PySLI: A software for Sparse Layer Inversion using Python

*Satyajit Swain *, Mamata Jenamani, Aurobinda Routray (IIT Kharagpur), Sanjai KumarSingh[4], Rajeev Tondon[5], CS Bahuguna[6] (GEOPIC, ONGC, Dehradun)*
satyajitchiku.swain@gmail.com

Keywords

Sparse Layer Inversion (SLI), Basis Pursuit Decomposition (BPD), Linear Programing (LP), Graphical User Interface (GUI)

Abstract

This paper aims to develop a python based interactive geophysical software called PySLI – Python based software for Sparse Layer Inversion. This software will help the geophysicists to perform inversion with ease. The software has three major components: data visualization, well analysis and Sparse Layer Inversion. We implement two algorithms for sparse layer inversion: *Basis Pursuit Decomposition* (BPD), and *Modified Basis Pursuit Decomposition using linear Programming*. The software also provides a facility to select the right wedge dictionary and model parameters showing a range of options to choose from.

Introduction

Sparse layer inversion detects thin layers which are invisible in the conventional seismic method. Tuning thickness of seismic method is $\lambda/4$, and resolution is $(\lambda/8)$, where λ is the wavelength of seismic wave. The Seismic method is unable to detect layers whose thickness is less than tuning thickness. Sparse layer inversion increases the resolution of the seismic method and can detect thin layers.

Sparse layer inversion increases resolution of seismic inversion and finds hidden reservoirs that are not detectable using conventional seismic methods. Resolution of the seismic method is defined as the ability to separate or distinguish two or more closely spaced layers. Widess(1973) defined the theoretical resolution limit as 1/8th of the wavelength $(\lambda/8)$. The tuning thickness usually occurs at about $\lambda/4$ and in practice; layers below tuning thickness are not readily resolved (Kallweit and Wood, 1982). Chopra et al. (2006) have used spectral inversion technique to resolve thin layers below tuning thickness. Spectral inversion resolution is further increased to 1/16th of wavelength by Puryear and Castagna (2008). Some other algorithms such as Matching pursuit decomposition (MPD) (Nguyan and Castagna, 2010), Multichannel matching pursuit decomposition (MCMP) (Wang, 2007,2010) have been developed to improve uniqueness and spatial continuity. Zhang and Castagna, (2011) have introduced Basis pursuit decomposition (BPD) which has many advantages over earlier algorithms. BPD is better in handling interference between dictionaries and has efficient computational ability. It can exhibit good lateral stability even when dictionary elements are not orthogonal. We have also included a modified basis Pursuit approach which further enhances the seismic resolution and has been tested with real data. With all this functionality and visualization a software is designed named PySLI.

We faced a number of challenges while building the proposed *PySLI* software using open source Python Libraries. First, the complexity of the algorithms and parameter adjustments which not only consumes a lot of computing resources but also requires frequent manual intervention. Second, the volume of seismic data itself possess a lot of challenge. Third, the GUIs also consumes a lot of resources. Managing the memory and parallelization of the algorithms are the key in making a successful software in this case. We present below the SLI methodologies implemented, the software design overview, implementation overview, and a use case scenario of the proposed software.

SLI Methods implemented

A. Basis Pursuit Inversion

The forward convolution model can be represented as

$$d = Gm + n$$

Where d is the Seismic trace column vector, m is the Coefficient vector, G is the kernel matrix, and n is the noise Vector.
Basis Pursuit is an optimization routine which minimizes the $L2$Norm and hence induces sparsity in the solution vector. It minimizes the following objective function:

$$\text{Min } [\|d - Gm\|_2 + \lambda\|m\|_1]$$

The reflectivity series can be decomposed into a summation of even and odd impulse pairs as follows

$$r(t) = \sum_{q=1}^{Q}\sum_{e=1}^{E}(a_{q,e} * r_e(t,q,e,\Delta t) + b_{q,e} * r_e(t,q,e,\Delta t))$$

B. Linear Programming Approach

To modify the basis pursuit problem, the least-squares solution has been converted into the least absolute solution.
This provides less constraint on the major coefficients than minor reflection coefficients. Hence the objective function can be rewritten as

$$\text{Min } [|d - Gm| + \lambda|m|]$$

This can be substituted in the following form.

