BTech 451 Mid-Year Report

PyjAmaseis

An Application for Educational Seismology

Name: Saketh Vishnubhotla
UPI: svis267
ID: 2655131

12th August 2014

Table of Contents

Introduction

Seismology is a topic hardly dealt with or taught in schools. This is due to several reasons ranging from not having a set curriculum to not having the tools in the classroom to demonstrate and simulate earthquakes. In an attempt to promote the teaching of Seismology in schools and to make it the best educational experience for the students, the Incorporated Research Institutes of Seismology (IRIS) have invested into an educational program Seismographs in Schools in a bid to provide all the required components to teach Seismology in schools. IRIS has created a curriculum involving classroom activities, quizzes, and learning content. This resource is freely available for teachers to access, allowing them to structure it into a subject. However, there is more to what is being offered. The government has funded thousands of schools to acquire a TC1 educational seismometer that can be used in conjunction with the curriculum provided by IRIS to make it a highly interactive course for students. Unfortunately the software that is available today was not developed from a teaching perspective. Therefore making them too complicated for teachers and students to use.

My 4th year BTech 451 project is based on creating a software application that can meet the needs of the IRIS curriculum and most importantly reduce all the complexity whilst providing all the important features required for teachers and students to engage with seismic data.

In my project so far I have focused on the core features such as reading from the TC1, saving the data, and finally displaying it live on the screen. This report also talks about the technologies used in creating the application and the challenges I have faced so far. Along with this, a list of all the work I plan to do before the end of the semester is covered.

This report serves to outline and investigate into the problem, and implementation of a potential solution. This report contains all the work that has been carried out from the third week of first semester up to the third week of the second semester including the intersemster break.

Feedback from project presentations has also been taken into consideration and has been addressed in this report.

Setting the Scene

Seismology and Seismic Waves

Seismology is the study of earthquakes and seismic waves that move through and around the earth. Seismic waves are the waves of energy caused by the sudden breaking of rocks within the earth or an explosion. They are the energy that travel through the earth and is recorded on seismographs.

IRIS (Incorporated Research Institute for Seismology)

Founded in 1984 with support from the National Science Foundation, IRIS is a consortium of over 100 US universities dedicated to the operation of science facilities for the acquisition, management, and distribution of seismological data. IRIS programs contribute to scholarly research, education, earthquake hazard mitigation, and verification of the Comprehensive Nuclear-Test-Ban Treaty.

Figure 1. IRIS – Incorporated
Research Institutes of Seismology

IRIS Seismographs for Schools

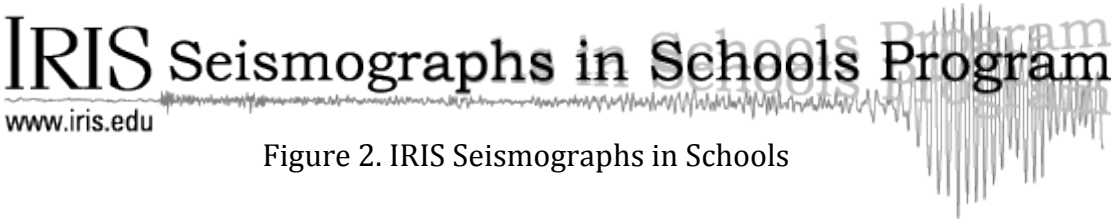

Figure 2. IRIS Seismographs in Schools

One of IRIS's educational programs – Seismographs in Schools, serves teachers across the country and around the world using seismic instruments such as the TC1 educational seismometer or real-time seismic data in K-16 classrooms. The IRIS's site includes tools to share seismic data in real-time, curriculum to teach seismology, classroom activities, and technical support documents for seismic instruments. Their hope is to bridge the gap between science classrooms to create an international educational seismic network.

TC1 Educational Seismometer

The TC1 is a simple vertical seismometer. Its natural period is less than one second. The current version reflects the latest set of modifications since the first version in 2005. It is very "Kid" friendly and uses a "Toy Slinky Spring", magnets, and a coil. The entire seismic system is composed of the sensor, an amplifier called NERdaq, a computer, and a software application.

The signal coming directly from the seismometer is quite weak and requires amplification in order to see activity. It also requires conversion form an analog

to digital signal. Martin Smith and Chris Knudsen have created the NERdaq: an amplifier, filter and Analog-to-Digital signal converter in one device.

It may be a surprise to those unfamiliar with the TC1 that this seismometer does a remarkable job of seeing and recording earthquakes, small and large, from near and far. It can record earthquakes from all around the world. Most home-built seismic sensors target recording events greater than 7M Worldwide. This system does much better, recording greater than 6M in most cases worldwide and local events greater than 2M.

The government has provided funding to thousands of schools in USA to acquire the TC1 seismometer. In a similar way, the NZ government is also looking into doing the same.

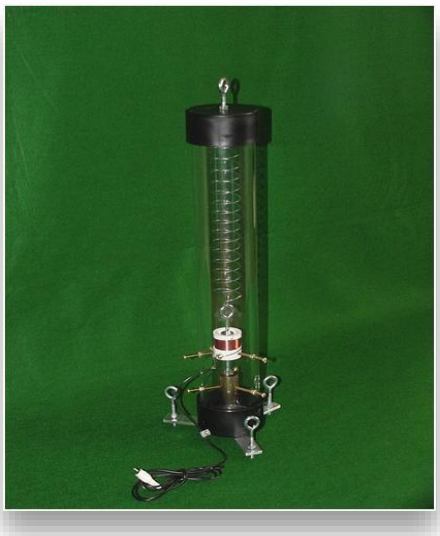

Figure 3. TC1 Seismometer

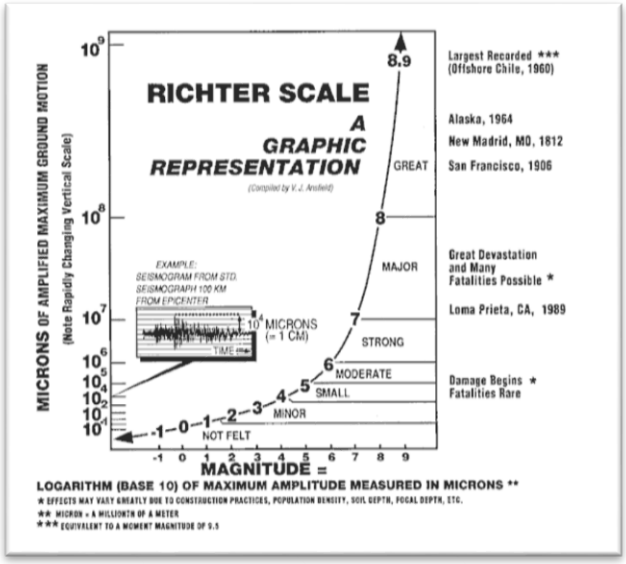

Figure 4. Richter scale showing Magnitudes

Problem

I would like to now summarize what has been said above and introduce the problem this project aims to address. As you have read, IRIS has an educational program called Seismographs in Schools that they have made available to teachers teaching Seismology in schools. IRIS aims to provide the curriculum, technical documentation for various educational seismometers and classroom activities and software tools to teach Seismology and allow students engage with seismic data.

