Geiger Counter Lab Analysis Software

Raymond Ehlers* and Sam Kohn†

Columbia University, Columbia College, Class of 2013

Abstract

The Geiger Counter analysis Software, written in Python, provides complete analysis from saved wave file to histograms of relevant quantities. This manual describes usage of Audacity for audio recording, as well as the usage of the Geiger Counter analysis software. In addition, this manual documents the code used in the software itself, allowing for potential future modification. This software was written for the E. K. A. Advanced Physics Lab at Columbia University.

*Electronic address: rje2114@columbia.edu

†Electronic address: sjk2163@columbia.edu

Figure 1: The Audacity upper interface

I. RECORDING DATA

Based on the current experimental setup, data can be recorded through a simple 3.5 mm audio jack. Be certain to plug in the USB connector from the voltage controller in addition to the 3.5 mm plug into the microphone port of the computer. This may already be completed. See the lab manual for specific instructions on setup of the experiment.

A. Audacity

Audacity (`http://audacity.sourceforge.net/`) is a free, open source, cross platform tool for audio recording and editing. While in principle any any audio recording software would be sufficient, Audacity has a strong set of features, as well as good flexibility and usability, such that it good choice for this lab. To record in Audacity, it is as simple as pressing the large, red record button near the middle of the upper interface, which can be seen in figure 1. However, a number of options must be setup to record the data correctly, which will be described below.

B. Recording Data Using Audacity

To process the recorded data correctly, the sampling characteristics must be set before recording. To specify these settings, open the Audacity sampling preferences under `Edit` $->$ `Preferences` $->$ `Quality` (or `Audacity` $->$ `Preferences` $->$ `Quality` on Mac OS X), and set the Default Sample Rate(recommended value of 96000 Hz) and Default Sample Format (this must be 16-bit).

Before beginning to take data, the correct audio input must be established. This is set in the two drop-down menus to the right of the microphone icon. The drop-down menu is located in the lower toolbar of the upper part of the Audacity, and can be seen in figure 1. The first drop-down menu, with the value of "Integrated Microphone Array" in the figure,

determines the source of the audio. Finding the appropriate source is described in the lab manual. In short, unless you are very familiar with the computer, it is effectively trial and error. The drop-down menu to the farthest right must be set to 1 (Mono) Input Channel. The voltage controller only outputs mono pulses, and an added signal may cause problems in the analysis software, so it is critical to ensure that Mono is selected.

1. Timing the recording

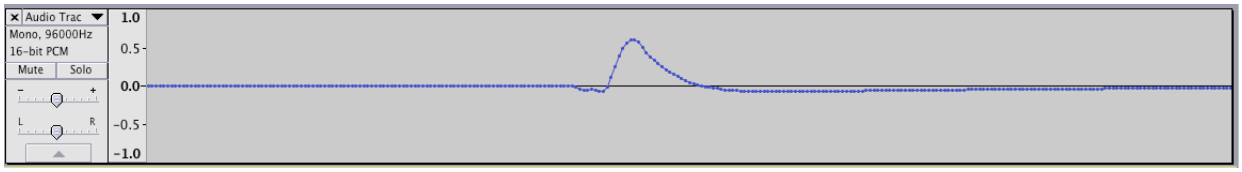
While recording can be completed manual in Audacity, for this lab, it is convenient to set Audacity to record for a specified length of time. This will allow the student to avoid a need to edit the audio file after recording. This feature is accessible under `Transport` $->$ `Timer Record`... See the lab manual to determine the appropriate duration to record for different samples. Clicking "OK" will begin the recording.

2. Exporting the Audio Data

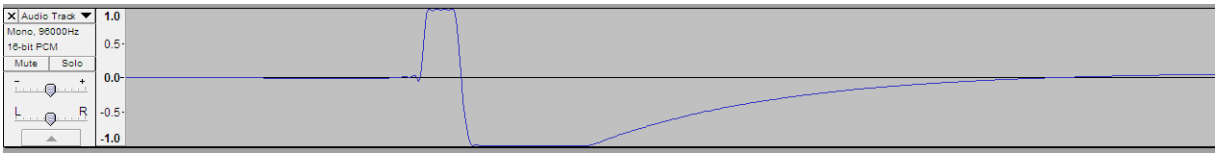
After the data has been collected, it can be saved in an Audacity file, but this will not be useful for analysis. Instead, the data must be exported as a 16-bit signed wave file. To export the data, select `File` $->$ `Export`. In the dialog box, be certain to change the "Save as type" drop-down menu to "WAV (Microsoft) signed 16 bit PCM." After this has been selected, choose the location and file name for your data, and save it. The data file is now ready to be analyzed.

