EE1103 Numerical Methods

Quiz 2a

October 26th 2016

Group 7

Rajat Vadiraj Dwaraknath - EE16B033

Nithesh N Hariharan - EE16B144

Chopra Dhruv Naveen - EE16B107

We first import numpy and matplotlib using pylab, and increase the size of the plots.

In [1]:

```
# Importing matplotlib and numpy directly
# Python 2.7
%pylab inline
rcParams['figure.figsize'] = 15, 12 # Increase size of plots
rcParams['font.size'] = 15 # Increase font size in plots
```

```
Populating the interactive namespace from numpy and matplotlib
```

Removing noise from the data

The noise in the data was removed by applying a convolution filter on the data. The kernel used was a gaussian, as opposed to a constant function(which is equivalent to a moving average filter) because it had a smaller impact on the height of the peaks in some of the data. The gaussian used was a normalized gaussian with unit standard deviation, varying from -3 to 3, with a total length of 21 points, resulting in 10 points on either side of the central point. The filter yielded good results as can be seen in the plots that follow. One must note that in the case of the SS1 data, an extremely high variation of the signal was seen at a particular interval of time, which could not be removed by the filter. This indicates that the source of this variation is not random noise, but some sort of interfering signal.

The plots

Note that these plots will appear only if the notebook is run after running make in the local directory.

The function used to plot the data is defined below:

In [2]:

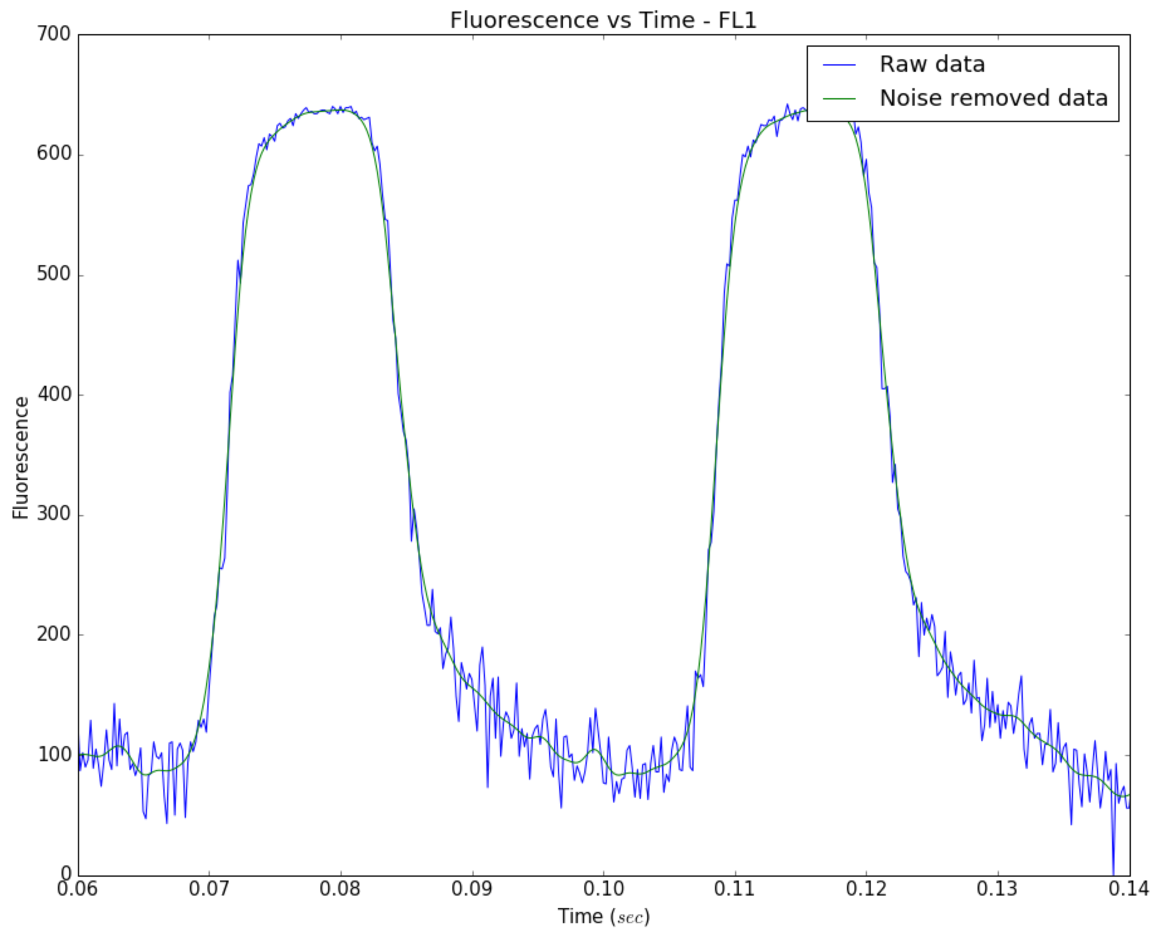
```
plt = matplotlib.pyplot
def plotData(s):
    """
    Python code to plot the data in a text file.
    The first column is assumed to contain the X values and the
    successive columns are assumed to contain the values of
    different data sets at the corresponding X values. These data
    sets are plotted individually. If the first three lines of the
    file begin with #, they are assumed to contain the title, the
    x and y labels(comma separated) and legend(comma separated)
    respectively.
    """
    # Open the file
    fo = open(s, 'r')

    lines = fo.read().split("\n")

    title = s+" X vs Y"
    labels = ['X','Y']
    legend = []

    if lines[0][0] == '#' and lines[1][0] == '#' and lines[2][0] == '#':
        title = lines[0][1:].strip()
        labels = [i.strip() for i in lines[1][1:].strip().split(',')]
        legend = [i.strip() for i in lines[2][1:].strip().split(',')]

    if len(labels) != 2:
        labels = ['X','Y']


    lines = [x.strip() for x in lines if len(x) != 0 and x[0] != '#']

    # Close the file
    fo.close()

    # List of time instants
    t = []

    # List of state vectors
    states = []

    for i in lines:

        # Separate the columns
        values = i.split(" ")

        # First column contains time instants
        t.append(float(values[0]))

        # Add each successive column as a new state in the state vector
        states.append([float(i) for i in values[1:]])

    # Plot the data
    plt.plot(t,states)
    plt.xlabel(labels[0])
    plt.ylabel(labels[1])
    plt.title(title)
    plt.legend(legend)
```

Following are the plots which compare the raw and filtered data.