However, the problem here lies with the software. The current software that is available and is being used by the teachers to teach is far too complicated to be used in a classroom scenario. Most seismology suites are built for professional seismologists and the less professional ones are still too difficult for students to use or learn seismology from.

So you can think of this as a three-piece puzzle, where the first piece is the curriculum, which is in already place (provided by Seismographs for Schools program), the second piece is the hardware (TC1 educational) which is also available, but the final piece – a simple easy to learn and use application that supports the curriculum and provides the basic features required to engage with seismic data is missing.

Existing Solutions

There are a number of libraries and modules available that let us work with seismic data but not too many applications. Among these most are limited to one platform (runs on only one operating system) and are too complex for students to engage with.

A professional seismic processing and imaging solutions suite is offered by Paradigm. This software is highly professional and was built for professional seismologists. The Paradigm seismic processing and imaging solutions reduce uncertainty and improve reliability through better seismic signal quality, positioning, and content. Their proprietary algorithms translate billions of bits of seismic data into highly accurate, high-definition images of the subsurface, enabling geoscientists to visualize the earth's formations.

Although Paradigm seismic processing and imaging solutions provide high definition imaging tools and accurate seismic data plotting, it's far too complicated for teachers let alone students to use in order to learn seismology.

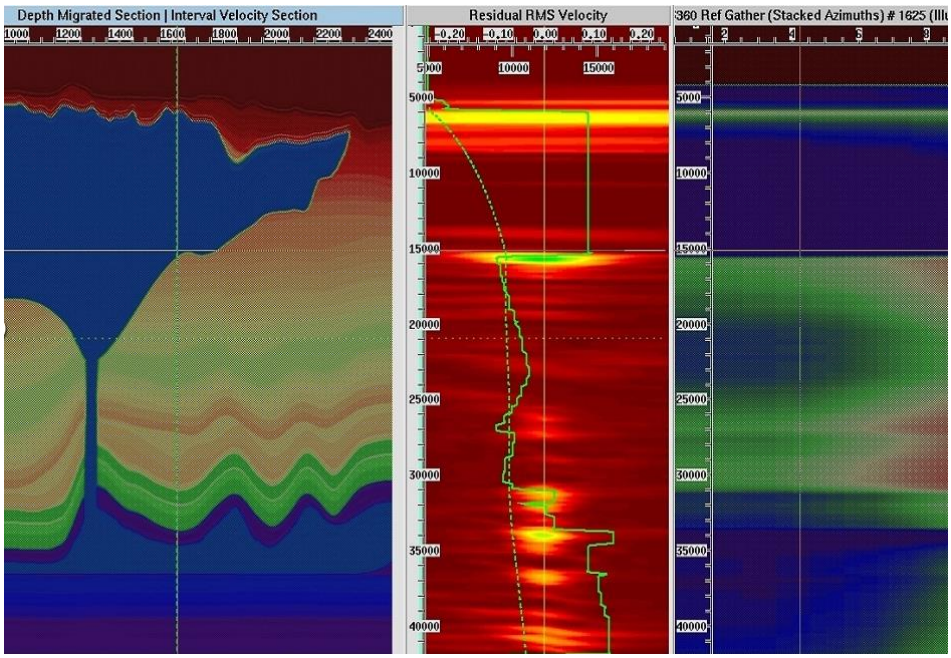

Figure 5. Paradigm software showing GeoDepth velocity determination, modeling and imaging

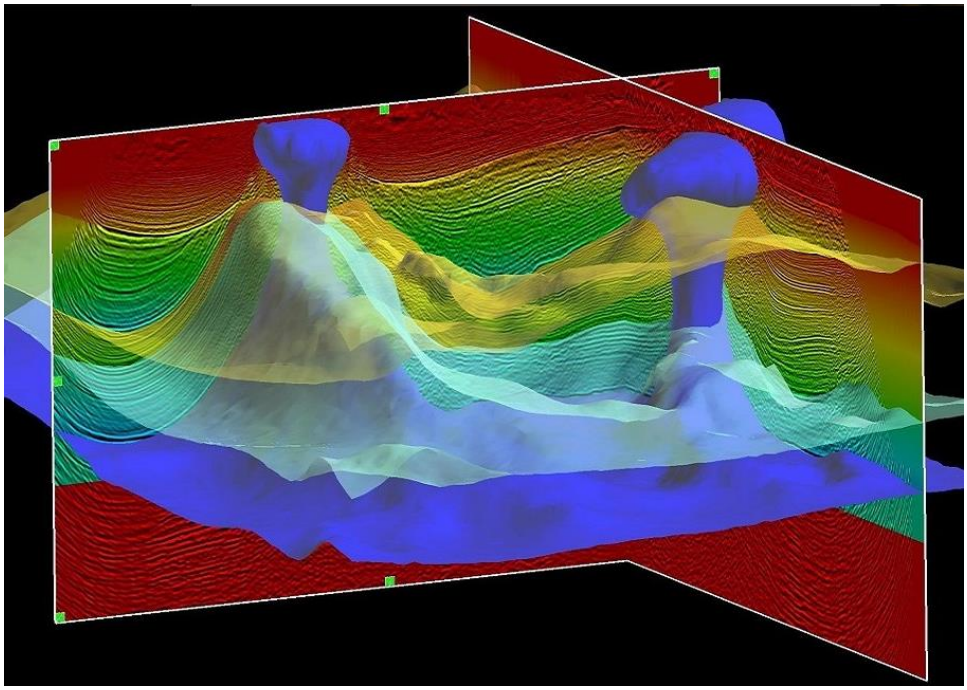

Figure 6. Paradigm software showing seismic velocity model with salt

Let's now take a look at the software that is currently being used to teach seismology. This software is called jAmaSeis.

jAmaSeis facilitates the study of seismological concepts and allows users to obtain data in real-time from either a local instrument or from remote stations. As a result, users without an instrument can utilize the software. Users can view a graphical representation of seismic data in real time and can process the data to determine the characteristics of seismograms such as time of occurrence, distance from the epicenter to the station, magnitude, and location (via triangulation).

