Characterization and Removal of HIS TOF Accidental Tails

Evan Shimoun
Professor Jim Raines

Summary

Solar Orbiter is a space probe which launched in February of 2020 and is currently orbiting the Sun over its polar regions. Its goal is to use its various instruments to collect data on the sun. The specific instrument I am working with is the Solar Orbiter's Heavy Ion Sensor. This sensor collects various types of ion data from the sun's solar wind. One such type of data collected is the energy per charge of the ion and the time of flight of the ion, which is how long it takes for the ion to pass through the sensor. These data points are plotted as the energy per charge versus the time of flight and are used to characterize what elements the ions passing through the sensor are. However, sometimes when ions are passing through the sensor, multiple can pass through, triggering the sensor to either register a very short or a very long time of flight. This is what's known as a time of flight accidental and is invalid data. This project developed and tested multiple methods to characterize and eliminate these time of flight accidentals through the use of the programming language IDL.

Introduction

Solar Orbiter

Recently launched in February of 2020, the Solar Orbiter space probe is currently orbiting the sun around its polar regions. The mission is helmed by the European Space Agency as a part of its Cosmic Vision program and was developed in collaboration with NASA and universities from around the globe. The spacecraft, as a part of its mission, contains multiple suites of instruments in order to conduct heliospheric physics and other space physics research. These instruments are the Energetic Particle Detector (EPD), Magnetometer (MAG), Radio and Plasma Waves (RPW), Solar Wind Plasma Analyzer (SWA), Extreme Ultraviolet Imager (EUI), Coronagraph (METIS), Polarimetric and Helioseismic Imager (PHI), Solar Orbiter Heliospheric Imager (SoloHI), Spectral Imaging of the Coronal Environment (SPICE), and X-ray Spectrometer/Telescope (STIX).

<u>Heavy Ion Sensor</u>

Part of the Solar Wind Plasma Analyzer suite is the Heavy Ion Sensor (HIS). This sensor was developed in conjunction with the Southwest Research Institute, IRAP, Toulouse, the University of Michigan, the University of New Hampshire, and the NASA Goddard Space Flight Center. The Heavy Ion Sensor takes measurements of ions from the solar wind, collecting data on the mass, charge, energy, direction, and velocity. The main components of the Heavy Ion Sensor are as follows: the ion steering (IS), the electrostatic analyzer (ESA), the time of flight (TOF) telescope, and solid-state detectors (SSD). The data collected by the Heavy Ion Sensor which is of particular interest for this project are the measurements of the energy per charge of the ions, which is essentially a measure of how energetic the ions are, and time of flight of the ion, which is essentially a measure of the velocity of the ion, as it is how long it takes for the ion to pass through the TOF sensor.

