approach usually is presented as a regularization matrix of form

$$\begin{bmatrix}0 \\ \vdots \\ 0\end{bmatrix} = \begin{bmatrix}\delta & -\delta & & & \\ & \delta & -\delta & & \\ & & \cdots & & \\ & & & \delta & -\delta\end{bmatrix}\begin{bmatrix}p_1 \\ \vdots \\ p_h\end{bmatrix} \text{ (8)}$$

where δ is the Lagrange multiplicator (weight of the smoothness). Refrapy then concatenates a zeros vector to the $\vec{d}$ numpy arrays of the 2[nd] layer. The algorithm also forms the regularization matrix in a numpy array, concatenating it into the G matrix of the refractor layer. This approach yields $p_1 \cong p_2$, $p_2 \cong$

$p_3,\ldots,p_{h-1} \cong p_h$ (smoothed parameters). The $1/v_k$ parameter is usually left unconstrained for reasons already mentioned. From this point forward, the ordinary least squares (7) can be used to compute an unique constrained solution for the $\vec{m}$ vector.

By changing the value of δ in (8), the algorithm computes and stores different unique constrained solution - due the different weight applied - resulting in different residual values for each solution found, which are also stored in the memory in a way that the program can select the solution related to the lowest residual found to represent the final velocity model by computing (3). The ideal and applied approach is to let the δ value float freely so that a range of solutions are created and residuals values can be compared. It is presented by Meju (1994) that, for a significant range of δ, the solution related to the least residual value might not always match the parameters of the inverse problem given a solution calculated with a constrained a priori information (the most appropriate solution), but is the best possible estimative by the use of this technique one could find to solve a non-unique overdetermined linear inversion problem with lack of a priori data, which ignores the effects of possible inclinations of the refracting boundary and other structural anomalies.

For the results, a synthetic model and seismic sections were created with the Seismic Unix package, and a sample survey was executed in the Campus of the University of Brasília. The parameters used to create the synthetic model were: thickness of the first layer = 5 m, v1 = 300 m/s, v2 = 2100 m/s, 24 channels with a 2 meters spacing from one another and 5 sources locations along the whole profile. For the field acquisition, the impact of a sledgehammer as an energy source was used. The seismic energy was registered by high frequency geophones distributed in a linear profile with a 2 m spacing from one another. To test the program, all the processed data and the respective results were compared with the ones obtained with the trial version of the Geometric's Seisimager/2D commercial program, an often used software in seismic data processing all around the world.

Results

A comparison between the developed and the commercial phase picking programs can be visualized for real data in Figures 2 and 3. The Sispick program presents a simple and very similar processing routine when compared to the commercial reference software (Pickwin).

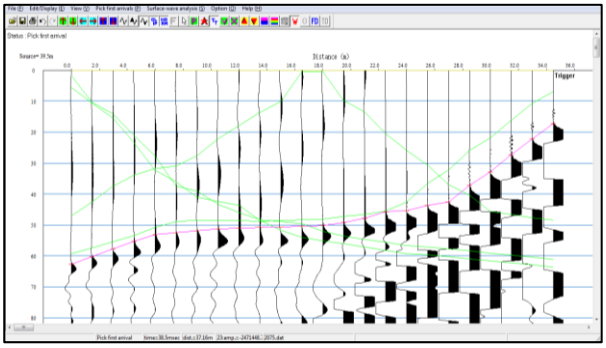

Figure 2 *–First breaks picking on a SEG2 seismogram made with the Pickwin program from Seisimager/2D.*

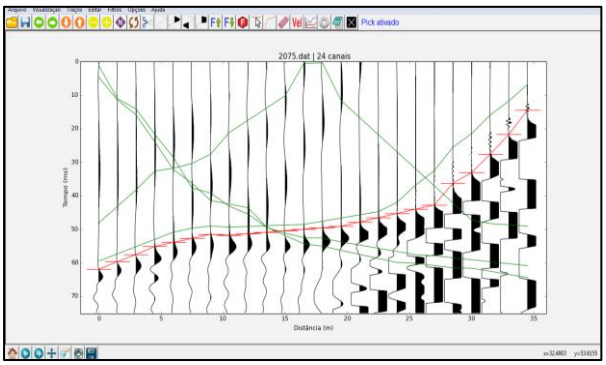

Figure 3 *– First breaks picking on the same SEG2 seismogram made with the Sispick program from Refrapy.*