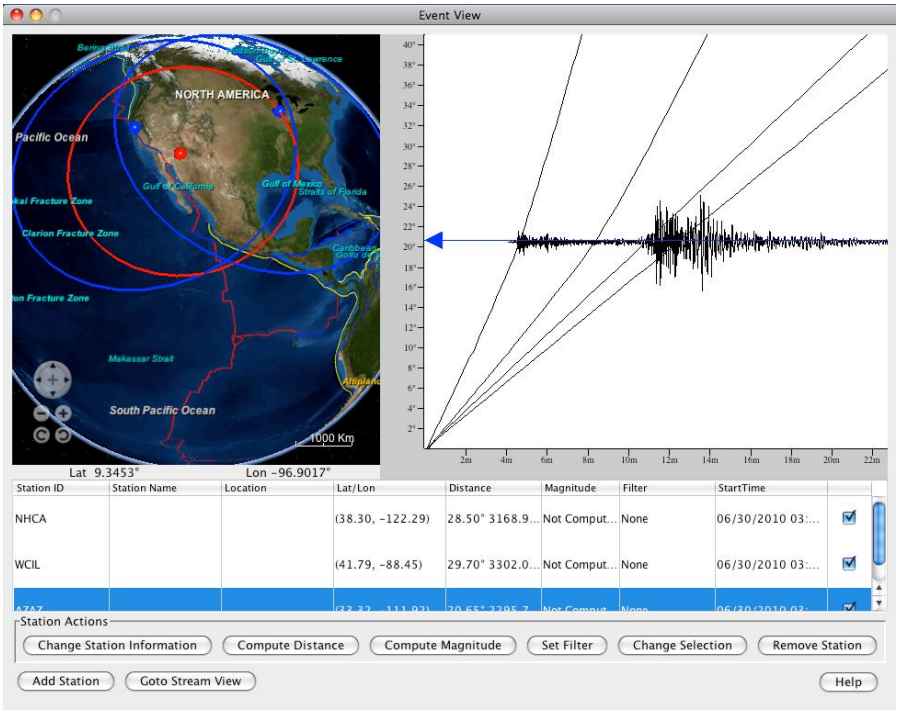

Figure 7. jAmaSeis Event mode analysis

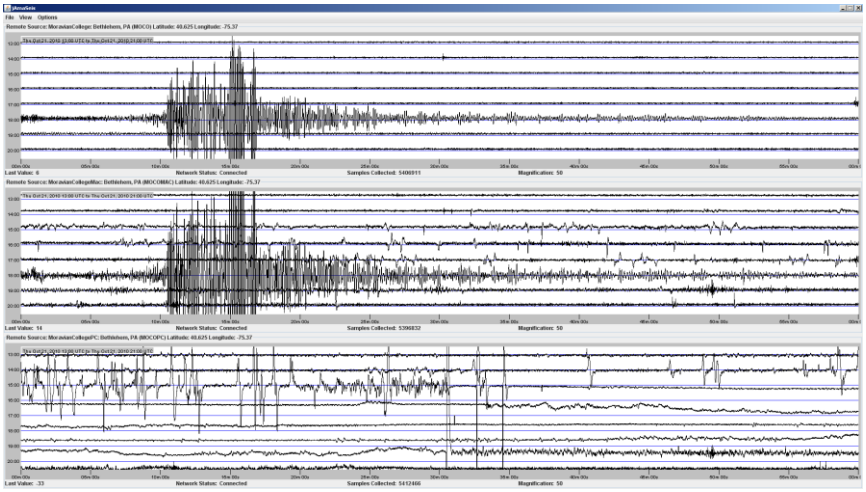

Figure 8. jAmaSeis Stream View

jAmaSeis contains all the basic functionality required in a seismology data processing suite to fit the Seismographs in Schools Program curriculum and goals, however it has been reported to be a struggle for students to start getting comfortable with using and learning from this software as the user interface is not easy to understand and not many learning cues provided. The software provides the functionality but is not focused on being a learning tool and most of the intricate functionality also is not used.

Solution

The missing piece of the puzzle is a software application that can replace jAmaSeis whilst ensuring it contains all the core functionality required to be an effective seismology data processing and sharing suite. The solution needs to be specifically targeted to support the learning of students by integrating effective visual cues that increases student engagement and understanding of seismology. Hence this application will need to provide all the essential features of a seismology suite and improve on where jAmaSeis lacks which is not being focused on ease of use and intuitiveness.

The solution will aim to incorporate the following elements:

- The essential functionality
    - Collecting information from TC1 seismometer
    - Processing this data
    - Displaying data live
    - Saving data
    - Sharing data
- Simple and Intuitive Interface
- User Friendly
- Accuracy
- Robust
- Fault Tolerant

Thus creating a software application for seismology by incorporating the above functionality, we can create the perfect learning environment for students to learn and engage with seismic data.

Project Overview

Project Outline

The aim of this project as identified above, is to create an application that is cross platform and incorporates all the important elements described above.

The project scope is to build a simple and intuitive native python based application that will process, display and share earthquake signals with other schools online. I will be developing a robust and user-friendly application that can be easily used by teachers and students to engage with seismic data. I will build a python-based interface by incorporating ObsPy (seismic data processing suite in python) and Matplotlib (comprehensive 2D plotting library).

Core Functionality

The core functionalities that is required to be built into the application are:

- Reading (from TC1 seismometer)
- Processing (decoding, casting)
- Displaying (Live Plotting)
- Saving (mseed)

- Sharing (Seedlink Server)

Extra Functionality

The extra functionalities that will be looked into are:

- Zoom in/out on plot
- Save selected part of a plot as an mseed
- Automatic Screenshots of the plot
- Set custom x (minutes) and y (hours) axis
- Visual cues that map seismic data to preset scenarios to identify cause of a seismic behavior

These extra functionalities are required in order to make the application more effective and easier to work with. Some of these allow the student to get a more detailed view of the seismic data displayed and some enhance the learning by pointing at specific parts of the seismograph and explaining its behavior.

Technologies and Services

To carry out this project my supervisor has requested me to develop the application in Python and utilize the ObsPy and Matplotlib modules to achieve the required functionality.

Below are the technologies and services that will be used to develop the application:

- **Python 2.7**
  - Application will be developed using the Python programming language. This application will be cross platform.

- **ObsPy 0.92**
  - ObsPy is an open-source project dedicated to provide a Python framework for processing seismological data. It provides parsers for common file formats, clients to access data centres and seismological signal processing routines, which allow the manipulation of seismological time series.

- **Matplotlib 1.3.1**
  - A comprehensive plotting library for Python
  - Used for plotting the data collected from the TC1 seismometer

- **GNUPlot 4.6.2**

- Plotting application used to plot data in text file

- **Seedlink Plotter**
  - Creates 24 hour day plots using hour long mseed files

- **Seedlink Server**
  - Hosts mseed files
  - SEED stands for Standard for Exchanging Earthquake Data
  - Mseed stands for Mini Standard for Exchanging Earthquake Data

- **jAmaseis 1.00.0**
  - Existing seismology application which I referred to, to learn how to display seismic data

- **Git**
  - Used to keep track of various versions of the application and keep track of incremental changes

- **tKinter 3.3.5**
  - Graphical user interface library for python

Project Scope

There are 3 components to this project:

1. Local Python-based application that can process, display and save seismic data (mseed)

2. Adding remote access capabilities by being able to post saved mseed files and regular application screenshots to Seedlink Server, and retrieving and displaying seismic data and screenshots from the Seedlink server

Implementation

The aim of the development is to work through each core feature and implement it correctly then move on to extra functionality. After creating a requirements document with Dr. Kasper van Wijk, I began working on my project. I started by setting up the Python environment and installing the PySerial module on my laptop running windows.

