Seismic Data Analysis and Signal Processing for Smart-Solo Geophones

Bryan Azbill

Abstract

Beyond wave detection, smartsolo geophones are furnished with a wide array of sensors that provide data necessary for signal processing seismic data, sensors that provide data about orientation and position, recieved by gps, as well as temperature, an adjustable gain and sample rate, damping coefficients, and more. Metadata from these systems are dumped into a text file, along-side a miniseed encoding that stores the actual seismic detection. By providing such a broad arrangement of metadata, SmartSolo geophones can be deployed simply and flexibly, in contrast to devices that require far more extensive testing for identifying theirfrequency response, geographic surveying to be precisely positioned, sheltering to keep a constant temperature, and other challenges. While such single stations, having been deployed at wide distances all around the globe, are ideal for identifying the hypocenters of major seismic events, SmartSolo arrays are more ideal for local deployment, able to detect smaller but more proximal events. This makes them ideal for seismic surveying with intent to identify local fracture networks, induced seismicity, shear wave splitting, or similar interests. By their configurability and flexibility can the SmartSolos provide seismic data suited to the particular needs of a survey. However, some technical understanding is necessary to determine the proper settings necessary before deployment in order for data to be collected that best suits the particular details of the expermint. Furthermore, once the smartsolo data is collected, the challenge remains for it to be properly processed. Making appropriate decisions to meet these challenges require a combined application of data science, physics, and signal processing, in order for a SmartSolo array to be deployed at its full potential. Fortunately, multiple packages for python are available to greatly simplify the work. This report will discuss in detail the challenges of seismic signal processing with smartsolo geophones, scripts in python with the packages available to overcome them, and experiments already performed with the smartsolos. Spectrograms, timeseries, geographic maps, bode plots, and cross correlations have all been generated for discussion and analysis of the data collected during a smartsolo deployment of twelve geophones.

1 Introduction

Data from a seismic array presents multiple identification challenges, including:

b) Detecting seismic arrival events for the whole array c) Through greater precision, detecting differences in arrival time between geophones a) Determining direction of wave displacement c) Detecting seismic arrival amplitudes d) Determining the type of waves arrivingClustering arrivals by seismic events Determining earthquake times and hypocenters

Machine learning systems provide new and effective algorithms for overcoming some of the challenges of seismic processing. A convolutional neural-net system with a "Unet" architecture, phasenet, is capable of identifying both wave arrival time and amplitude from seismic data with high accuracy. However, even well trained machine learning systems must be fed well processed and structered data if they are to be effective. Therefore, scripting with pythons well designed and documented package for seismic array processing, Obspy, must be done before smartsolo data can be processing through Phasenet, which uses obspy as well. Additionally, accurate as phasenet is for triggering on seismic arrivals, its only optimized for sample rates near 100hz, but some tasks may require a much higher resolution. For example, if a wave velocity field is desired from a seismic survey over a geographic region no larger than a few kilometers in diameter, high frequency sampling is required to resolve the miniscule differences in time it will take surface waves, which travel several kilometers a second, to traverse relatively small distances between geophones. Even with deployments ranging several kilometers, previous experiments with measuring shear wave splitting have demonstrated that sampling at a rate of 500hz and months worth of data greatly improved the precision they needed to measure directionally anisotropic differences in the seismic velocity field, a precision of less than ten meters a second. On the other hand, weeks of high frequency data can easily fill up hundreds of gigabytes for an entire seismic array, and becomes rather cumbersome to process if not preconditioned effectively. A straightforward method of such preconditioning is to run a prelimenary phasenet event detection for data from a single geophone, compressed to 100hz, in order to identify the short time windows during which seismic waves are traversing