FL1

In [3]:

```
plotData("cleanFL1.txt")
xlim(0.06, 0.14)
show()
```

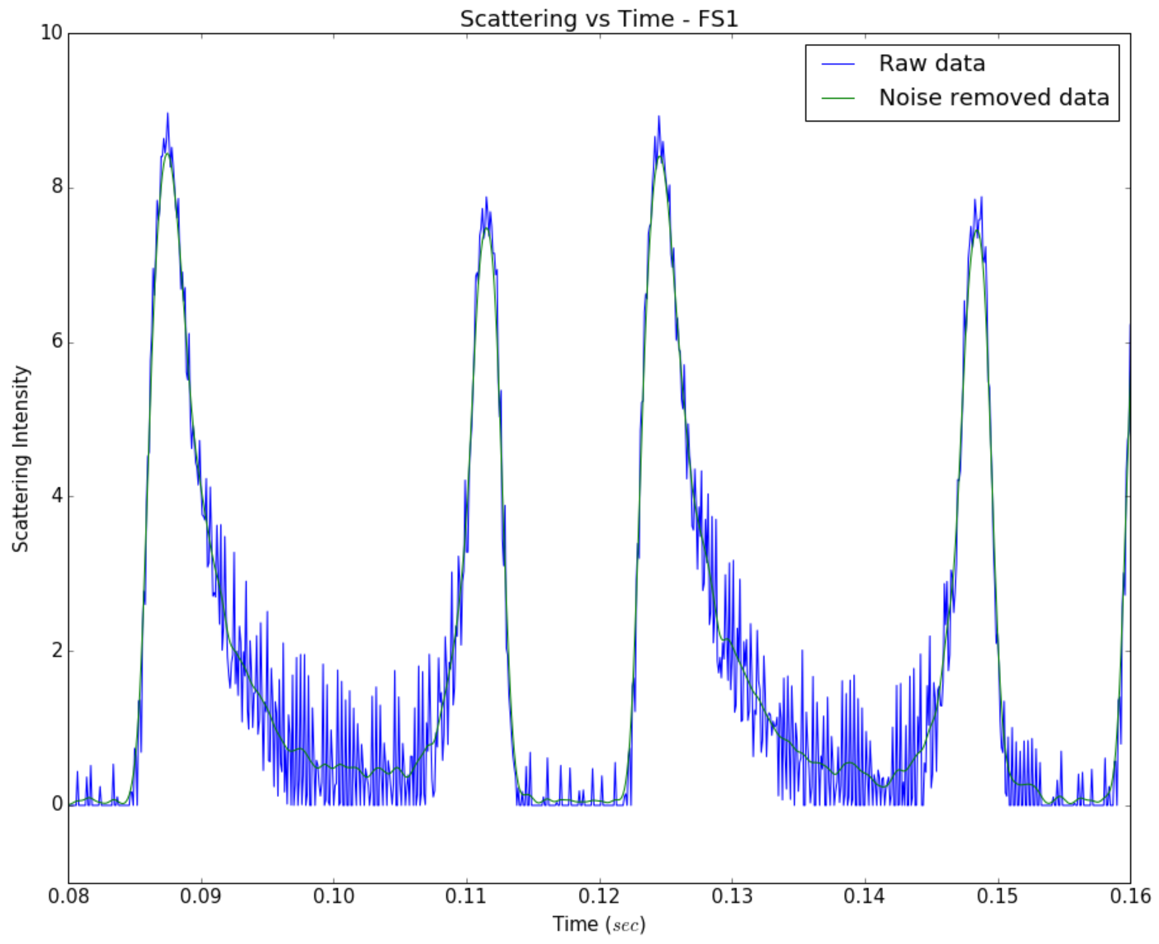


We see from the above plot that the noise has been effectively removed without reducing the heights of the peaks appreciably.

FS1

In [4]:

```
plotData("cleanFS1.txt")
xlim(0.08,0.16)
ylim(-1,10)
show()
```

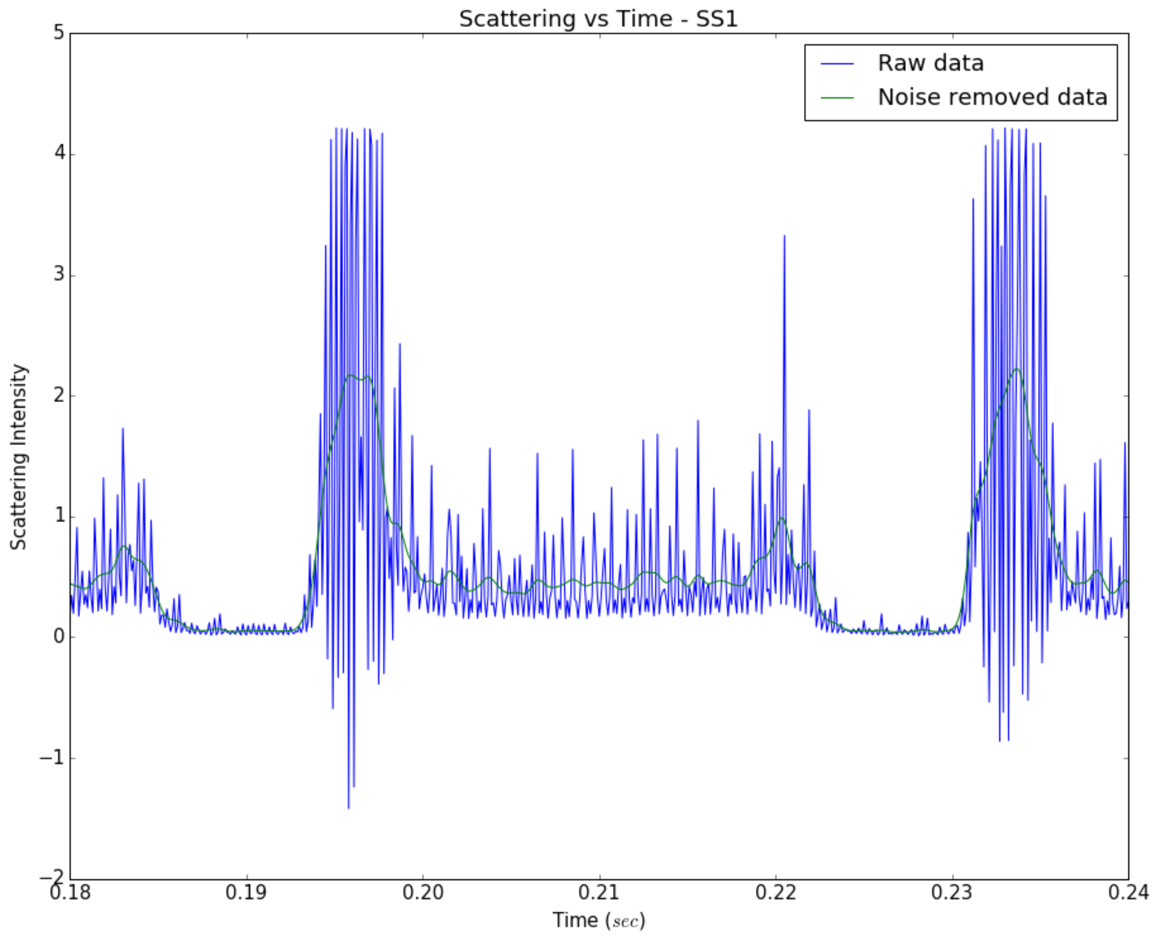


The noise in the SS1 data was also removed effectively.