Pymaseis v0.1- Reading

The first version of the application was simply focused on reading the data coming in from the TC1 via the serial port and printing these values to see what I was getting.

The Python script I wrote, therefore I opened the port to read this data using the *stream.readline()* method from the PySerial module. I then saved this information in a .dat file (data.dat) which would then be plotted using the application called GNUPlot.

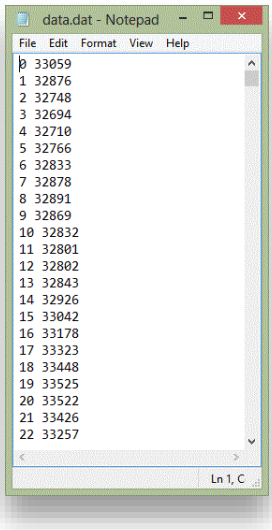

Figure 9. Data from TC1 saved in data.dat

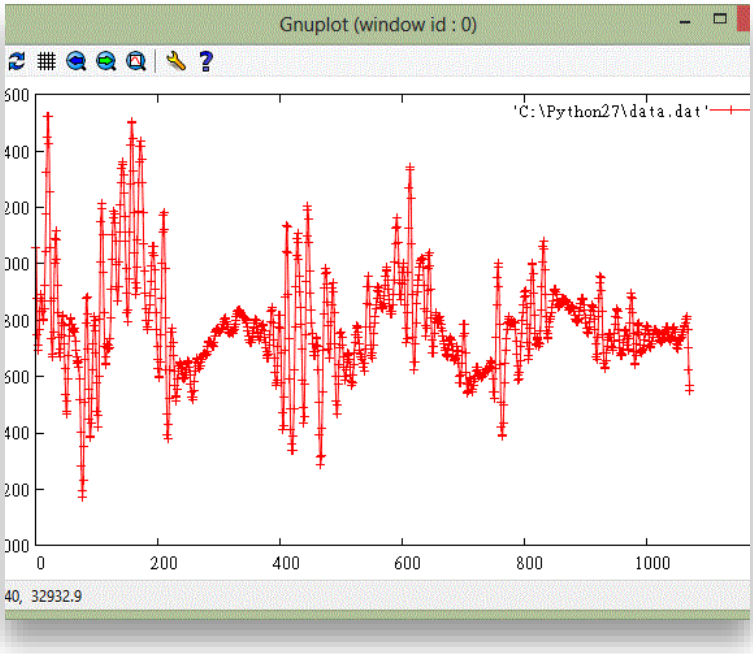

Figure 10. GNUPlot plotting data from data.dat

The image to the left shows the data saved into a data.dat file, and the image to the right illustrates that data after being plotted using GNUPlot. Here we can see most values lie between 32000 and 33000.

Pymaseis v0.2 – Processing and saving (mseed)

From here I focused on the second core feature which was to save the data in an mseed formatted file. This is a very important feature that is needed to be developed because in order to share the data collected by the TC1 we have save it in the mseed format. This functionality is important and is required for the Seismographs for Schools program because their goal is to create a network of schools that can share data with each other.

The image below shows many mseed files that were generated as the script was run. The way this works is first I create a header object in Python which contains the locational information (Geo location, station ID etc...). The header object along with the array of data which stores values from the past hour are used to create a trace object. The trace object is then saved as an mseed file. This is done with the help of the ObsPy module. The ObsPy write() method allows us to call it on a trace object and save the information (header and data) as an mseed file for example: trace.write(header, data) <- where data is an array containing all the values collected in the last 1 hour.

As per the requirements, I am saving hour long mseed files (an mseed created after every 3600 seconds) but this can be changed to save them at shorter intervals.

Along with saving mseed files, I also introduced a simple graphical user interface to start and stop the data collection. For this I used tKinter 3.3.5.

Also in this version I used the trace.plot() method to create a static plot of every trace object created before saving it as an mseed file. So altogether from saving the data collected to mseed, having a simple graphical interface and plotting regular static plots of the data, this version did the core features well. An issue I had with this version was that the windows that was display with the plot had to be manually closed before the next window could come up.

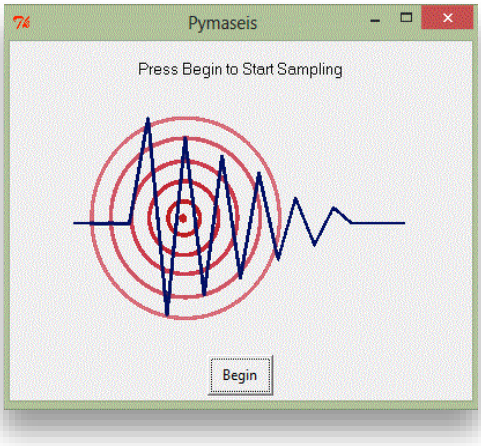

Figure 11. Pymaseis first graphical user interface made using tKinter

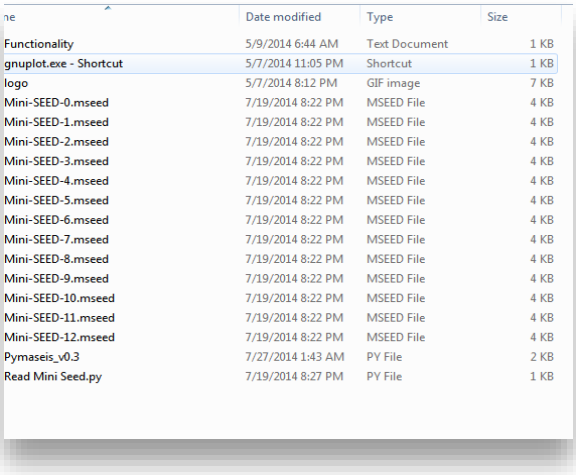

Figure 12. Pymaseis generating saving mseed files every x amount of seconds

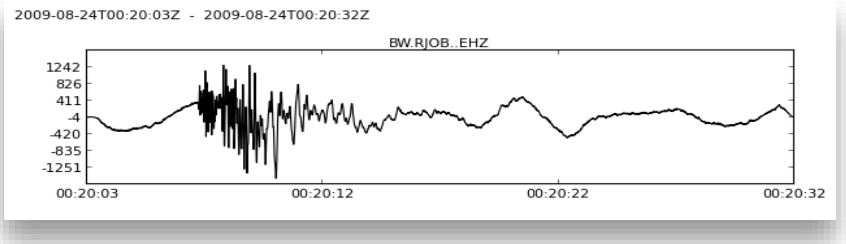

Figure 13. Plot of data displayed when trace.plot() is called after saving mseed

Pymaseis v0.3 – Displaying (Live)