the array. This way, a cross correlation between high frequency station datastreams can be applied over these narrowed time windows determing the seismic arrival differentials one event at a time.
There are also other operations datastreams must be preprocessed with in order for results to be accurate that would be rather cumbersome across large high frequency datasets, excepting they are only applied to short event time windows. For example, each smartsolos station records its own roll, pitch, and yaw. These measurements are required to perform a geophone attitude adjustment, which means to transform the wavestreams from geophone centric X,Y,Z to a N,Z,E alignment. Even if great effort is put into deploying the devices with a perfect orientation,they are likely to be at least a little crooked. and besides, Phasenet won't even process wavestream channels unless their labeled as N,Z, and E components.
Theres a separate sort of calibration is necessary for the amplitude of seismic events to be accurately predicted: the sensitivity of wavestream response to ground motion as a function of frequency. Referred to as the transfer function, it varies between each geophone and each of their three components. Phasenet needs these transfer functions defined and formatted in a technical xml file which can be generated in python once the seismic array is structured with obspy. There are also other reasons to structure seismic arrays with obspy also has its own advantages, beyond working with other systems such as phasenet. It's provides algorithms to plot wavestreams, spectrograms, transfer functions and maps. It has all the signal processing routines most projects will need, and its structure also help keep workflow organized. Once the seismic data is successfully formated through obspy to be fed into phasenet, the phasenet itself reads and preprocesses data with obspy as well. Automating the clustering of seismic arrivals is done with an algorithm called GaMMa. GaMMa also makes predictions for event time and hypocenter that can be further improved later with another algorithm called hypoDD. Others before me have already scripted an automation of this entire flow, from reading miniseeds and frequency response files to predicting the details of a seismic event, and even scripted them to run on the cloud for live detection in a project called QuakeFrow. Unfortunately, the differences between the sort of networks QuakeFlow was written for and SmartSolo arrays mean seismic processing can not be entirely trivialized through QuakeFlows algorithms. $IGU-16HR_3C$ SmartSolos experimented with in this report do not broadcast their data, but only store their data in solid state memory. The additional work described in this report was done because the data formatted by the SmartSolos gps and other systems required its own unique scripting. Therefore much of the code simply has to do with with reading and processing the smartsolo data into obspy. The reward of such work is when users of SmartSolo arrays are able to deploy them where and when they please, read the data off a SmartSolo connection rack, and then simply run a python script which accurately automates the entire process of reading data logs, transfer functions, and miniseed files to predicting event times, hypocenters, velocity fields, and more, while also providing options for maps and figures to display results for experiments.

2 Design

Before being deployed, settings must be set on the devices, which can be done through a smartsolo data transfer rack, and the soloLite software. The most important settings on smartsolo devices are their sample rate, and channel gain. After its been deployed once a smartsolo has been retrieved from the ground and the dirt cleaned off, it needs to be positioned in a smartsolo rack transfer socket for its data can be read via usb connection. The waveforms are stored in collections of 256mb miniseed files, and everything else goes into a file titled DigiSolo.LOG. All the metadata needed for an experiment can be retrieved from this file, which holds a great of other useful information. Infact, timestamped data is collected in DigiSolo.LOG for over 95 parameters that cover everything from gps records, booting and device info, memory fill, the starting time of each miniseed file, and more, which all gets read to be processed in pythons seismic array package, obspy. Obspy has a specific, nested object structure for describing seismic arrays. At the top of this structure are Inventory objects. The inventory objects hold network objects. Network objects hold station objects, representing geophones, and each station has three channel components to represent the x,y,z components of each geophone.
Rather than manually coding such a large number of details, it is better write a script, which as it reads through the smartsolo log, keeps a dict to track every parameter its seen in the file above. It either adds a new entry, when a novel entry parameter is encountered, or adding to a pre-existing data list. The time, if available, is extracted from the rest to be used as an index. This way, everything in the entire log file gets read into python in a clean syntax. Once this is complete, Some of the time series require further processing which reduces them to a single metric value. The SmartSolo GPS records values every few minutes, and a simple average is sufficient for position and orientation, but it's necessary to run an outlier detection first, because some of the timeseries collect spurious data, even if only during deployment and collection. Then, object-oriented inheritence provides a convienent way to tie data directly to the obspy objects. With a class SS_Station,

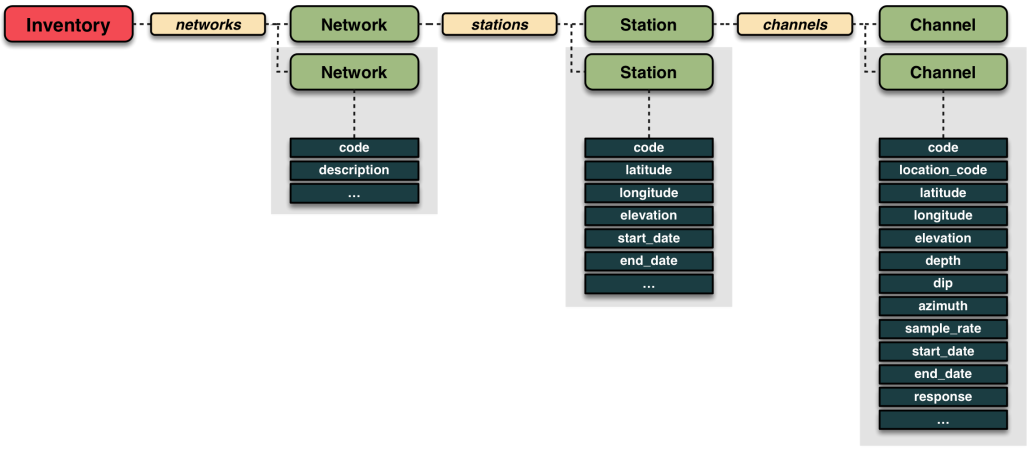

Figure 2.1: A diagram of the obspy data structure, from the documentation

written in extension of the Obspy station objects, SS_Stations have been extended with a readLog function to be called during their initialization. SS_Net and SS_Inventory are similarly defined. Functions are then written to SS_Net that iterate initialization, plotting, and other operations for each station in the network, where necessary.