Real data

The inverse method to define the solution vector for the speed and depth parameters of the refractor layer is performed in the Sisref module, which creates an easily interpreted velocity model once the travel-time curves are created with the Sispick program. Figures 4 and 5 present the interpretation of travel-time curves from the real data set with Sisref (Refrapy) and Plotrefa (Seisimager/2D). Both velocity models created at the end can be visualized in Figures 6 and 7.

An average difference of 1.23 m between the calculated thickness of the first layer from both results was found (Fig. 8). The Sisref program returned an average thickness of 6.58 m for layer one, while Plotrefa returned 7.81 m. The Refrapy program calculated v1 = 338 m/s and v2 = 2009 m/s. The commercial application returned the exact same result for v2, but a value of 353 m/s for v1, a 4.25% error, which directly affects the difference found in the thickness of the first layer.

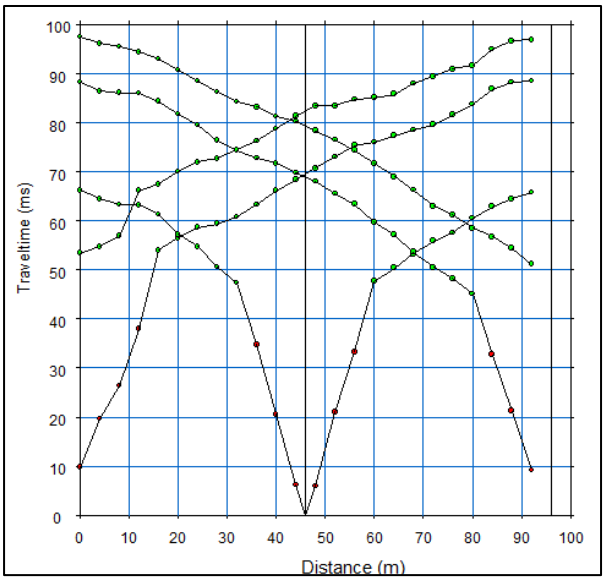

Figure 4 *– Interpreted time-travel curves of the simple field survey in UnB (Plotrefa – Seisimager/2D).*

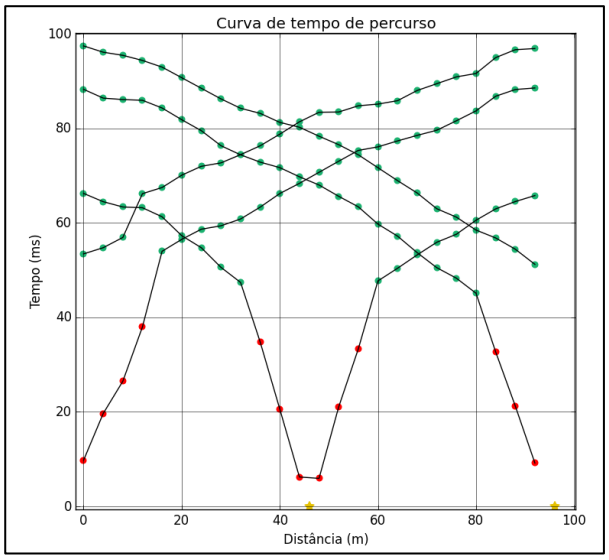

Figure 5 *– Interpreted time-travel curves of the simple field survey in UnB (Sisref - Refrapy).*

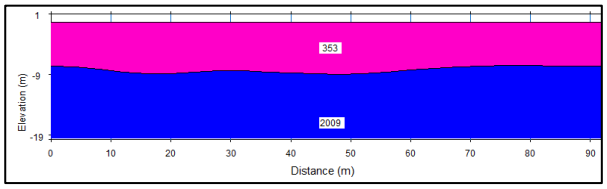

Figure 6 *– Calculated velocity model created from interpreted time-travel curves in Fig. 4 (Plotrefa – Seisimager/2D).V1 = 353 m/s and V2 = 2009 m/s.*