In this version of Pymaseis I worked on the most visually appealing aspect of the application, the live updating plot. Initially I was saving all the data that was being read from the TC1 to an array, clearing the plot, and plotting that array every 10 milliseconds. With this method all the previous points were constantly redrawn with the new values. This proved to be very inefficient and also started to produce latency issues. Therefore to tackle this issue I looked into a way to plotting without clearing the already existing plot and plotting the new values to the existing plot. I achieved this by firstly removing the method that clears the plot (plt.cla()) then as new values were read, along with appending them to the hour long array for mseed files, I saved them in a separate array or plotting only. Hence the array for plotting always contained only the new values that need to be added to the plot. Using this array, I started plotting on the same plot which gave me the result that I wanted except each set of values were distinct and were not connected to the previous set of values. This is because in order for the joint to exist the array of new values must start from where the previous plot left off. So after understanding this, I would retrieve and save the last value of the array in a variable that was then inserted into the array before the new values were added. That way the plot would start from the last point the previous plot finished giving us a continuous plot. All latency issues were solved and data was being plotted accurately. The x axis of the plot represented in minutes in order to ensure I was accurately plotting the data to the right minute, second and millisecond, I used the datetime module to get the current time in minute, second and millisecond for each value that was read from the TC1. This was then used to plot the value. This proved to work really well and create a highly accurate plot.

The next phase in this version was to get the axis correctly representing the current hour (y axis) and minute (x axis). Using the set xticks method that was provided in the matplotlib module, I set the x axis to go from 0 – 60 with an interval of 1. Setting the y axis ticks was trickier. The aim here was to have 24 plots in 1 plot. Initially I tried using the add_subplot() provided by the matplotlib module to create 24 distinct plots in 1 figure. But this didn't work out too well. The plots were too small to view and couldn't be joined which wasted space in between. So then I decided I will stick to 1 subplot inside the figure and translate the values accordingly to create 24 plots in 1subplot. To begin with, I wrote a method that calculates the number that occurs the most in an array, and used the method to calculate the value TC1 provides when it is at rest. That means, when the seismometer is not disturbed by any seismic activity, it rests on or near a certain value and my aim was to find out what that was. Why I need it will be explained shortly. After several tests I found that the mode number was roughly 32750. With this information I understood that the graph oscillated between the values of 30750 and 34750 with 32750 being at the center of oscillation. Thus I multiplied this distance by 24 to get the upper limit of the graph. All values that come from the TC1 are between 30750 and 34750 and depending on the hour they are translated to the appropriate hour. Once I managed to get 24 plots to

show on 1 subplot, I looked into getting the ytick labels. This was required because until now the y axis was not informative. It has to display the hours, so using the datetime module I calculated the current hour and wrote a method which used this information to generate an array containing the next 24 hours along with providing their appropriate am/pm information.

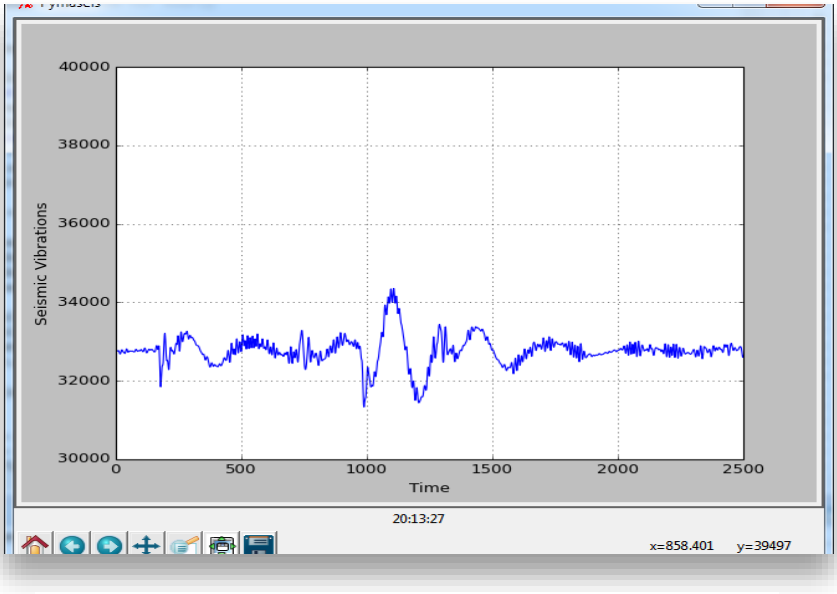

Figure 14. Pymaseis first attempt at live plotting

The above image shows the first version of live plotting where I was plotting the whole array repeatedly as it received new values. Note the x, y axis only represent the values that were coming in, and not representing the hour and minute that they were read from the seismometer.