The smartsolo transfer function, which defines its frequency response, however, can not be accurately deduced from data within the LOG file. Instead, Smartsolo has provided a single testing port for such data. The testing port sits aside the data transfer ports, can be operated using SmartSolo's testing software, and actually pulses a waveform to each channel for the device, simultaneously recording its response for the devices, and storing the records in an excel sheet. More scripting is then required for each device to be able to find the test which matches within these files, extract the data, and then apply further mathematical processing to predict and define the devices transfer function in way that can be read and understood by the PhaseNet code. More precisely,

$$H(s) = \frac{s^2}{(1 + T_D * s) * (s^2 + 2 * T_D * D_d * w_0 * s + w_0^2) + KH * s^2} \tag{1}$$

This is the transfer function defined in the user manual for IGU-16HR_3C smartsolo geophones. w_0 is the resonant frequency of the device, D_d is the damping coefficient

Twelve geophones were scripted and deployed on September 26th. Geophones were deployed in six pairs, with pairs spaced about a meter apart. For each pair, one device was set as a control, while the others were set to experiment with the devices settings, including higher gain, lower gain, higher sample rate, lower sample rate, minimum phase anti-aliasing, and constant gps. Four of these twelve were recovered, read, and redeployed on September 3rd. Then all geophones were recovered on october 13th, and all were read but one, which was had a corrupted file system, from having been one of those redployed on the 3rd.

3 Results

From the records of eleven geophones, the following plot shows all numerical timeseries recorded by SmartSolo devices, excluding only timeseries for relating to device memory

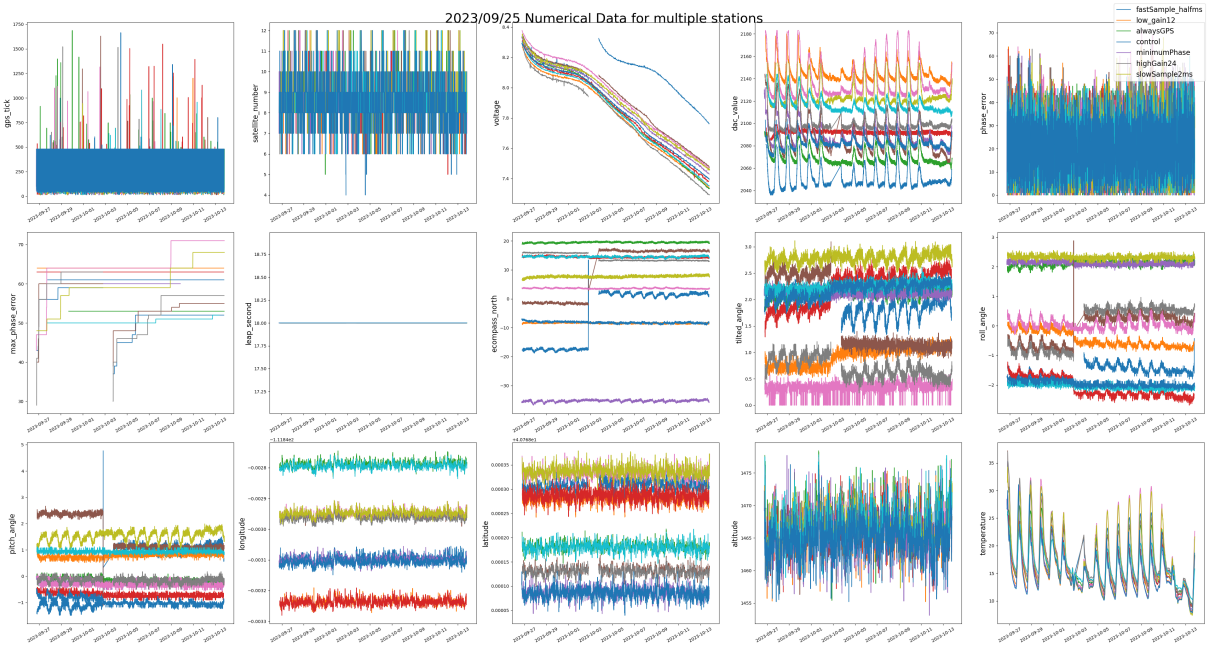

Figure 3.1: Numerical Data Plots for 11 Geophones

A few things stand out in this image. The orientation measures strongly correlate with temperature. This is undesirable, but can be fixed with a detrending. The high gain device used more power. The positions of some of the geophones were swapped when they were redeployed on the 3rd. Latitude and Longitude averages were then exported to google earth to generate the following plot:

Between the image and current state of the field position error of the deployment is undifferentiable. From the standard deviation of the latitude and longitude time series, the precision is given to about half a meter. However, it is observed that the magnitude and direction of error is nearly the same between geophones, and therefore the measured distance between them is even much more accurate than that. The precision of the small distance between the closely deployed pairs of geophones is clearly visible in the google earth figure.

A sample waveform collection is plotted in **??**. It appears there may have been a seismic event recorded at about 9:00 UTC. However, by generating a spectrogram with obspy, a much better understanding of of the signal can be read.

With most of the sample rates set to 1000 hz, the amount of data collected from eleven three channel geophones was massive, 134.5GB. Such large datasets make operations such as channel orientation adjustments incredibly slow and cumbersome. However, a review of data shows the geophones were able to recieve virtually no signal above 100 hz

4 Conclusion

5 Future Works

6 References

A Appendix A

Figure 3.2: positions for 11 Geophone deployment

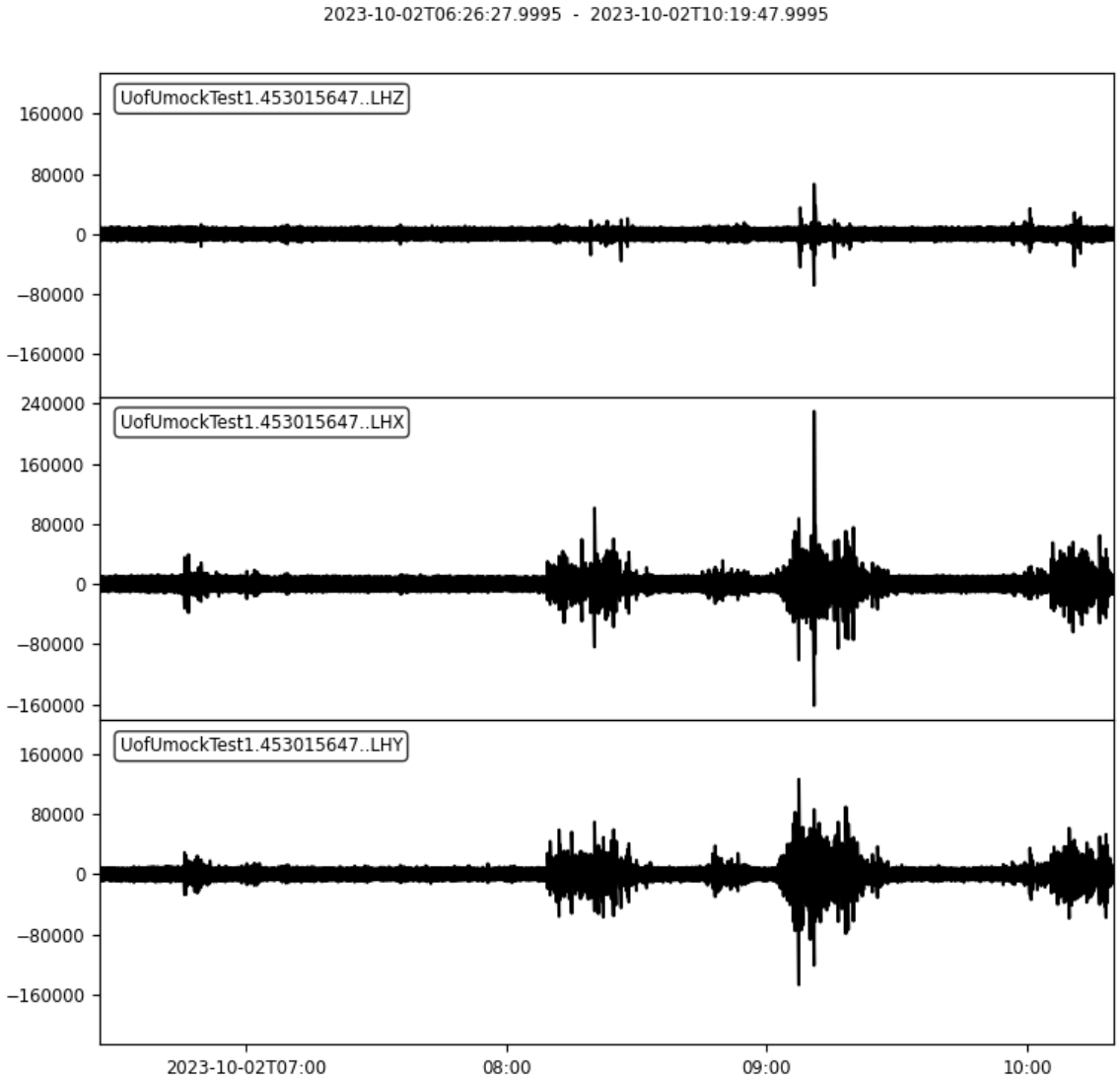

Figure 3.3: positions for 11 Geophone deployment

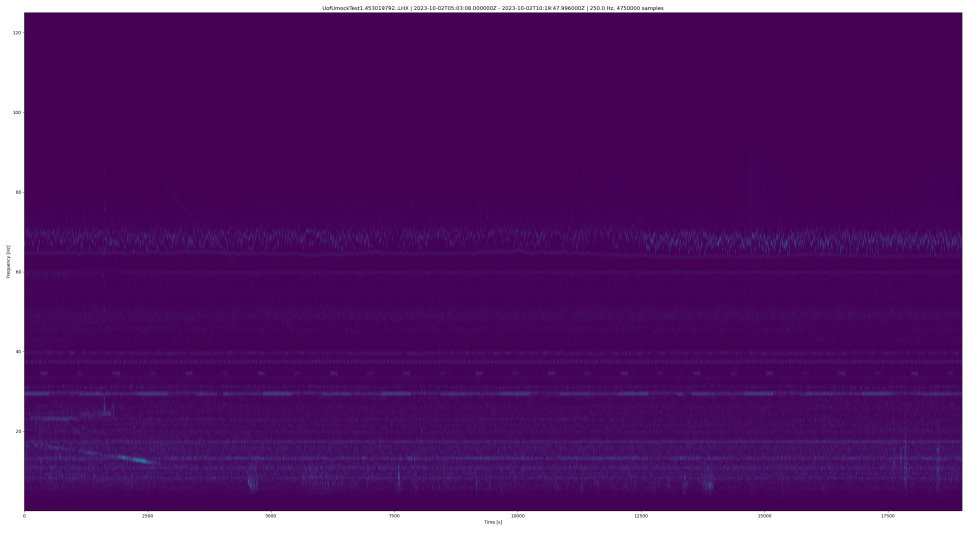

Figure 3.4: The spectrogram of the waveform in **??**

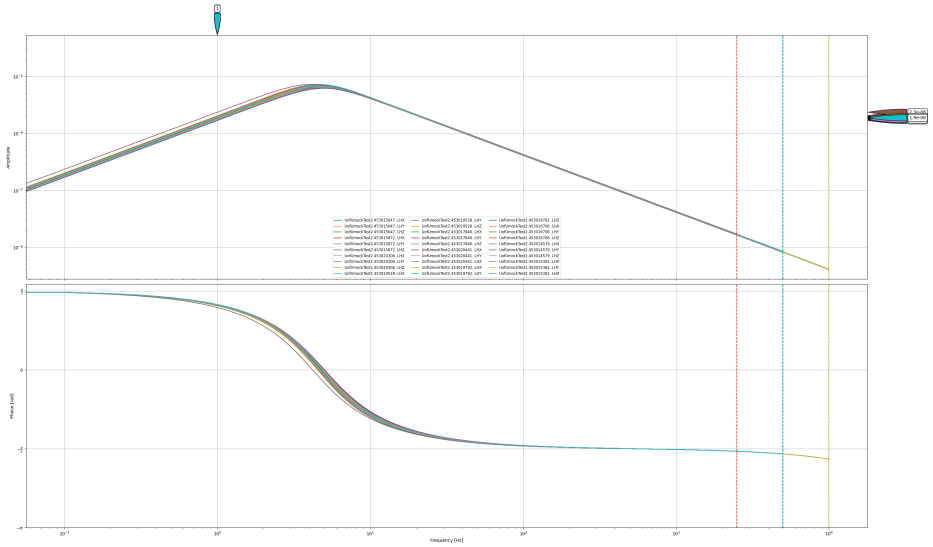

Figure 3.5: The spectrogram of the waveform in **??**