$$\text{Min } (y + \lambda z)$$

Where $y = \|d - Gm\|_1$ is the accuracy parameter and $z = \|m\|_1$ is the sparsity norm. λ is the trade-off parameter between sparsity and accuracy of the solution. With the increase in λ value, sparsity in solution vector increases. High λ value is not helpful for this problem as it would only reveal the significant reflection coefficients. Decreasing the λ value allows the presence of other minor spikes in the solution vector, but after a specific limit, decrease in λ value would only create ghost events in the solution vector. We have chosen the value of lambda after experimentation with well log data. In this paper, the interior point method has been used to minimize the objective function as a linear programming problem.

Design overview

The software architecture is shown in Fig 1. And the corresponding explanation is explained in Table 1.

Blocks	Description
1)Data	
a) Upload segy section	To upload seismic data.
b) Upload well log data	To upload well log data.
c) Exit	To smoothly exit from the GUI.
2) Sparse layer inversion	
a) Synthetic Reflectivity	To generate reflectivity from well.
b) Parameterization using well log	This section is use to determine lambda value.
c) Seismic inversion	This consist of different algorithm to generate inverted reflectivity.
d) Well Map	This section helps is showing information of all the wells in that survey area.
e) Export seismic data section	This section is used to write the data inverted again into segy format.

Table 1. Brief Overview of the Software Architecture

Implementation Overview

Besides the basic python packages such as numpy, scipy, matplotlib necessary for scientific computing environment we use five more packages with their dependent packages which is explained in Table 2.

Memory management

As the seismic data has very large size so designing a software is a real challenge which handles a data without clogging the GUI as well as memory. So the first difficulty is to load the data through the GUI for inversion process. For that we are using the open source segpy module to read and write the seismic data as the data is large in size it will take time to load which in turn will cause the memory issue and GUI will not respond. So for that a very popular technique for software development is used i.e. threading technique which will not clog the GUI and

will continuously monitor the progress of the data being read using a progress bar.

Data can be used multiple times to load, for that the GUI creates a pickle which holds the data path and header info and in turn fetches the data directly from the hard disk without loading onto the memory which in turn reduces the reading time of the data drastically. Seismic information is important to inspect for proper processing for example time range, sampling time, inline, cross line range etc. software will fetch all the details from the seismic data and will show to the user.

Code parallelization

The algorithm will take its own time to invert and the number of traces to invert is large so if we go by serial processing it will takes days to invert a small section which is not at all feasible. We need to use some good parallelization techniques to process multiple traces at a time which will reduce the inversion time drastically we can also go for GPU as these are the best machine for parallel computation . We have use the multiprocessing module to handle parallel computing.

SL NO.	Packages Used	Tasked Performed
1.	**PyQt5**	Developed Graphical user interface with this module
2.	**Segpy**	Reads and writes the seismic data (.sgy)
3.	**Lasio**	Reads and writes the log data(.las)
4.	**Gurobipy**	Performs the linear programming.
5.	**Multiprocessing**	Used for parallelization of the algorithm
6.	**Numpy , Scipy**	Former is used for mathematical operations and the later for signal processing respectively.

