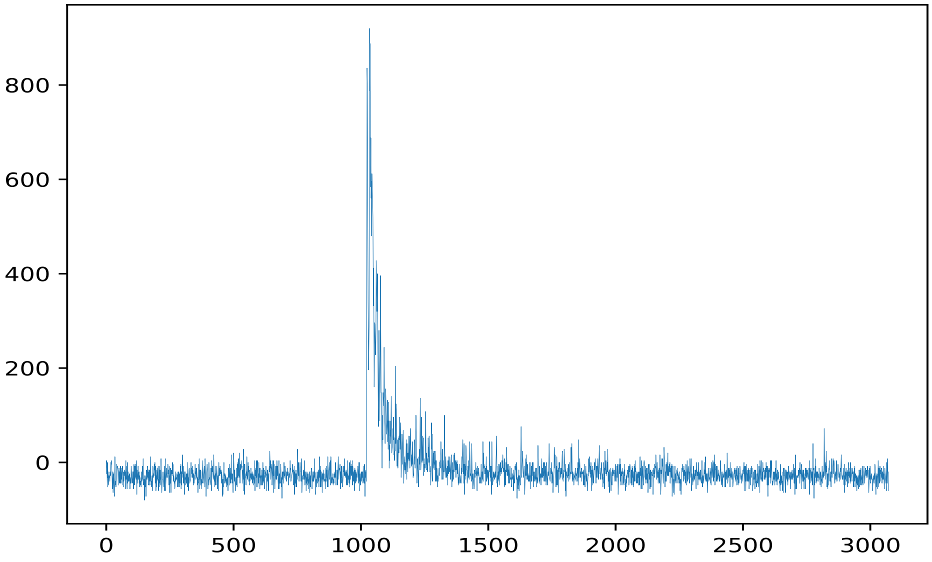
# Project 5: Data Visualization and Analysis

CS 1410

## Background

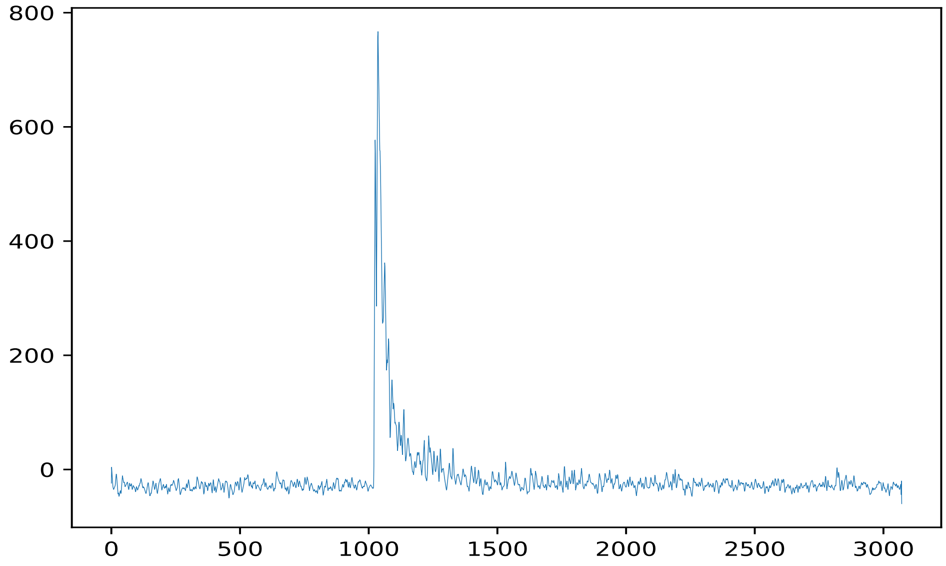
This project is based on actual work done for the Department of Homeland Security. The goal is to detect the presence of dangerous materials, such as in a point of entry at an airport or other location. All items pass through a scanner which reports certain radioactive emissions as voltage spikes. The software takes the digitized voltage measurements and examines them for such spikes, which we call "pulses". Here is a graph of one of the data files provided by the scanner:



Because the data is quite jagged, we first "smooth" it before analyzing it. We will form a *new sequence* of numbers obtained by replacing every data value, starting with the 4th entry and ending with the 4th from the last entry, with a *weighted average* of itself and the six points around it, using the following formula (the current point to be smoothed is $y_i$):

$$(y_{i-3} + 2y_{i-2} + 3y_{i-1} + 3y_i + 3y_{i+1} + 2y_{i+2} + y_{i+3}) \,//\, 15$$

So, in our new sequence of smoothed data, the center point of every interval of seven points, $y_i$, is replaced by applying the formula above. The first and last 3 data values are left unchanged from the original raw data, since there aren't a sufficient number of points before or after them to use the smoothing formula. The original data array is left unchanged. Here is a plot of the smoothed data:



It is better to use the smoothed data for detecting pulses.