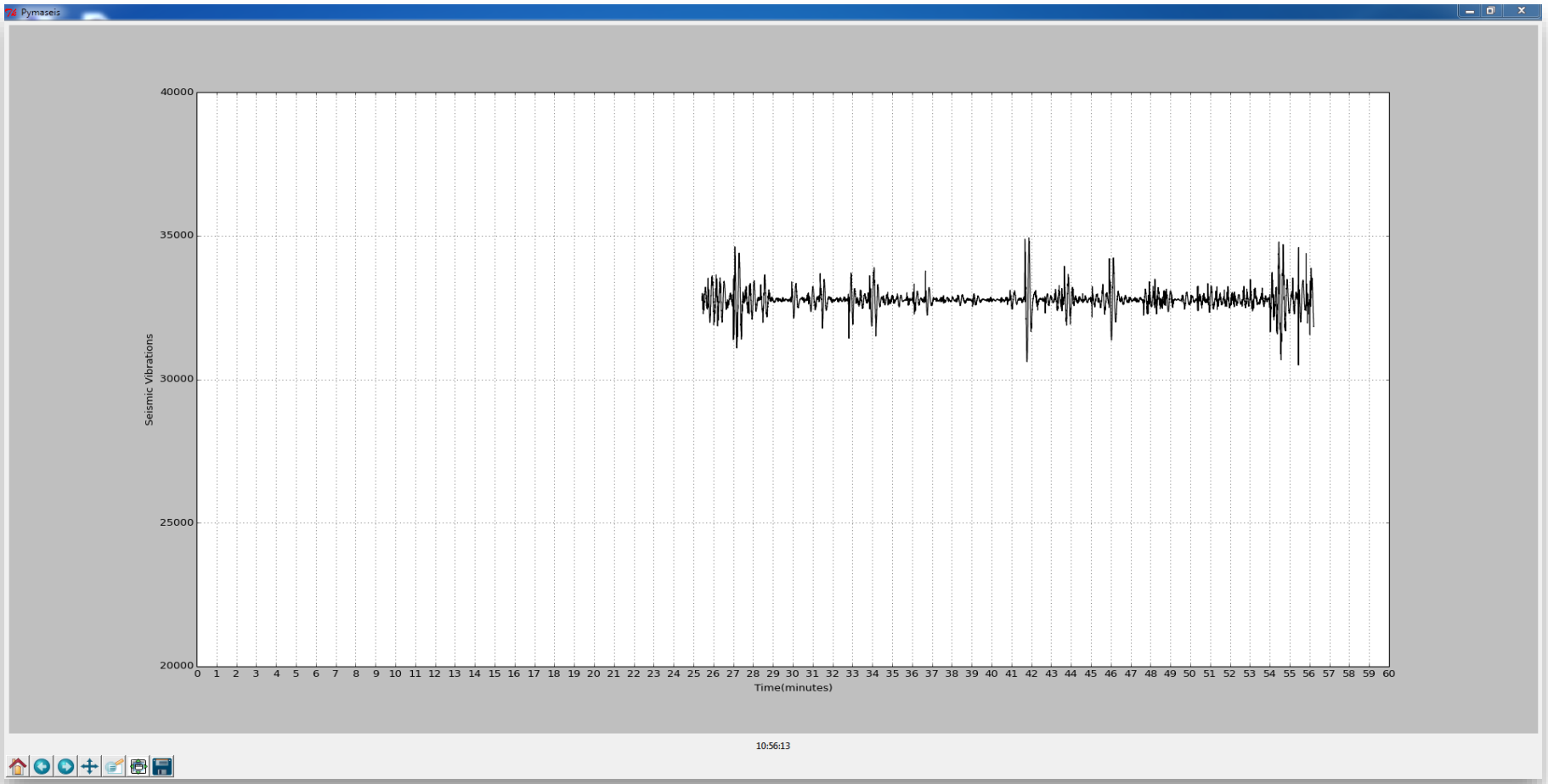

Figure 15. Pymaseis x axis corrected and accurate plotting in x axis implemented

Moving on from there as explained earlier, once I managed to get the live plotting by plotting only the new values in the array, I worked on making the x axis more meaningful. As you can see the x axis now goes from 0 to 60 with intervals of 1.

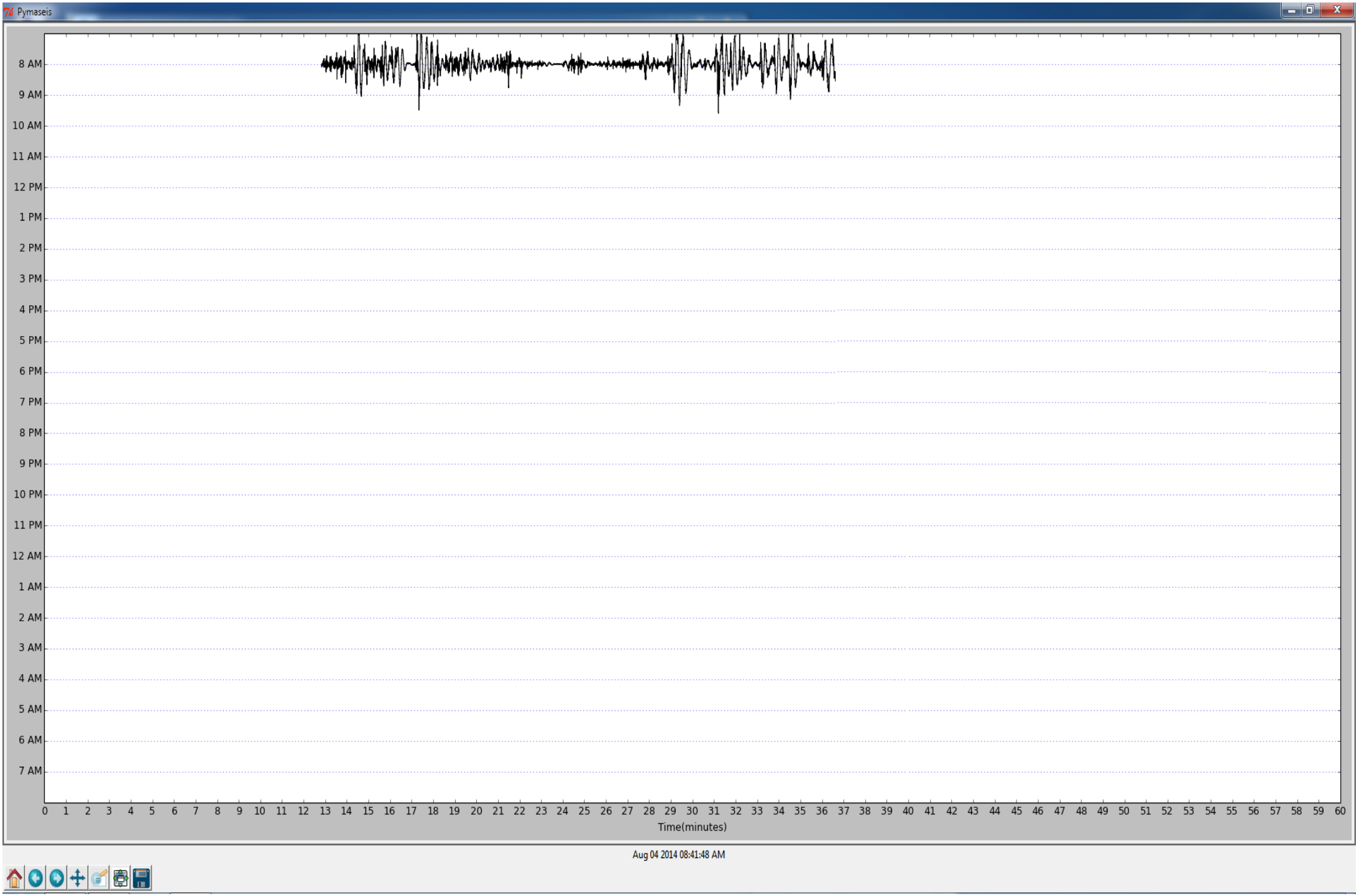

Figure 16. Pymaseis y axis corrected and hourly plotting achieved with total of 24 hours

Finally the last thing I worked on this version was getting the y axis to display the hours. This has been explained above. Currently this is as far as I have gotten with my development of the application – plotting live data to the exact millisecond (x) and hour (y). Note once the top plot reaches 60 minutes, it automatically starts at potting on the start of the next hour. This was demonstrated in the live demo at the mid-year presentation.

Challenges

I faced various challenges over the course of the development so far. Some of these are described here.

Version Control System

For a large part of the project so far I didn't use a version control system, and slowly I found myself struggling to find the right Python script among all the test scripts I had made. Also due to not naming them correctly, lost track of what script contained which functionality. I tried to get each core function working independently before joining them. So due to not being able to effectively keep track of the files I was using, I ran into duplication issues and was also unable to keep track of changes. That's when I realized I need to use a VCS to manage my project and started using Git to organize and keep track of my project progress.

Inefficient Plotting

This has been explained before, and I will briefly repeat it here. I was using 1 array to hold all the values that were being read from the TC1, and this array was used to create the plot. All new values were appended to this array and the plot was cleared and this array holding all the old and the new values was plotted. This was done every 10 milliseconds. This style of plotting quickly caused latency issues between the events that took place and captured by the TC1 and the actual time they were displayed on the plot. The plot would also freeze after a while. To fix this, I created a second array that only stored the new values, and plotted this array over the previous plot.