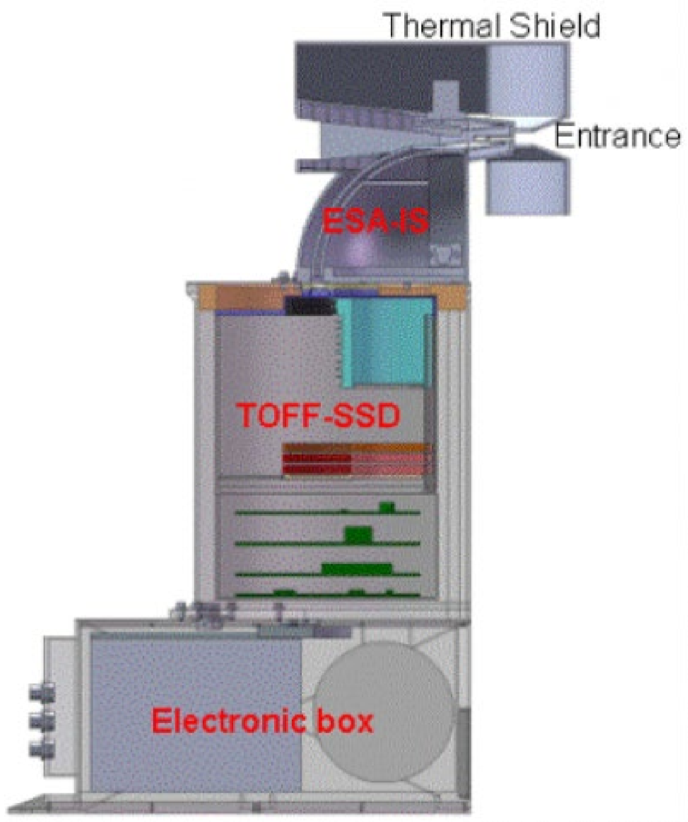


Figure 1: Diagram of the Heavy Ion Sensor modules

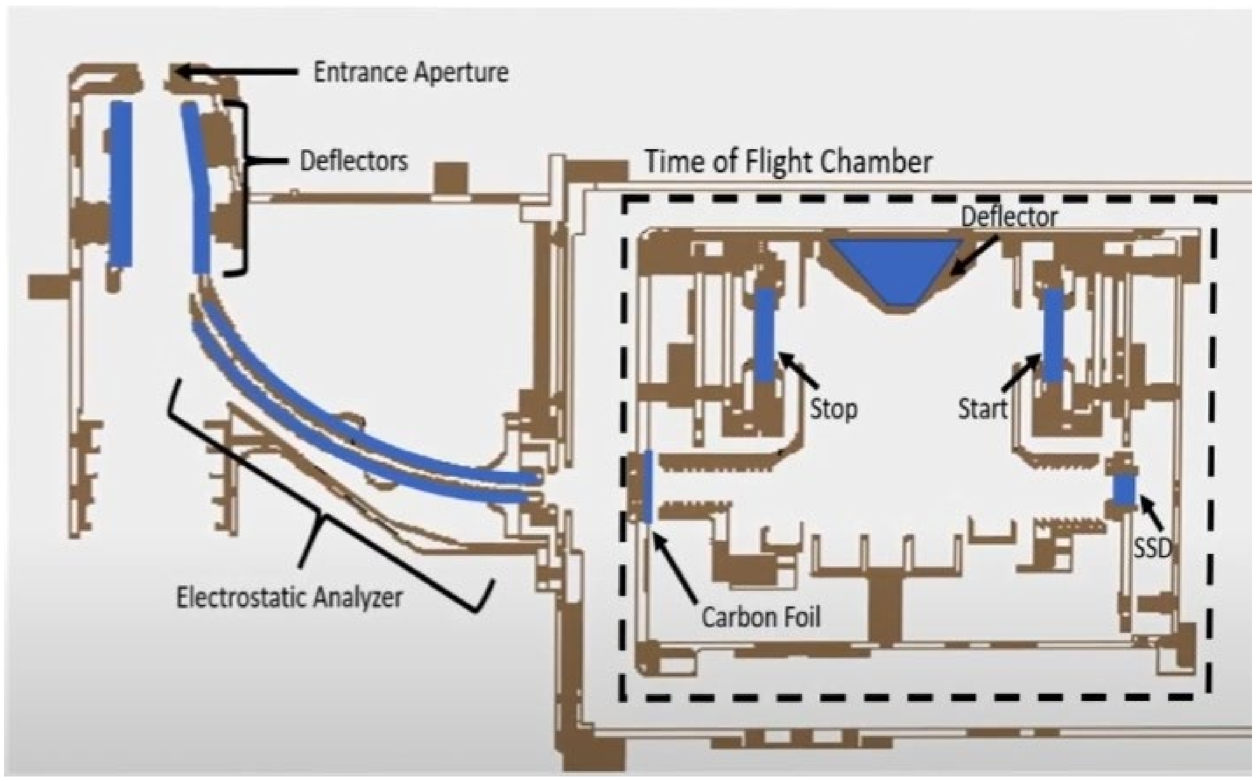


Figure 2: Schematic of the Heavy Ion Sensor. Ions enter through the entrance aperture and are redirected by use of the deflectors (IS) into the electrostatic analyzer (ESA). From there the ions pass through the carbon foil in the time of flight (TOF) chamber where the ions are neutralized, kicking off an electron which pass to the start detector. The neutralized ion then travels to the solid state detector (SSD) which it then impacts, producing an energy measurement and kicking off another electron which passes to the stop detector. The impacts of the electrons upon the start and stop detectors are what trigger the measurements of the time of flight data.