3. The Waveform in Audacity

One benefit of Audacity is that it illustrates the waveform of the data, allowing users to investigate the shape of the waveform, similar to an oscilloscope. Based on this display, a number of properties may be determined. The shape of the waveform will be determined by the specific Geiger counter, as well as the performance of the sound card recording the data. An example of the same pulse in recorded by two different sound cards can be seen in figures 2a and 2b. In figure 2a, the waveform is not clipped (truncated in amplitude) in the recording, so all of the features of the waveform can clearly be seen. In figure 2b, the



(a) Unclipped waveform



(b) Clipped wavform

Figure 2: Waveforms in Audacity

amplitude of waveform is clipped, such that some of the features cannot be seen. However, either type of waveform should be acceptable for analysis. It is important to note the relative scale on the left of both images as a check of the threshold value. The analysis software depends on a user determined threshold for the necessary height of a pulse to be considered a count by the software. The Audacity scale is an arbitrary scale from 0 to 1, so the threshold can be calculated by multiply the fraction of the arbitrary height by 32767 (the largest possible number in a signed 16-bit integer). However, this scale is not particularly accurate, such that the threshold should be determined in the analysis software, rather than based on the Audacity scale. This process is described in section II A 1. Despite the inaccuracy of the Audacity scale, it is still useful as a sanity check to ensure that the threshold is reasonable.

II. ANALYSIS SOFTWARE

First, software usage will be discussed, followed by a more technical discussion of the code. This analysis code was written by Raymond Ehlers and Sam Kohn, CC 2013. If major issues are encountered, feel free to contact either of us.

A. Analysis Functionality

The analysis software can be seen in figure 3. It can be started by double clicking on the shortcut `GC` *Analysis* in the main directory in Windows (if that does not work, double click

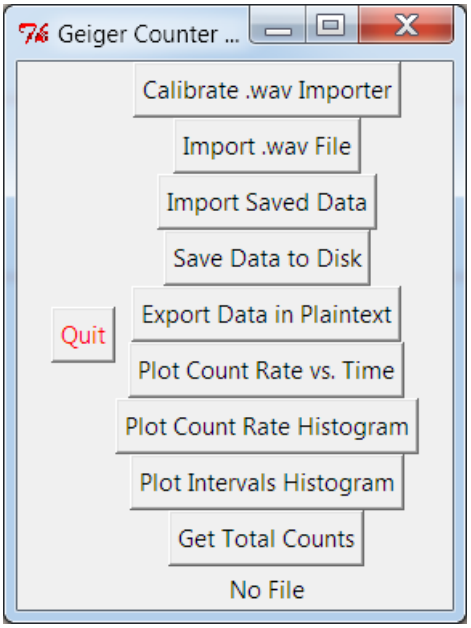


Figure 3: The Geiger Counter analysis software

on `GUI.py`. For more information, see section II B, including running in a python interpreter, or on a Mac or Linux system). There are a large number of functions, accessible through buttons on the right. The button on the left quits the program. There is a status indicator at the bottom that dynamically updates based on the progress in analyzing the wave file. The following operations describe the operation of the analysis software.

1. *Calibrate .wav importer*

The input for this function is a wave file taken with a source in the Geiger counter. Further, it must included at least one full count within the first 5 seconds of the file. This will return a display of the maximum pulse height, along with the values surrounding it. The student should calculate 90% of the pulse height. This is the threshold. If any of the variables in the experiment change (ie. voltage, sample rate, etc), this threshold should be recalculated. Therefore, it should generally be repeated at the beginning of each analysis of a wave file. If the file is imported from saved data, the threshold was already set when the analysis was completed originally and hence the threshold determined here is not necessary.

2. Import .wav File

The input for this function is a wave file taken with a source in the Geiger counter, for a length determined by information in the lab manual. After the file is chosen, the threshold value, a number between 0 and 32767, which was determined in section II A 1, must be input. The software will analyze the entire wave file, producing a data set. Note: this can take quite a bit of time and processing power (on our modern laptops, we were processing about 1 second of data in 1 second). Therefore, it is important to be patient. The progress is displayed at the bottom of the window.

3. Import Saved Data

If analysis has already been completed and the data set was saved, the data set can be imported through this function. Because it has already been analyzed, the import should be instantaneous. Consequently, it is highly recommended to save each data set after it has been analyzed initially.

4. Save Data to Disk

Once the initial import analysis is complete, this function allow a student to save the analyzed data to disk. This function is extremely useful, as it allows the student to re-import the data set in the future without having to rerun the long initial analysis function. Note: This function exports in a binary format. For an export of the raw data, see section II A 5.