SS1

In [5]:

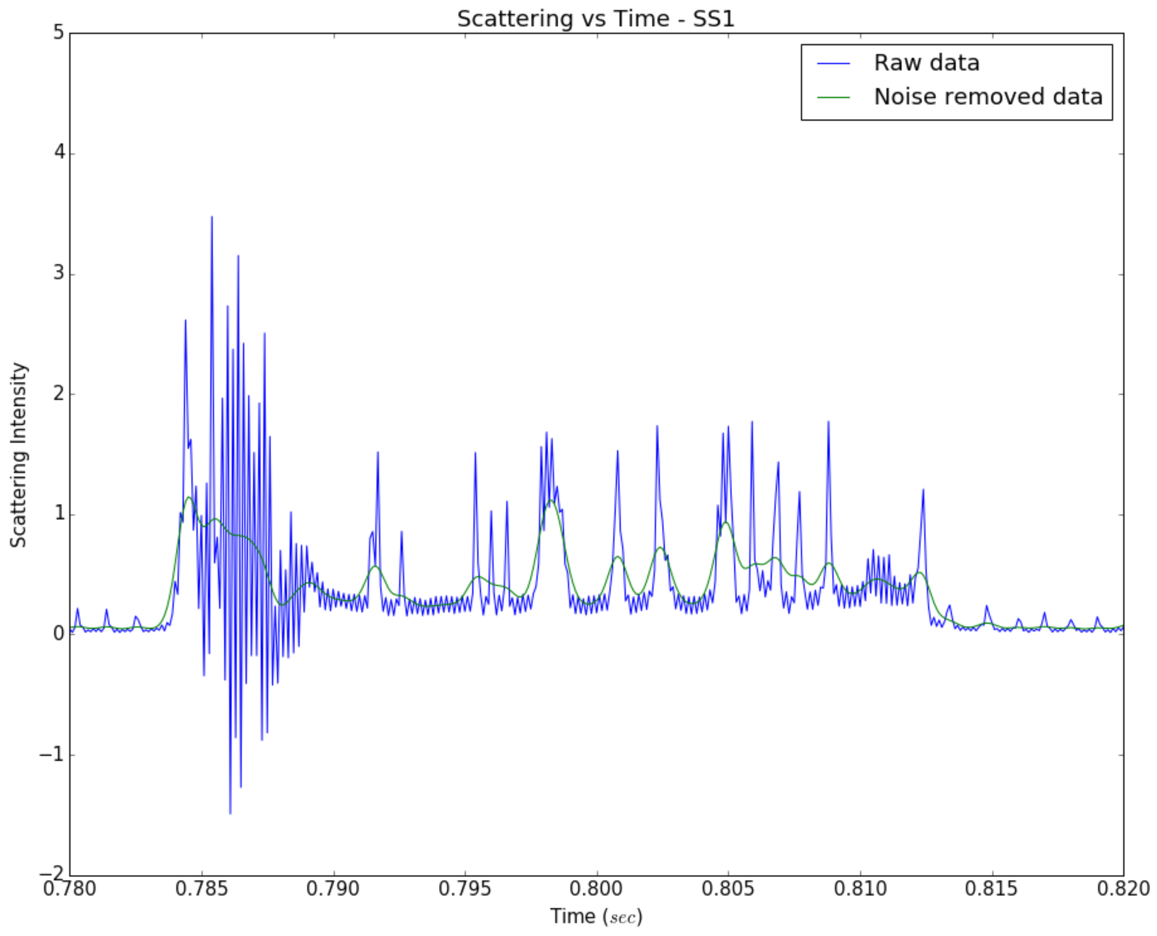
```
plotData("cleanSS1.txt")
xlim(0.18,0.24)
show()
```



In the case of SS1 however, a significant impact on peak height is seen. This problem is dealt with in the section on peak finding. If we focus on a particular part of the SS1 data as shown below, we can see a signal whose amplitude is comparable to that of the expected peak, which remains even after filtering of noise. This is the interfering signal which was mentioned earlier.

In [6]:

```
plotData("cleanSS1.txt")
xlim(0.78,0.82)
show()
```



Data Analysis

Peak detection

The method we used to find peaks was in essence a thresholding operation. Our aim was to first locate the tips of the peak with maximum height, and find the rest of the peak using that information. To do this, we thresholded the data and found the maximum height in every continous sequence of points which lie above the threshold. Each of these sequences corresponds to a peak in the data. The threshold was chosen using a one step iterative method. A tentative threshold was chosen as a certain fraction of the difference between the maximum and minimum value of the data. Then, all data points which lie above this threshold were averaged and this mean was multiplied again by the factor to yield the final threshold value. This was done to mitigate the effects of outliers.

Peak Analysis

Finding the properties of the peaks

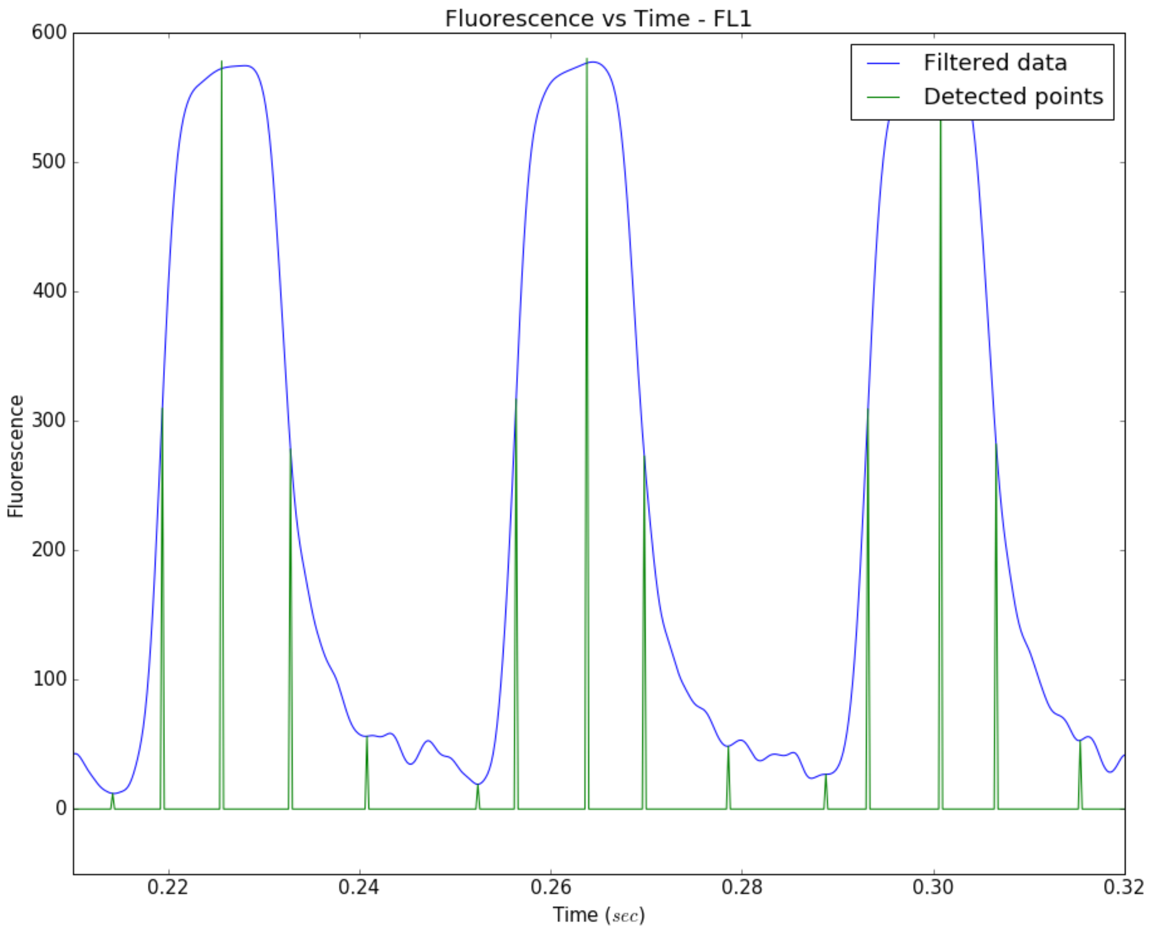
After finding the max heights of each of the peaks, the leading and trailing edges of the peaks were found using the first derivative of the data. We move to the left or right of the maximum height index until the slope of the data falls below a certain threshold, at which point we set that as the leading or trailing edge respectively. This threshold was determined by taking a fraction of the slope at the max height of the peak. A constraint was imposed on the edges such that they should lie below a certain fraction of the max height. The full-width at half-maximum was found by a simple thresholding operation on the data. The derivative method yields good results on the filtered data because the peak regions become much more smooth, resulting in a gradual change of the slope.