Multiple Subplots

This challenged I faced here has been briefly explained before and I will go into detail about it here. Matplotlib allows us to create sub-plots within 1 figure. That means you can create multiple separate plots within one figure. However as you can see in the image below, when I created 24 sub plots the outcome was not pleasing. The data plotted couldn't be accurately seen as the plots were too small and too close to each other. Hence I decided to move away from making multiple subplots to creating using 1 plot and translating the values according to the correct hour. This proved to be more effective and helped create a user interface that wasn't cluttered.

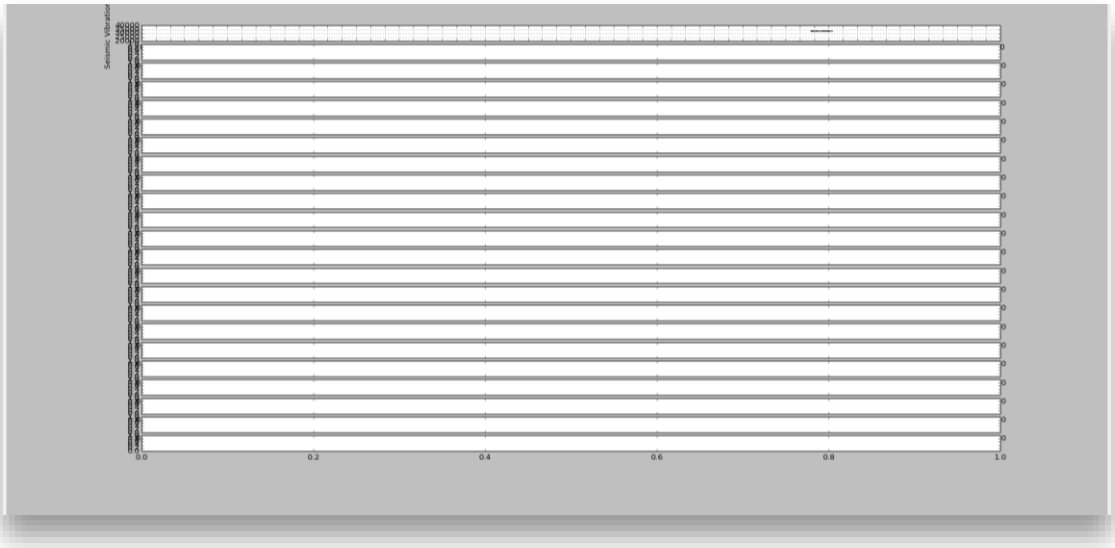

Figure 17. Using add_subplots() to create 24 subplots in one windows

Data Reading Frequency Issue

This is a recent issue that I faced which started last week, the image directly below shows the output that was being potted. The problem here was I was receiving values from the TC1 that when plotted resulted in plots as shown below. For example, the second image shows that in an array of values ranging from 87000 to 88000, an unexpected 919 value gets saved in the array. This causes the plot to draw a line from the current value to the 919 value. This results in distorted plots. Initially I thought this could have been a hardware problem, however after consulting my industry supervisor Dr. Kasper van Wijk I was told that the TC1 pushes only 18 values per second. Meaning the USB cable/and serial port will only hold 18 values per second and by trying to read more than 18, I get inaccurate values that when plotted result in distorted plots. Hence I fixed the rate at which I was reading values to address this issue.