5. Export Data in Plaintext

This button exports the current data set into a CSV (comma-separated values) file for use with other data analysis software. The format is one line of header, followed by each count time on its own line. The only input is the save location, accessible through the file chooser dialog window. Data cannot be re-imported into the analysis software based on this file format.

6. Plot Count Rate vs Time

This button plots a graph of the count rate as a function of time for the current data set. The only input is the sample size, which represents the duration over which the count rate is calculated for each point on the graph. For example, for a 100 s data set and 1 s sample size, the graph will have 100 points, each identifying the average count rate over the 1 s duration beginning at that point.

7. Plot Count Rate Histogram

This button plots a histogram of the count rates of the current data set. The two inputs are sample size and number of bins in the histogram. The sample size represents the duration over which the count rate is calculated for each element of the histogram. For example, for a 100 s data set and 1 s sample size, the histogram will contain 100 elements, each containing the average count rate over a particular 1 s segment of the data set.

8. Plot Intervals Histogram

This button plots a histogram of the time interval between consecutive counts in the current data set—so if there are N total counts, there will be $N-1$ elements in the histogram. The only input is the number of bins in the histogram.

9. Get Total Counts

This button opens a window displaying the total number of counts in the current data set.

B. Further Documentation

This section assumes some familiarity with programming and computers. If the student is not familiar, section II A should be sufficient.

The analysis software can be found on Github at `https://github.com/musicalrunner/Geiger-Counter`.

The analysis software is written in Python, and is dependent on a number of built in Python modules, such as Tkinter, CSV and Pickle. In addition, the software depends on the third party MatPlotLib and SciPy modules. For Windows, the easiest way to install these modules is with a third party Python installation, such as `Python(x,y)` (`http://code.google.com/p/pythonxy/`). For Mac or Linux, it is best to utilize pip, utilizing the commands `pip install matplotlib` and `pip install scipy`. Pip can be installed following the instructions listed at `http://www.pip-installer.org/en/latest/`. While Windows users could use pip, or *nix users could utilize a third party python installation, the approaches recommended above have yielded the best results for us.

1. Code structure

The code structure centers around three files, `GUI.py`, `PlotData.py` and `analyzeData.py`. GUI is the front end, which utilizes the analyzeData Python module and the PlotData graphing module. The analyzeData module implements the back-end functionality through an object called DataSet. The methods and members of the both files can be seen in `./documentation/GUI.html` and `./documentation/analyzeData.html`. These files are generated from documentation present in the source. The comments and documentation are quite thorough, and together form better documentation than could be explained otherwise.

2. File Output and Formats

As alluded to above there are a few different ways to export data from the analysis software. The Save Data to Disk function, described in section II A 4, saves the object using the Python Pickle module. The Pickle module creates a pseudo-binary file, which can be read in plain text, but is not a format conducive to parsing. Further, if the members of the DataSet object are changed, it cannot be guaranteed that Pickle will correctly re-expand the DataSet object to a functional state. In summary, you can edit the DataSet object members, but it must be done carefully with lots of testing for backwards compatibility. Despite these potential difficulties, Pickle is preferred method, as it greatly simplifies the I/O process.

If one desires to do analysis using another analysis package on the counts without such overhead, the data can be exported into a CSV, return separated file using the Export Data

in Plaintext function described in section II A 5. The file format is described in that section. As noted, it cannot be re-imported into the GC Analysis software. It should be fairly trivial to implement the import process for the count times, but one may be left without necessary values for a full DataSet object, such as sample rate. It is much simpler (and already implemented) to use Save Data to Disk instead.

3. Building Executables

There is currently code to build executables in Windows and Mac OS X. Building packages in Windows in fairly fragile and complicated. The Windows build utilizes `cx_freeze`, and should work fine for any Python 2.7 system, although it cannot be guaranteed. `Python(x, y)` includes Python 2.7 and `cx_freeze` as part of the default distribution. The package is built with `python cx_setup.py build` . Options can be changed in `cx_setup.py` if desired. If everything is built correctly, the executable should be under `./build/exe.win32 − 2.7/GUI.exe` . The `GC Analysis` shortcut should continue to work, assuming a 32-bit install of Python 2.7 (which can be installed on a 64-bit system).

The Mac package is built with `py2app`, which may be installed with pip. The package can be generated with by starting in the `./py2app/` folder, running `py2applet −−make-setup GUI.py`, followed by `python setup.py py2app` . The documentation can be found at `http://svn.pythonmac.org/py2app/py2app/trunk/doc/index.html#tutorial`. The application can be found in `./py2app/dist/` .