## Requirements

In this project, you will process all of the files in the current folder with a ".dat" suffix in the file name. These files are found on Canvas in the Project 5folder.

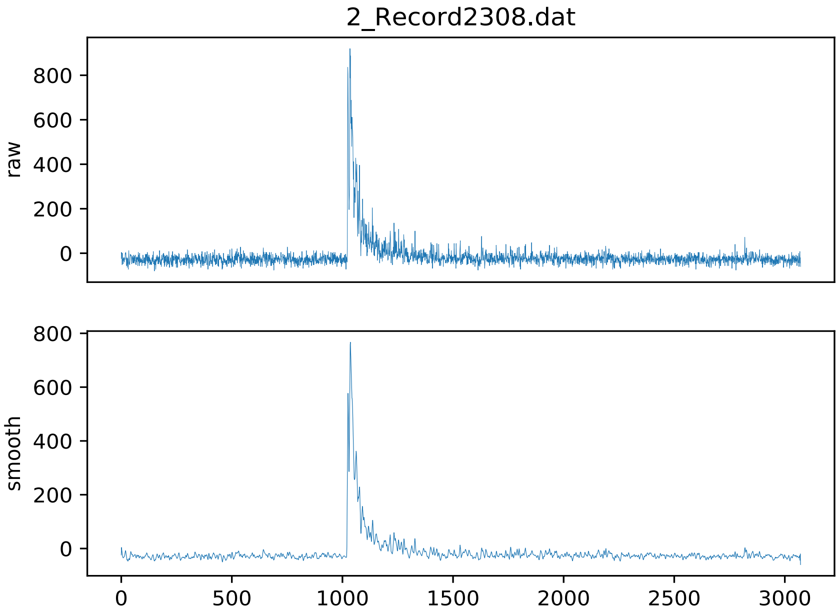
For each such file, you will:

1. Read the data into an array of integers (using `array.array` or `numpy.loadtxt`)
2. Smooth the data into another array of integers
3. Find the beginning of each pulse by examining the smoothed data
4. Compute the "area" of each pulse using the original, unsmoothed ("raw") data
5. Graph both the raw and smoothed data on the same plot, each with their own axis

You detect a pulse by looking for a "rise" over three consecutive points, $y_i, y_{i+1}, y_{i+2}$. The rise is the difference $y_{i+2} - y_i$. If the rise exceeds a certain "voltage threshold" (**vt** = 100), then a pulse begins at position $i$. Record this position. After finding a pulse start position, advance through the data starting at $y_{i+2}$ until the samples start to decrease before looking for the next pulse.

To compute the area of a pulse, you simply add the 50 raw $y$-values starting at the beginning of the pulse, or up to but not including the start of the next pulse, whichever comes first.

For each file, plot both sets of data in a single plot and save it to a file. For example, one of the files is named *2_Record2308.dat*, so you will store the following plot in the file *2_Record2308.pdf*:



Do **not** hard code the file names (see **main** below). You will also write the original file name, pulse start positions, and corresponding areas to the file *2_Record2308.out*. If your computations are correct, the following will appear in *2_Record2308.out*:

```
2_Record2308.dat:
Pulse 1: 1019 (3540)
Pulse 2: 1030 (18852)
```

The area for each pulse appears in parentheses after its start position. For the pulse beginning at (one-based for humans) position 1020 (position 1019 in the zero-based array, of course), you sum the voltages at raw data positions 1019 through 1029 in the array.

## Implementation Notes

Use **array.array** or **numpy.ndarray** (preferred) to hold the voltages.

Use **subplot** to plot both raw and smoothed data in a single plot. Make sure you include titles and labels as shown in the sample plot above. Use the **savefig** method in **pyplot** to save your plots to PDF files.

Use the following main program. All the work is done in **analyze**.

```
def main():
   for fname in glob.glob('*.dat'):
        analyze(fname)
```

Submit your source file (name it **dataplot.py**) and all of the **.pdf** and **.out** files. Your program should be able to handle an arbitrary number of **.dat** files.

**Note**: This is a short program (50-ish lines), but you only have a little over 1 week!

## FAQs

**Q**. Should I use **numpy.ndarray** or **array.array**?
**A**. Well, **array.array** is easier to learn, but **numpy.loadtxt( )** can read all the data at once without you writing a loop. **numpy.ndarray** is faster, but it won't matter in this small project. Also, you have to *install* **numpy**, whereas the **array** module comes with Python. You pick.

**Q**. Anything to watch out for in this assignment?
**A**. Yes! When you are detecting pulses, and more importantly, when you are calculating areas, you must make sure you don't accidently go off the right end of the array. Watch your boundaries.