With regard to the problem of height reduction due to the filtering, the max height was recalculated after locating the endpoints by finding the maximum value of the original raw data in the resulting interval. The area under the peaks was calculated by using trapeziodal integration to integrate the filtered data between the previously found endpoints.

A sample of 3 detected peaks from FL1 is shown below, with the green spikes corresponding to the endpoints, points at the half-maximum and the maximum height.

In [7]:

```
plotData("detectedPeak.dat")
xlim(0.21,0.32)
ylim(-50,600)
show()
```



Analysing the properties of the peaks

The height, full-width at half-maximum(or simply width) and area of the peaks were found using the methods described above. The histograms of these properties are plotted below. The function used to plot them is defined as follows:

In [8]:

```python
def plotHists(s):
    """
    Python code to plot the data in a text file as histograms.
    Each column is treated independently and plotted.
    The first two lines of input can be optionally used to determine the
    title and labels of each plot.
    """
    # Open the file
    fo = open(s, 'r')

    rawlines = fo.read().split("\n")

    # Extract the lines with data
    lines = [x.strip() for x in rawlines if len(x) != 0 and x[0] != '#']

    n = len(lines[0].split(" "))

    # Extract labels and titles if possible
    title = []
    labels = []

    if rawlines[0][0] == '#' and rawlines[1][0] == '#':
        title = rawlines[0][1:].strip().split(";")
        labels = [[j.strip() for j in i.strip().split(',')]
                  for i in rawlines[1][1:].strip().split(";")]

    if len(labels) != n or len(title) !=n:
        title = [s+" X vs Y"]*n
        labels = [['X','Y']]*n

    # Close the file
    fo.close()

    # List of data
    states = []

    for i in lines:

        # Separate the columns
        values = i.split(" ")

        # Add each successive column as a new state in the state vector
        states.append([float(i) for i in values])


    # Plot the data
    for i in range(n):
        plt.figure()
        plt.xlabel(labels[i][0])
        plt.ylabel(labels[i][1])
        plt.title(title[i])
        plt.hist([k[i] for k in states])
        plt.show()
```

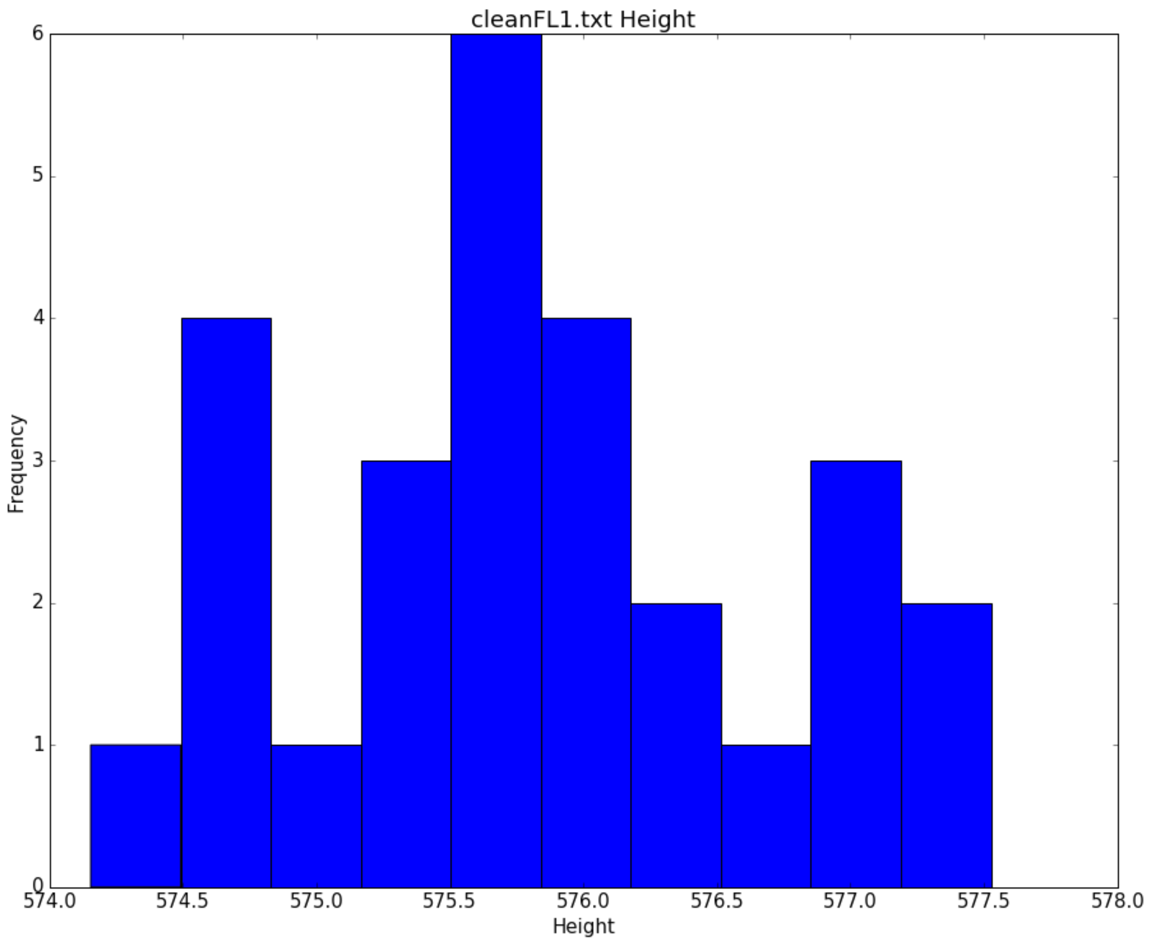
FL1:

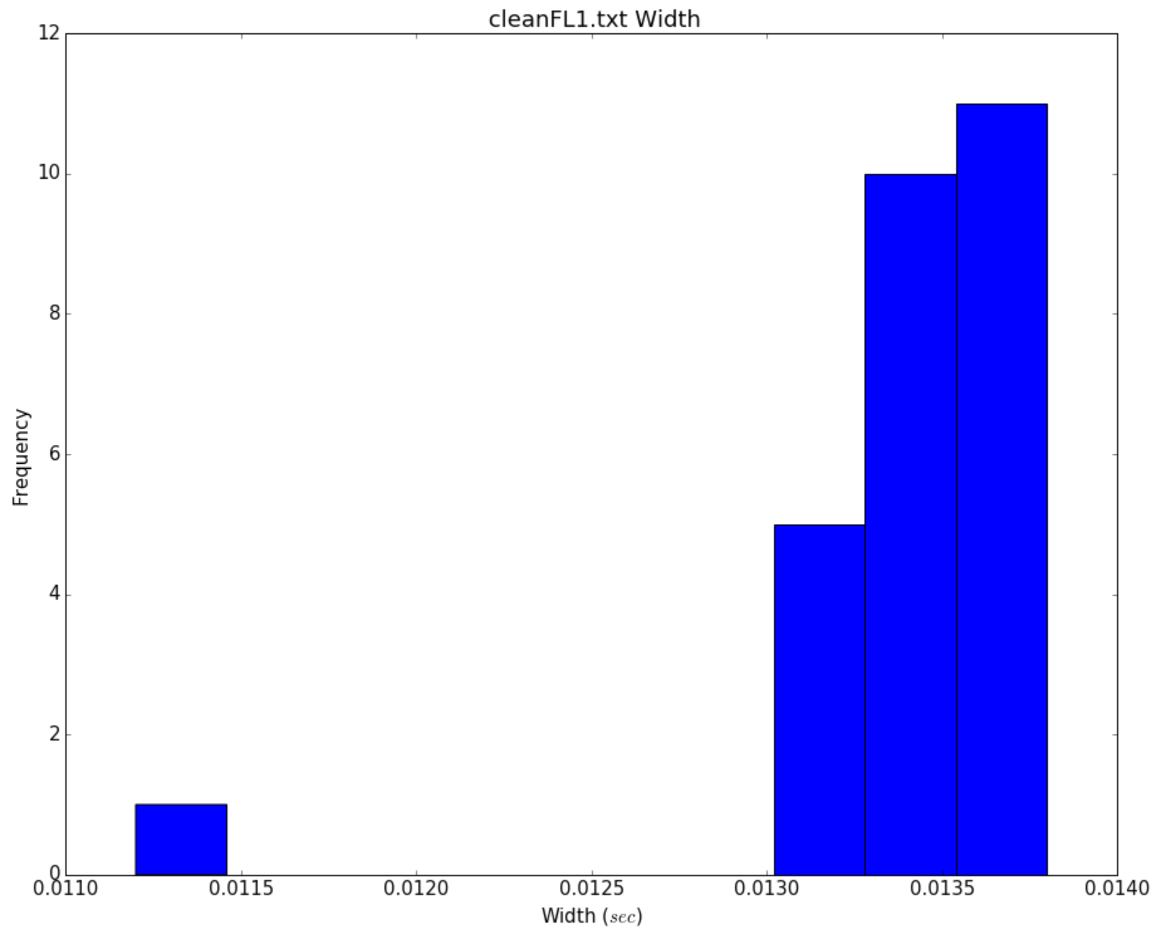
27 peaks were found with following statistics:

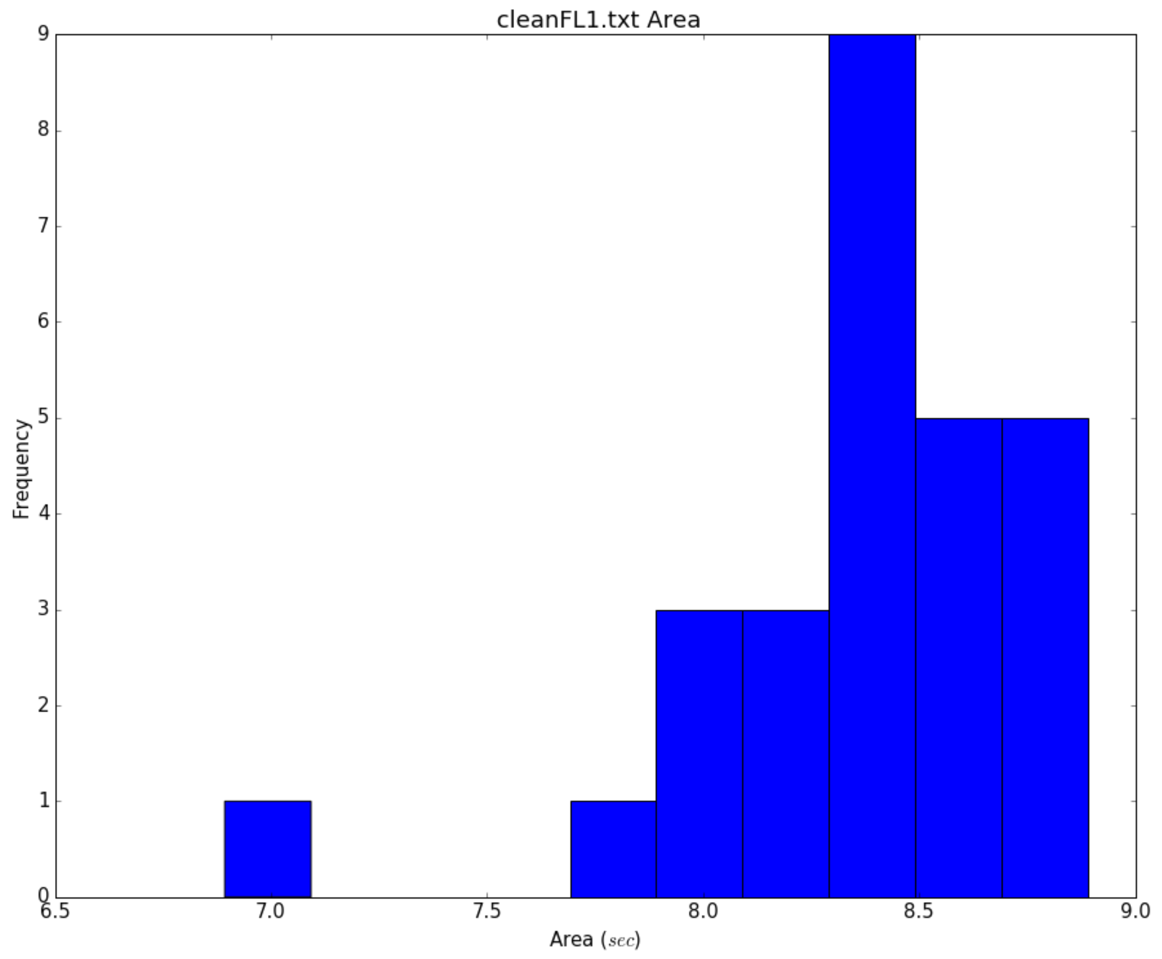
Property	**Mean**	**Standard Deviation**
Height	575.806407	0.855274
Width (sec)	0.013385	0.000465
Area (sec)	8.366186	0.384335

In [9]:

```
plotHists("peaksFL1.txt")
```


cleanFL1.txt Height
Frequency
Height
0
1
2
3
4
5
6
574.0
574.5
575.0
575.5
576.0
576.5
577.0
577.5
578.0


cleanFL1.txt Width
Frequency
Width (sec)
0
2
4
6
8
10
12
0.0110
0.0115
0.0120
0.0125
0.0130
0.0135
0.0140


cleanFL1.txt Area
Frequency
Area (sec)
0
1
2
3
4
5
6
7
8
9
6.5
7.0
7.5
8.0
8.5
9.0

It is clear from the above plots that for FL1 data, the peak properties follow roughly normal distributions, apart from some outliers in the data.