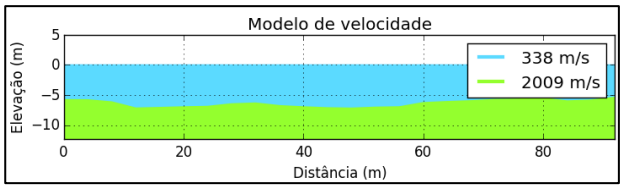

Figure 7 *– Calculated velocity model created from interpreted time-travel curves in Fig. 5 (Sisref – Refrapy).V1 = 338 m/s and V2 = 2009 m/s.*

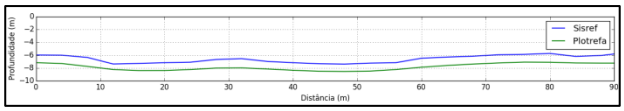

Figure 8 *– Comparison between the thickness of the first layer from both velocity models in Fig. 6 and 7.*

Synthetic data

Figure 9 presents the synthetic geological model of two parallel plane layers in depth generated by the Seismic Unix package (Cohen *et al*., 2013). Figure 10 shows the first arrivals picking in one of the 5 synthetically generated seismograms. All created seismograms were originally in ".su" format and were converted to SEGY by the Siscon module. The commercial picking program couldn't open neither SU nor SEGY waveform files.

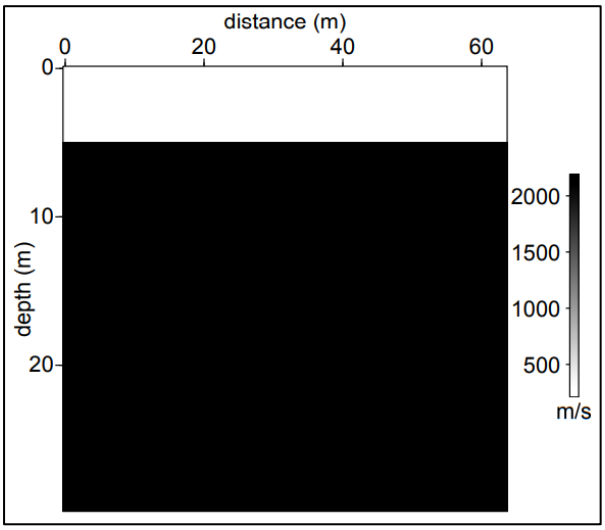

Figure 9 – Synthetic geological model of two parallel plane layers in depth generated by Seismic Unix.

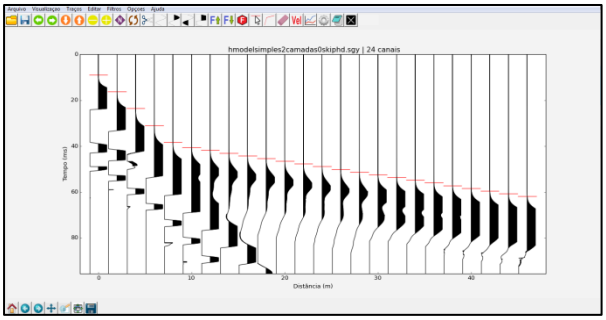

Figure 10 *– First breaks picking in the Sispick program (Refrapy) on a SEGY* synthetic seismogram created by Seismic Unix from model in Fig 9.

Figures 11 and 12 present the interpretation of travel-time curves from the synthetic data set with Sisref (Refrapy) and Plotrefa (Seisimager/2D). Both velocity models created at the end can be visualized in Figures 13 and 14.

The Refrapy program presented an average thickness of 4.7 m, yielding a 0.3 m difference when compared to the 5 m thickness of the original synthetic model (error of 6%). The commercial program returned an average thickness of 5.6 m, a 0.6 m difference when compared to the same synthetic model (error of 10.71%) (Fig. 15). Sisref (Refrapy) calculated v1 = 317 m/s (error of 5.37% when compared to the original 300 m/s of the synthetic model) and v2 = 1822 m/s (error of 13.24% when compared to the original 2100 m/s). Plotrefa (Seisimager/2D) returned v1 = 310 m/s (error of 3.23%) and v2 = 1820 m/s (error of 13.34%).