Table 2. The brief Overview of the Packages used

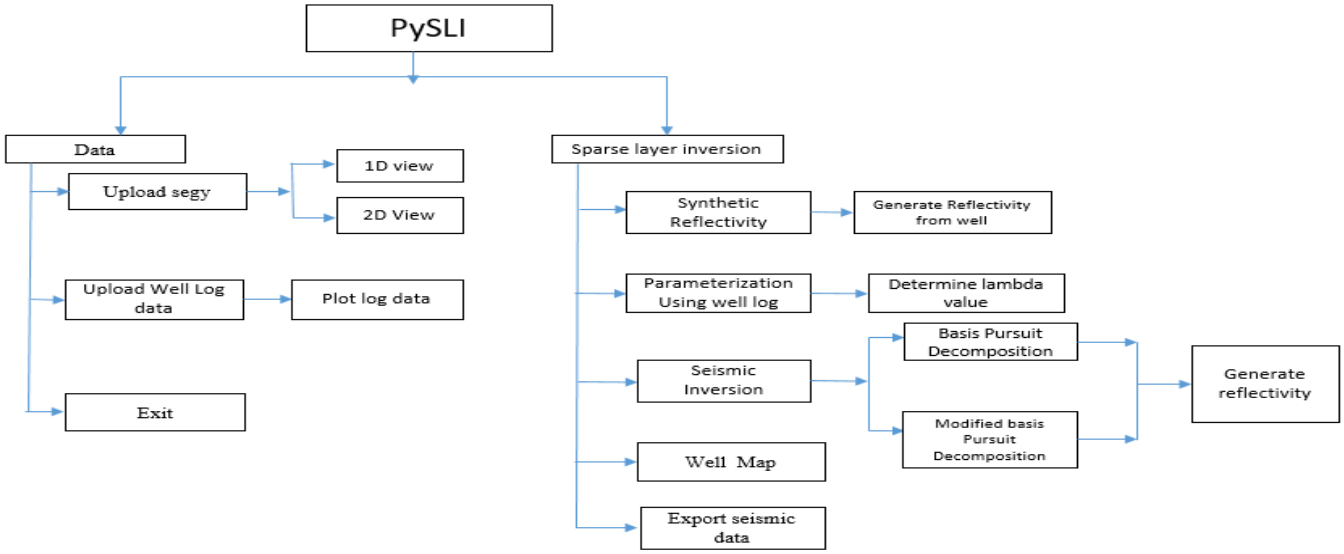

Figure 1. The Software Architecture

A Use Case of PySLI

Consider a scenario when a geophysicist wants to perform an inversion using this module. The following is divided into four different parts.

A) **Uploading the data:** To perform inversion we first need to choose a data to be inverted. So we will upload a data in the upload data section. The process is shown in Fig 2.

B) **Calculating the Optimum lambda Value:** To proceed for inversion the optimum value of lambda should be determined. The process is shown in Fig 3.

C) **Applying Algorithms:** After calculating the lambda value we will go for inversion using two available algorithms. We can choose any one of it and it will automatically open that algorithm section. After providing appropriate parameters and calculated lambda value the inversion process will start which can be monitored using the progress bar in that section. On completion of the process it will generate inverted reflectivity with *(.csv)* extension. The complete process is shown in Fig 4.

D) **Exporting to proper seismic format:** The *(.csv)* file generated should be converted into proper seismic format *(.sgy)* as the seismic is a convolution of reflectivity with a wavelet. So in this section we are taking Ricker as default wavelet. The complete process is shown in Fig 5.

After generating the inverted seismic. We can visualize the section i.e. original and inverted data in 2D. The comparison figure is shown in Fig 6.

The comparison clearly shows the hidden layers are visible which are not there in the original data.

Conclusion and Future Scope

PySLI currently implements two popular method for sparse layer inversion. The results are validated in only window platform with a specific dataset. It requires extensive testing with validation from industry experts. Though we parallelized some of the task for a CPU cluster, we need to make the algorithm faster. We are planning to write GPU code to reduce the inversion time and to improve 3D visualization.