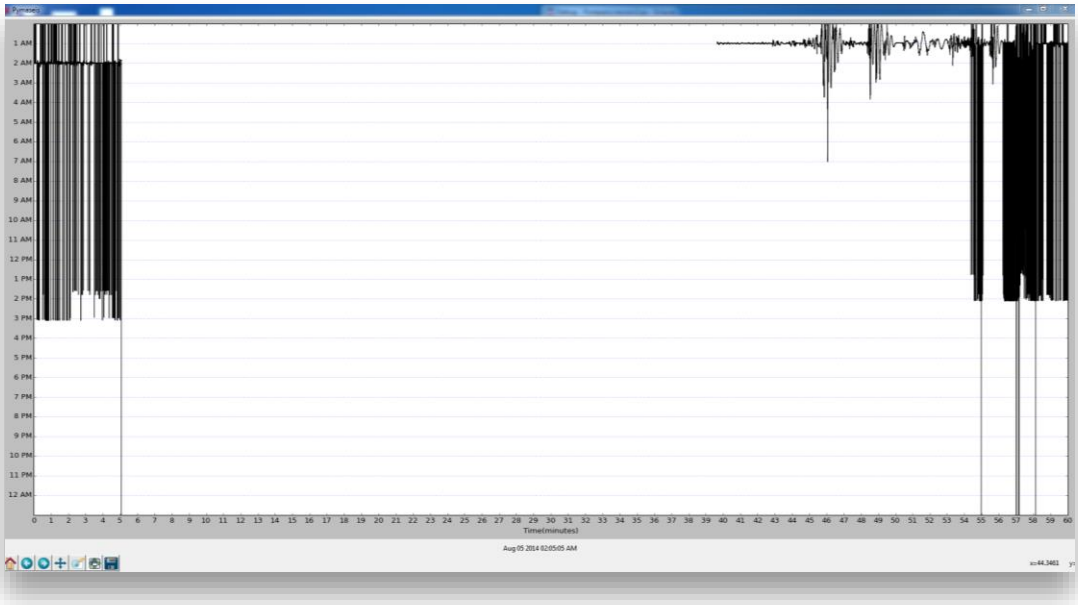

Figure 18. Distorted plot

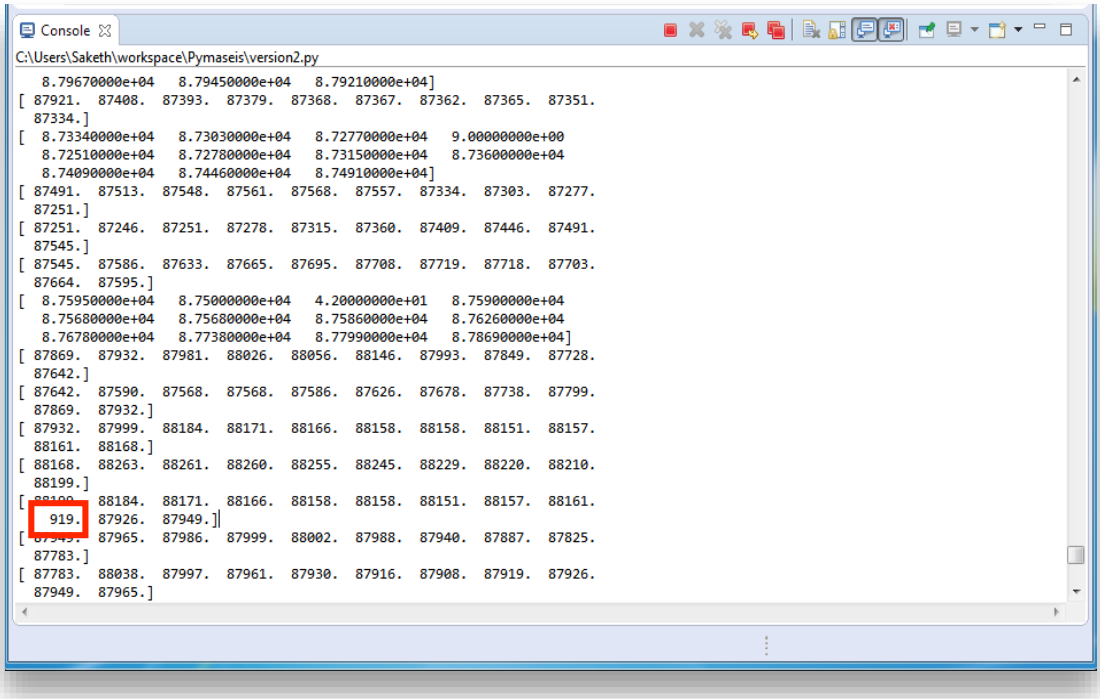

Figure 19. Cause of distorted plots found to be incorrect value in plotting array

Future Implementation and Considerations

The following is the work that will be carried out over the next 11 weeks before the semester ends.

Header information

Till now the mseed files that are being created contain only the array of information (data) but don't contain any header information (locational information). In order to do this I need to save the information entered by the user into a header object. To facilitate this I have created a simple GUI to allow the user to enter this information. This is done using tKinter 3.3.5 and looks like the image below. Currently the GUI is developed, but the backend still need to be written. Note the Latitude, Longitude and Elevation fields will be auto filled.

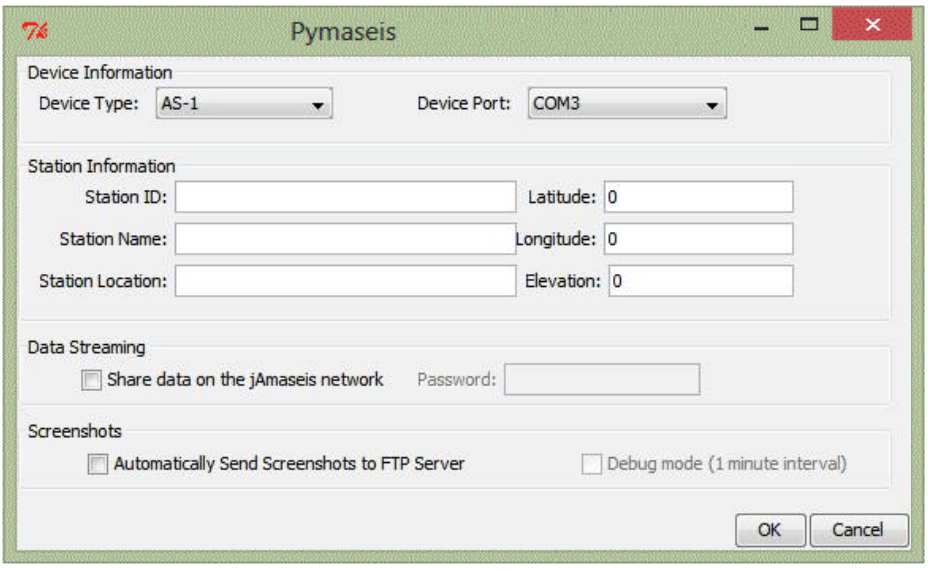

Figure 20. GUI for collecting locational information before plotting begins

Matplotlib Labels

One of the very important features that I will be looking into to making the user interface more targeted towards students and student learning, is to provide visual cues such as popups and labels that will display when a prerecorded seismic behavior is noticed. Labels are a feature provided by Matplotlib and I will be looking to incorporated them effectively to align with the overall learning goal of application. Examples of how I will be using pointers and labels are shown below.

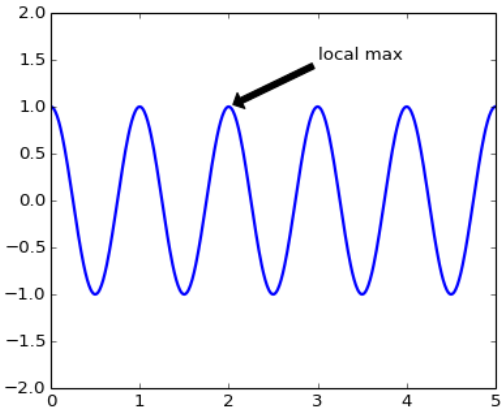

Figure 21. Matplotlib Pointers

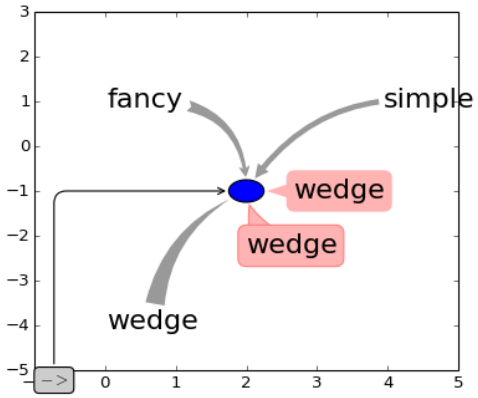

Figure 22. Matplotlib Labels

Remote functionality

This is the functionality that allows the application to post mseed files and screenshots to the Seedlink server and also retrieve and plot mseed files from the Seedlink server. This is phase two of the application and is very important because it is a requirement from IRIS Seismographs for Schools program where each school should be able to exchange earthquake data.

Threading

Till now the data reading model works like a sampler where it reads for a while then goes and plots that data, then comes back and reads and so on. Although this is working efficiently, the ideal situation would be to break the reading and plotting functionality and run them on separate threads to enhance the overall efficiency of the application. This will be looked into over the next few weeks.

Bibliography

[1] “Paradigm Advanced Science for everyone”. http://www.pdgm.com/solutions/seismic-processing-and-imaging/, August 2014

[2] jAmaseis: Seismology Software Meeting the Needs of Educators. 2009

[3] Larry Cochrane. Winquake version 2.8 documentation, October 2009. http://psn.quake.net/software/wq28doc.pdf.

[4] IRIS. Iris - education and outreach. http://www.iris.edu/hq/programs/education_and_outreach. Accessed August 3, 2014.

[5] IRIS. Iris - incorporated research institutions for seismology. http://www.iris.edu/hq/. Accessed August 3, 2014.

[6] Ben Coleman and Joseph Gerencher. A software system for real-time sharing of seismic data in educational environments. June 2008.

[7] Joseph Gerencher and Ronald Jackson.Classroom utilization of a multi-axis lehman seismograph system. Journal of Geological Education, 1991.

[8] Joseph Gerencher and Michael Sands. Online near-real-time seismic system for the classroom. Journal of Geological Education, 2004.