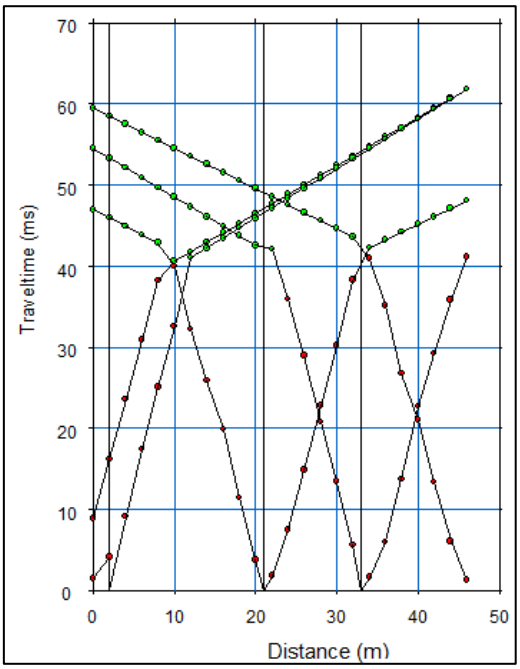

Figure 11 *– Interpreted time-travel curves of the synthetic data (Plotrefa – Seisimager/2D).*

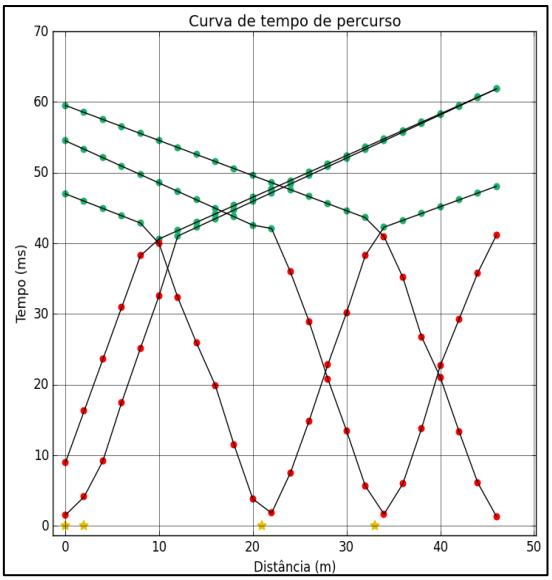

Figure 12 *– Interpreted time-travel curves of the synthetic data (Sisref – Refrapy).*

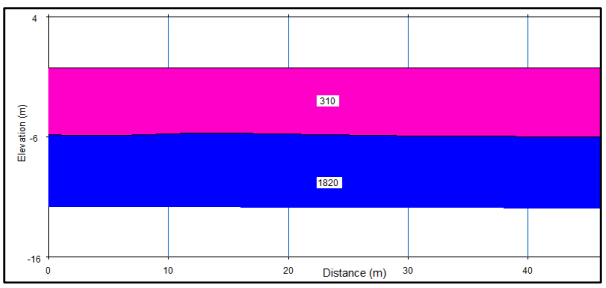

Figure 13 *– Calculated velocity model created from interpreted time-travel curves in Fig. 11 (Plotrefa – Seisimager/2D).V1 = 310 m/s and V2 = 1822 m/s.*

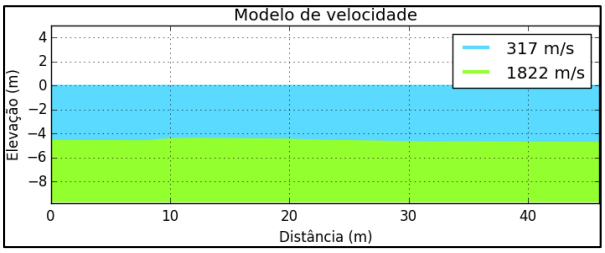

Figure 14 *– Calculated velocity model created from interpreted time-travel curves in Fig. 12 (Sisref – Refrapy).V1 = 317 m/s and V2 = 1822 m/s.*