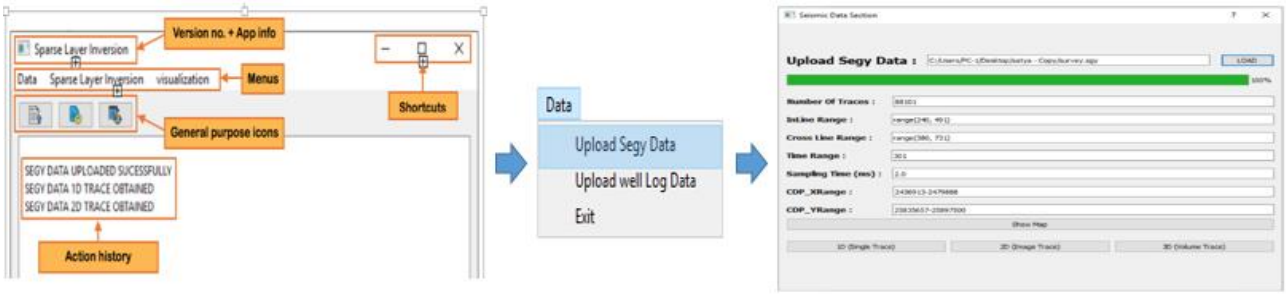

Figure 2: Workflow to upload the seismic data

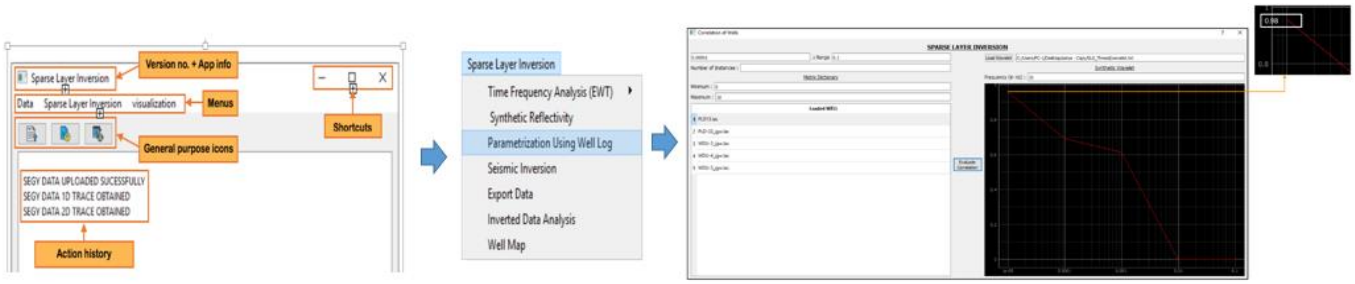

Figure 3: Workflow to determine the optimum lambda value.

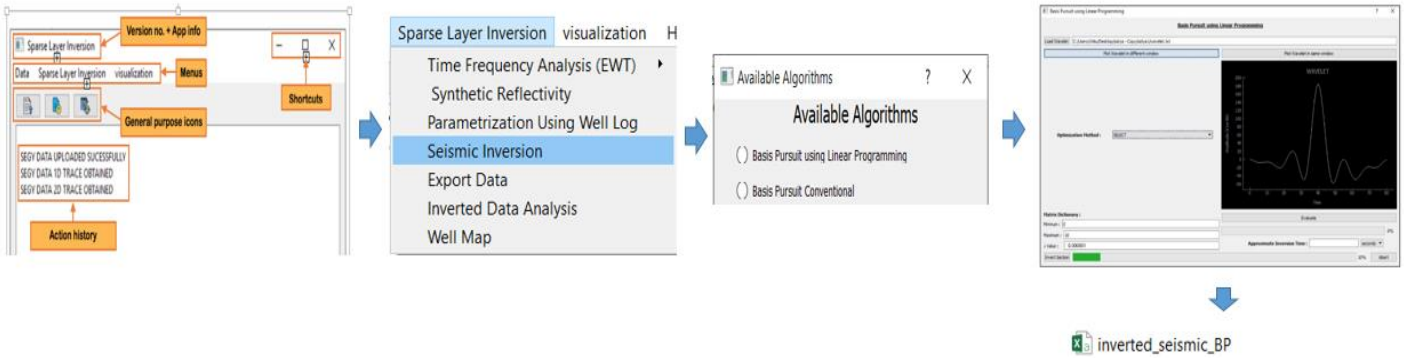

Figure 4: Workflow to apply particular Algorithm.

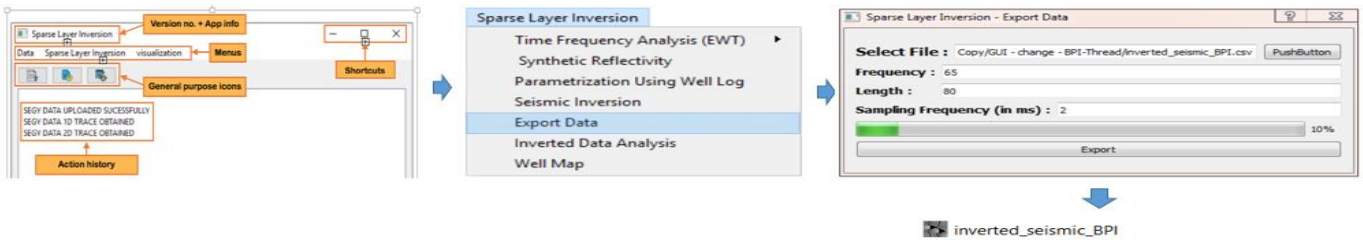

Figure 5: Workflow to convert the inverted data *(.csv)* into proper seismic format *(.sgy)*