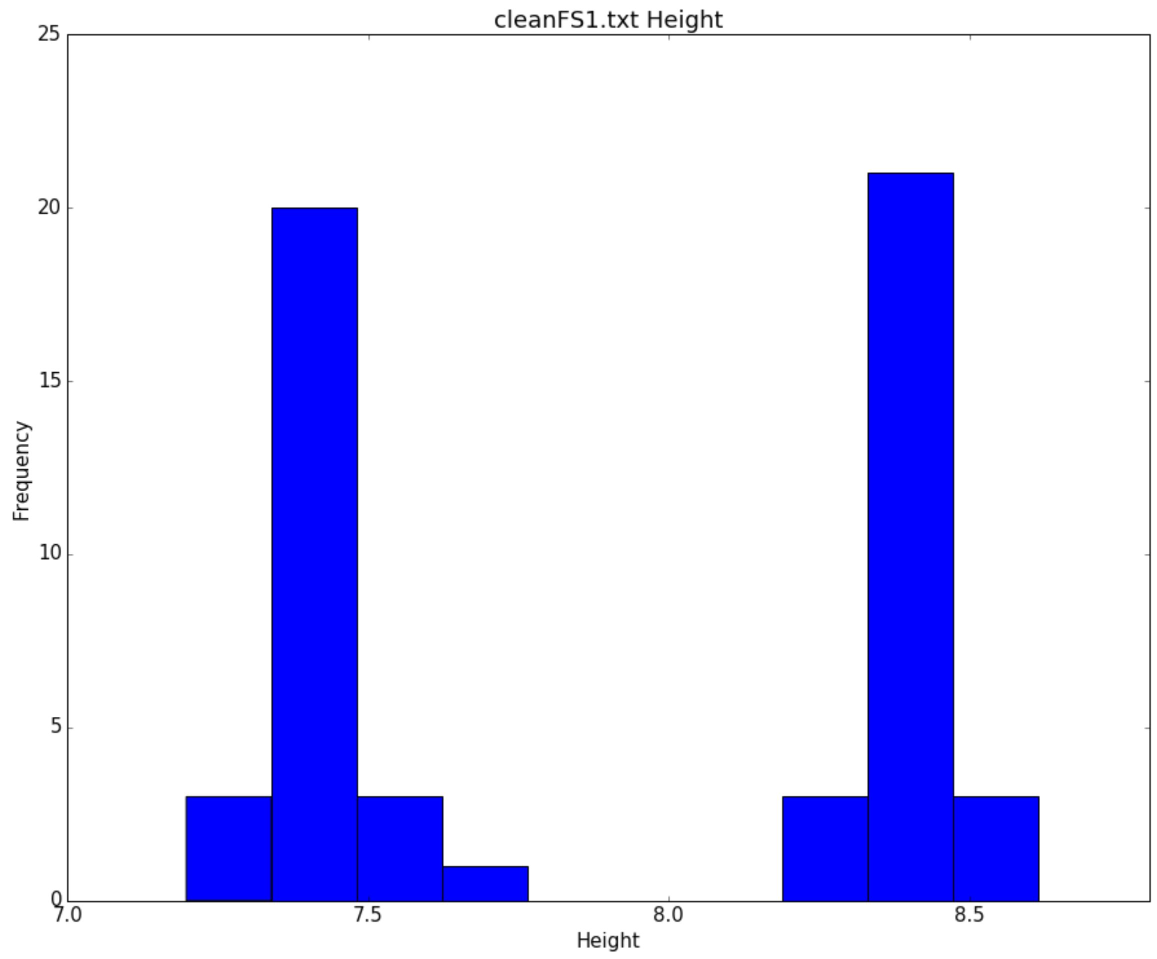
FS1:

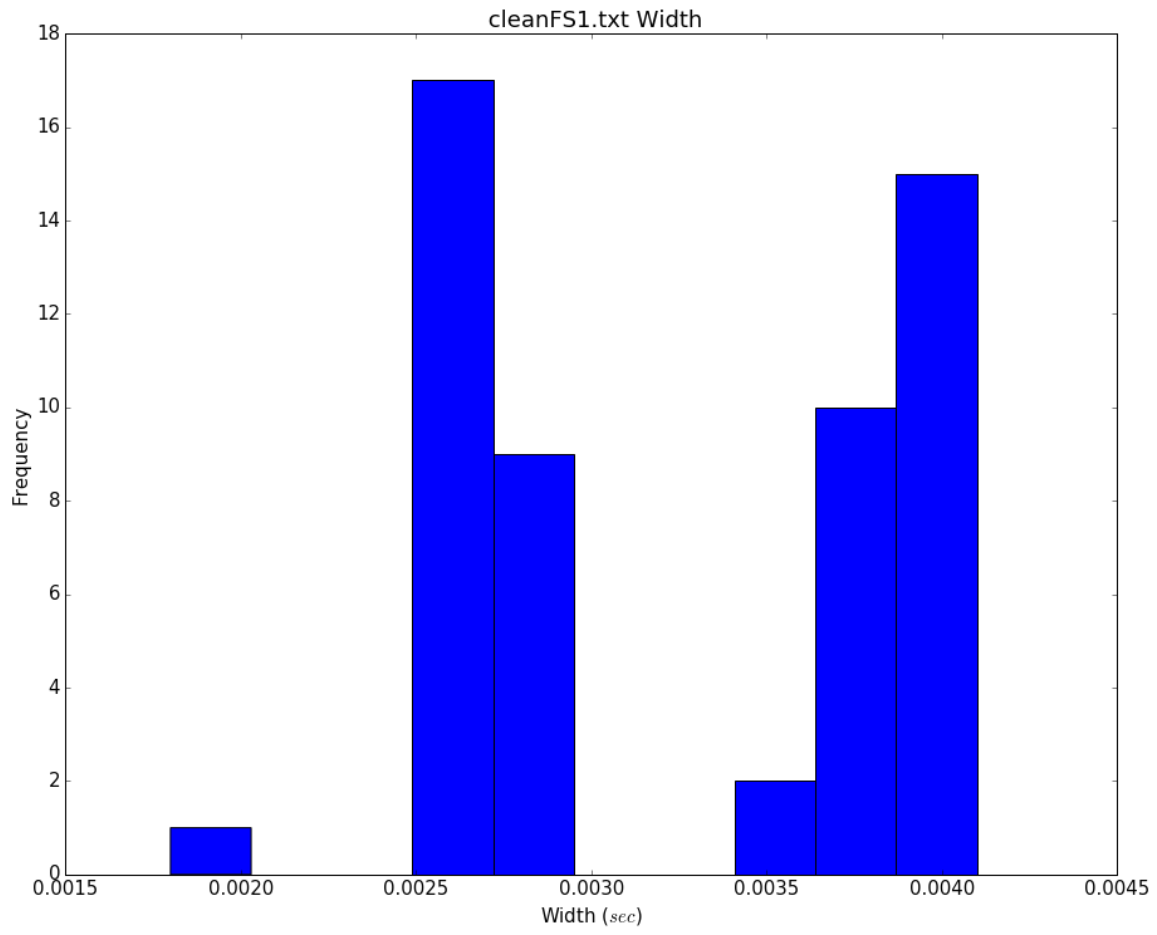
54 peaks were found with following statistics:

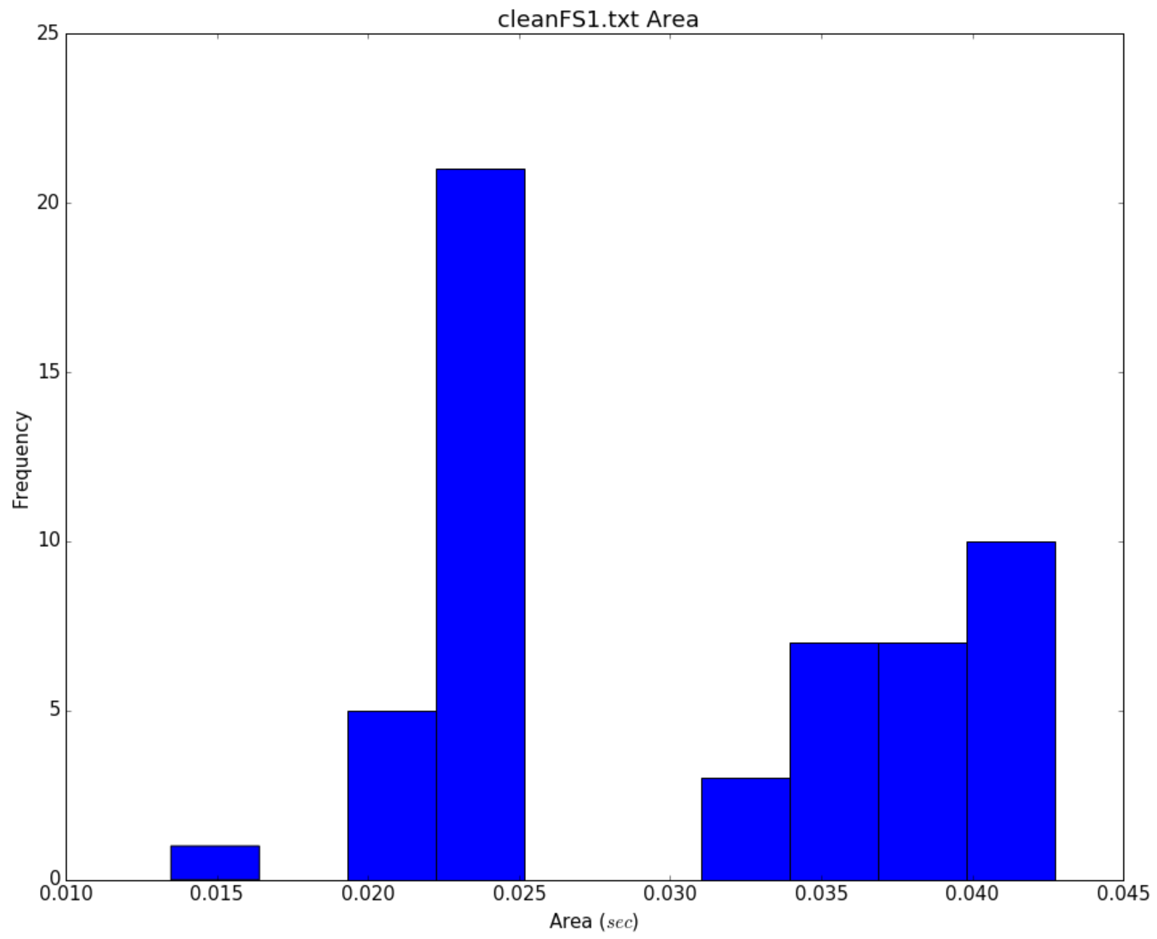
Property	**Mean**	**Standard Deviation**
Height	7.916457	0.499905
Width (sec)	0.003274	0.000619
Area (sec)	0.030494	0.008315

In [10]:

```
plotHists("peaksFS1.txt")
```


cleanFS1.txt Height
Frequency
Height
0
5
10
15
20
25
7.0
7.5
8.0
8.5


cleanFS1.txt Width
Frequency
Width (sec)
0
2
4
6
8
10
12
14
16
18
0.0015
0.0020
0.0025
0.0030
0.0035
0.0040
0.0045


cleanFS1.txt Area
Frequency
Area (sec)
25
20
15
10
5
0
0.010
0.015
0.020
0.025
0.030
0.035
0.040
0.045

It is clear from the above histograms that the peaks in the FS1 data set follow a bimodal distribution, for all the peak properties. Noting that the number of peaks found is double that of FL1, we can see that every two peaks in FS1 correspond to one peak in FL1.

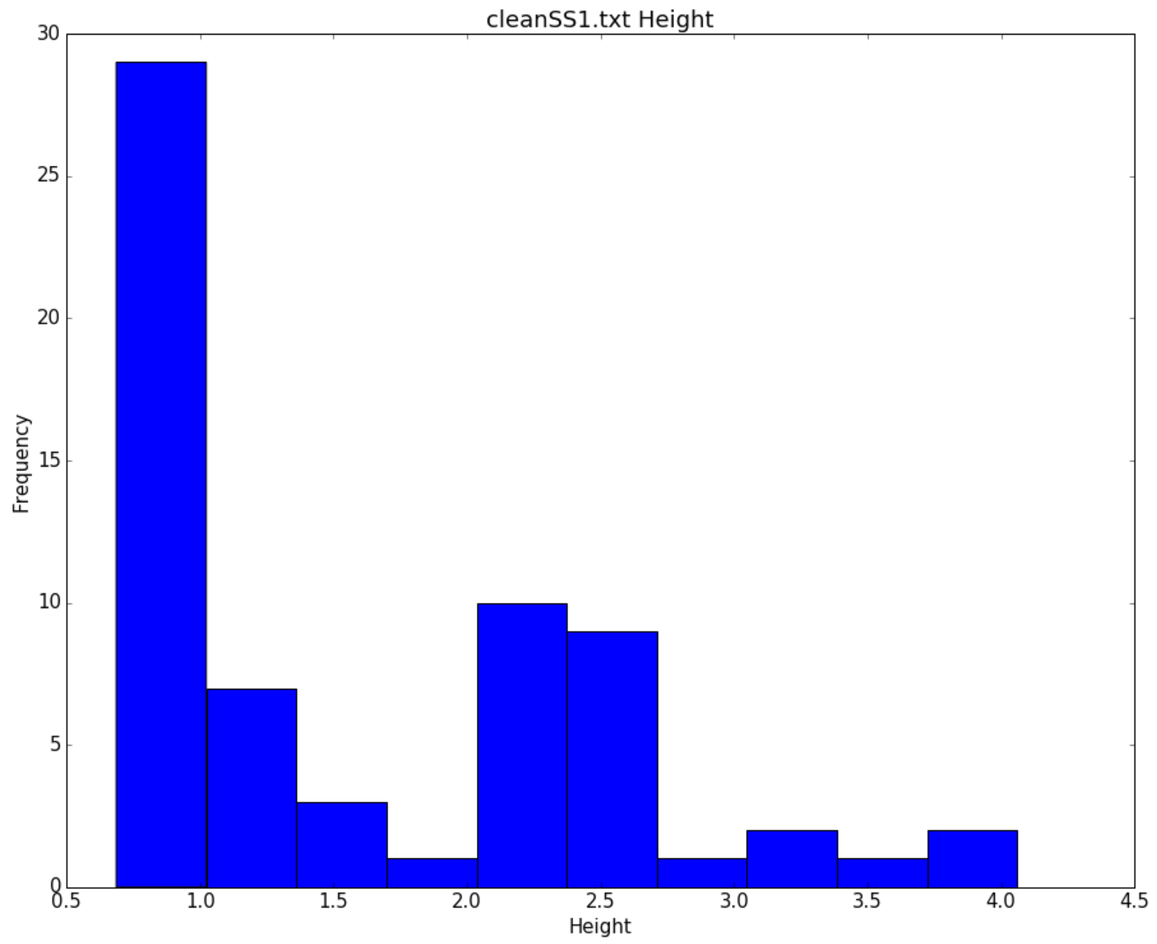
SS1:

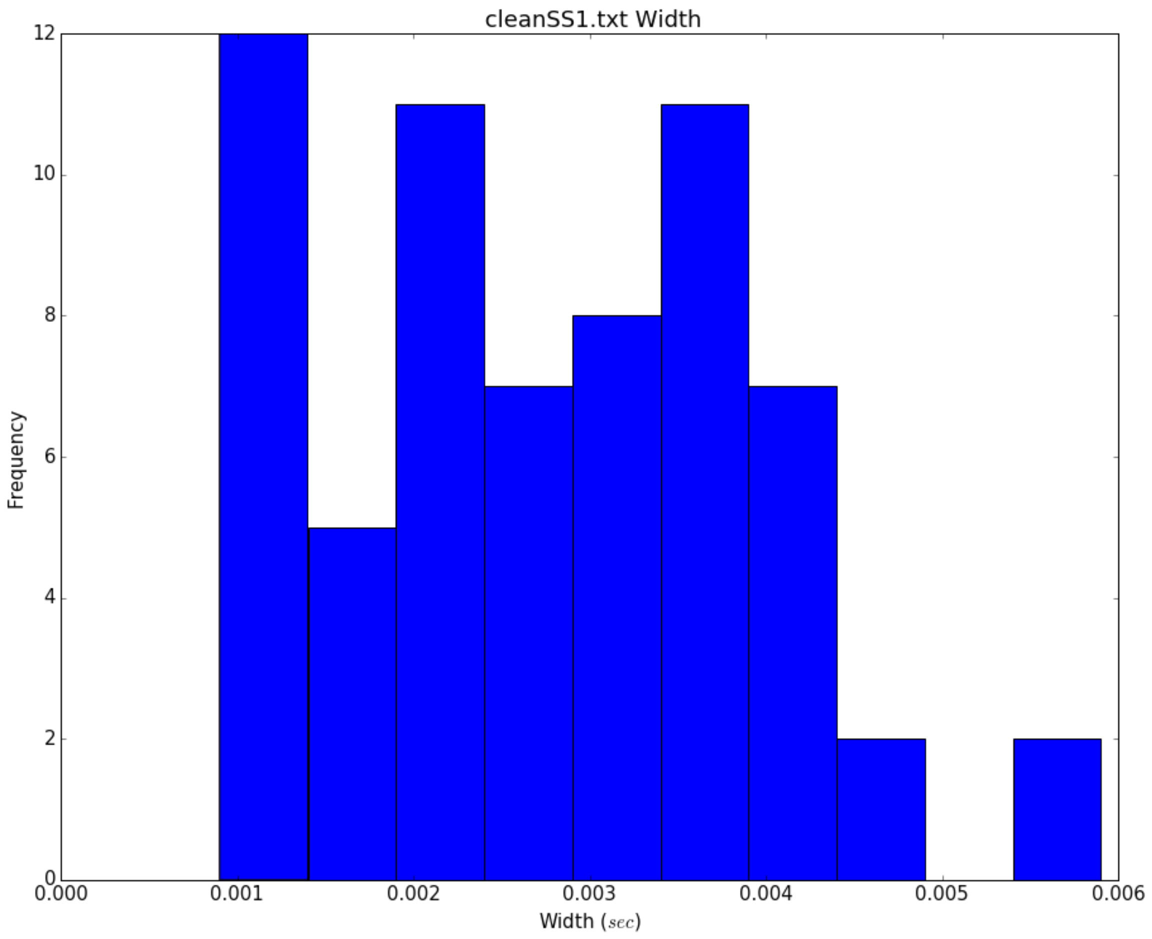
65 peaks were found with following statistics:

Property	**Mean**	**Standard Deviation**
Height	1.600141	0.904869
Width (sec)	0.002760	0.001150
Area (sec)	0.004053	0.002889

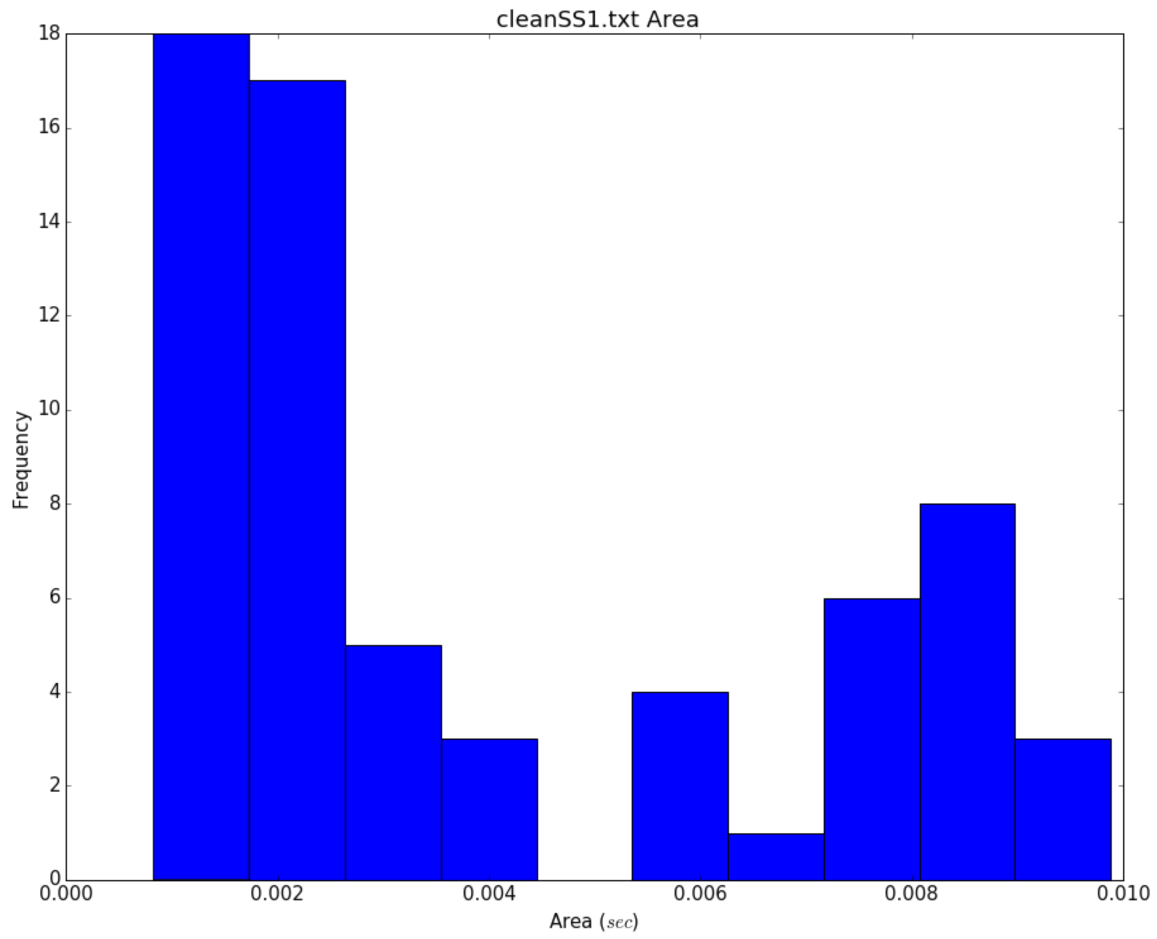
In [11]:

```
plotHists("peaksSS1.txt")
```


cleanSS1.txt Height
Frequency
Height
0
5
10
15
20
25
30
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5



cleanSS1.txt Width
Frequency
Width (sec)
0
2
4
6
8
10
12
0.000
0.001
0.002
0.003
0.004
0.005
0.006



While the expected number of peaks is equal to that of FS1, more were found due to the aforementioned interfering signal. However, a bimodal distribution is still evident in the histograms of all the properties(less so with width). This can be explained by the fact that the trailing peak in a pair of peaks is relatively short, and finding its width at half the max height may not be accurate in cases like those shown in the section on noise removal.