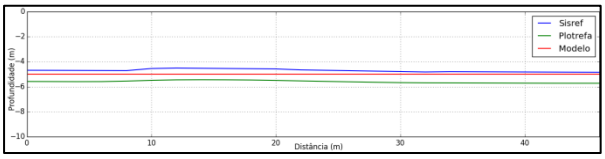

Figure 15 *– Comparison between the thickness of the first layer from the syntethic model and the velocity models from Fig. 13 and 14.*

Conclusions

The libraries used in the development of the free package showed good efficiency, streamlining data processing routines, optimizing flow control structures and allowing several approaches to the algorithm development. The commercial package presented some tools and interpretation methods not yet included in the free package, although Refrapy presented some useful functions not yet present in the last version of the paid software.

The results generated by all the Refrapy programs were considerably compatible with the results of the commercial package used as reference, both for real and synthetic data. The first breaks picking and trace processing program from Refrapy (Sispick) showed great efficiency, managing to open SEG2 and SEGY waveform archives and creating files of travel-time curves that could be opened in both of the intepretation softwares used in this work (Sisref from Refrapy and Plotrefa from Seisimager/2D).

For the real data processing, the error values found while comparing both interpratation softwares when appliying the time-term analysis method were not significant considering the dimensions of the layer, beeing in an acceptable range to characterize the subsurface in terms of seismic velocity and depth. Considering the processing of the synthetic data, both interpratation programs compared in this work presented a practically identical margin of error for the calculated velocities. However, the Sisref (Refrapy) program showed greater coherence regarding the thickness of the first layer of the original synthetic model.

The small differences between the calculated parameters between ours and the commercial software is one of the goals for future research on the inversion algorithm, along with the implementation of a third layer analysis.

Acknowledgments

ProIC – Programa de Iniciação Científica da Universidade de Brasília, for the research funding during the preliminary development of this work.

Seismological Observatory of University of Brasília, for providing the hardware used to create the synthethic data and for providing the facilities during some periods of the development of the algorithms.

Geosciences Institute of University of Brasília, for providing the equipment for the field acquisition survey used in this work.

References

BEYREUTHER, M., R. BARSCH, L. KRISCHER, T. MEGIES, Y. BEHR, J. WASSERMANN. *ObsPy: A Python toolbox for seismology*, 2010.

COHEN, J. K.; STOCKWELL, J. J. W. CWP/SU: Seismic Un*x Release 43R5: an open-source software package for seismic research and processing. Colorado School of Mines, 2013.

HATTON, L.; WORTHINGTON, M. H. L.; MAKIN, J. Seismic Data Processing: Theory and Practice. Blackwell Scientific Publications, pp.139-163, 1986.

HUNTER, J., D. *Matplotlib: A 2D graphics environment*. Computing in Science and Engineering 3, 9, 90-95, 2007.

JONES, E., OLIPHANT, T., PETERSON, P. *SciPy: open source scientific tools for Python*, 2001.

MEJU, M., A. *Geophysical data analysis: understanding inverse problem theory and practice*, chapter 4-5, 1994.

MENKE, W., *Geophysical Data Analysis: Discrete Inverse Theory*, Academic Press, Inc., New York, 1984.

SCHEIDEGGER, A., E., WILLMORE, P., L. *The use of a least squares method for theinterpretation of data from seismic surveys*. Geophysics, vol. 22, issue 1, pp 9-22, 1957.

SMITH, T. J.; STEINHART, J. S.; ALDRICH, L. T. Lake Superior Crustal Structure. Journal of Geophysical Research. Vol. 71, No. 4, pp 1141-1172, 1966.

WALT, S., V., D., COLBERT, S., C., VAROQUAUX, G. *The NumPy array: a structure for efficient numerical computation*.Computing in Science and Engineering, 13, 22-30, 2011.

WILLMORE, P. L. Comment on How Necessary are large-Scale Refraction Experiments?. Geophys. J. R. Astorn. Soc., 18:227-230, 1969.

WILLMORE, P., L., BANCROFT, A., M. *The time term approach to refractionseismology*. Geophysical Journal, vol. 3, issue 4, pp 419-432, 1960.