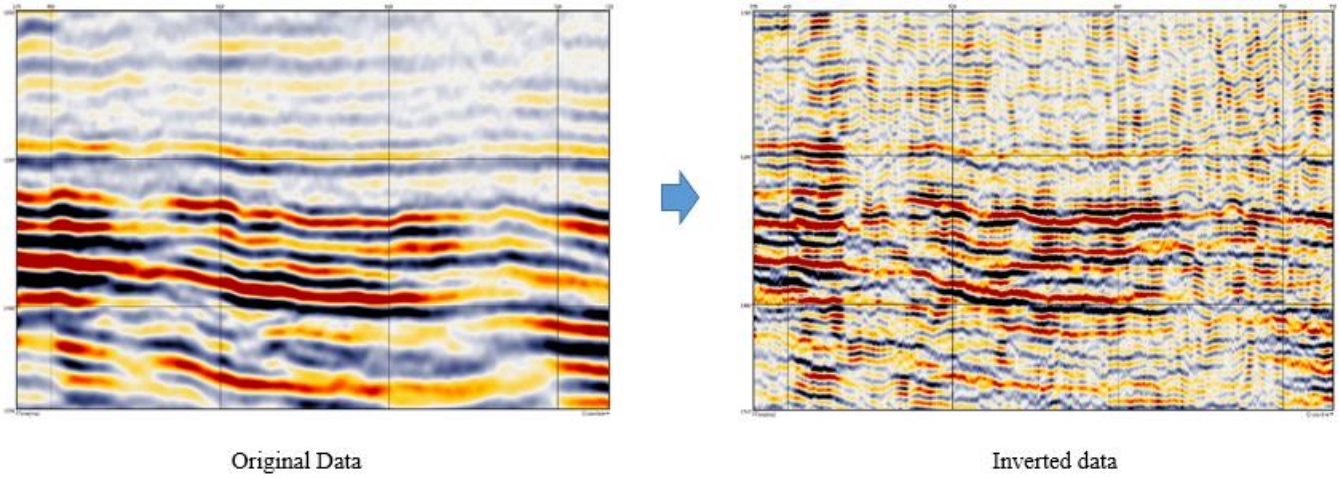

Figure 6: A 2D comparison between original and inverted data

REFERENCES

Chopra, S., Castagna, J.P., and Portniaguine, O., 2006, Thin bed reflectivity inversion: SEG Technical Program Expanded Abstract, 25, 2057-2061.

Donoho, D.L., 2006, Compressed sensing, IEEE T. Inf, Theory, 52, no-4, 1289-1306.

Kallweit, R.S., and Wood, L.C., 1982, The limits of resolution of zero phase wavelets, Geophysics, 47, no-7, 1035-1046.

Nguyen, T., and Castagna, J.P., 2010, High resolution seismic reflectivity inversion, Journal of Seismic Exploration, 19, 303-320.

Puryeara, C.I., and Castagna, J.P., 2008, Layer-thickness determination and stratigraphic interpretation using spectral inversion: Theory and application, Geophysics, 73, R37–R48.

Wang, Y., 2007, Seismic time-frequency spectral decomposition by Matching pursuit, Geophysics, 72, no-1, V13-V20

Wang, Y., 2010, Multichannel matching pursuit for seismic trace decomposition, Geophysics, 75, no-4, V61-V66.

Widess, M.B., 1973, How thin is a thin bed? Geophysics, 38, 1176-1180.

Zhang, R., and Castagna, J.P., 2011, Seismic sparse layer reflectivity inversion using basis pursuit decomposition, Geophysics, 76, no-6, R147-R158.

Acknowledgments

This work has been undertaken by Indian Institute of Technology, Kharagpur in collaboration with Geo data Processing & Interpretation Centre (GEOPIC), ONGC, Dehradun under the aegis of ONGC-PANIIT projects.