Energy per Charge and Time of Flight Data and Time of Flight Accidentals

For this project, the main data set which I analyzed was the energy per charge versus the time of flight of the ions. This data, as stated previously, was all collected by the Heavy Ion Sensor. The energy per charge data, or E/q data, was measured in units of steps, from 0 to an integer. The time of flight data was measured in units of channels, from 0 to 511 for a total of 512 channels. A lower speed of the ion meant a lower time of flight channel was recorded for that ion. To perform the analysis for this project, the energy per charge and time of flight data were plotted against each other in a spectrogram such as in Figure 3. Of particular note is the large bar structure towards the bottom of the spectrogram. The vast majority of the data points in this structure are what are known as accidentals, or invalid data points. These accidentals occur due to cases such as when multiple ions enter the time of flight chamber at the same time, resulting in either far too short or far too long time of flight recordings, creating an invalid data point. Many of these accidentals occur at a similar energy per charge step which results in the large bar structure known as an accidental tail. The main goal of this project is to characterize these accidentals within the energy per charge vs time of flight data set and eliminate them, so that proper data may be provided to scientists for use in their research and projects.

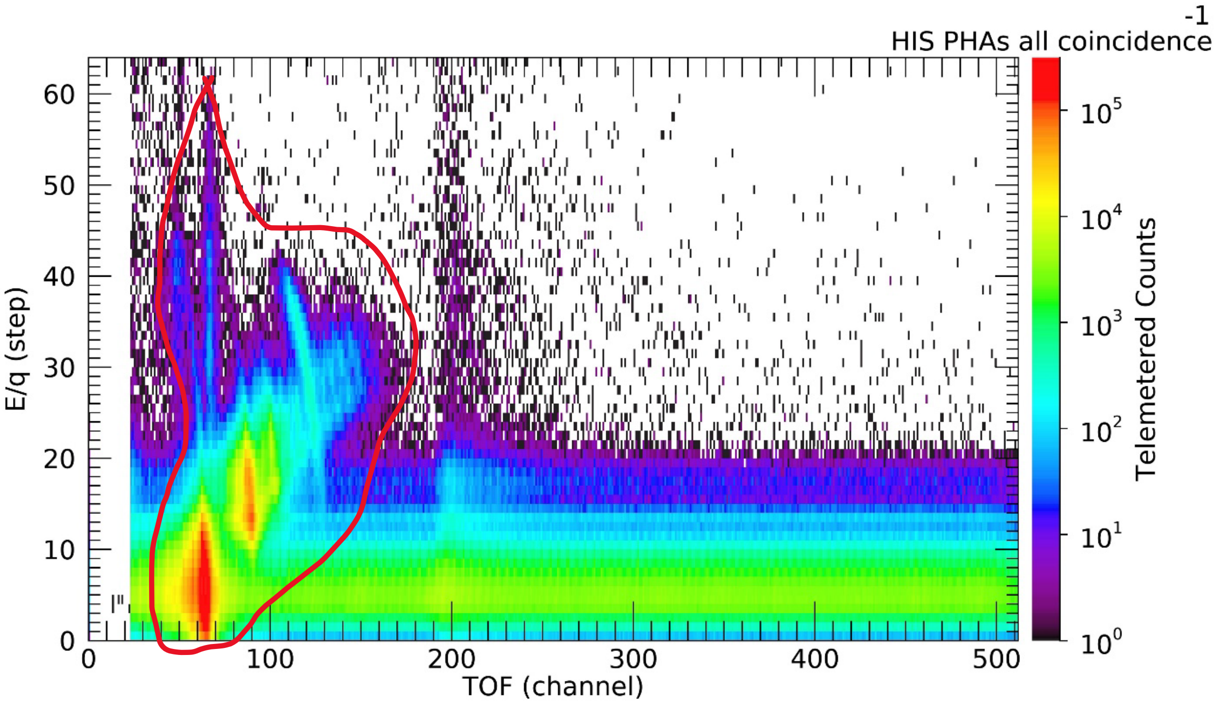


Figure 3: A spectrogram of the energy per charge data vs the time of flight data from the full day of January 2nd, 2023. This spectrogram consists of all data points from the day, including all of the accidentals. The area encircled in red is the valid data, while the remainder of the points are considered to be the accidentals.

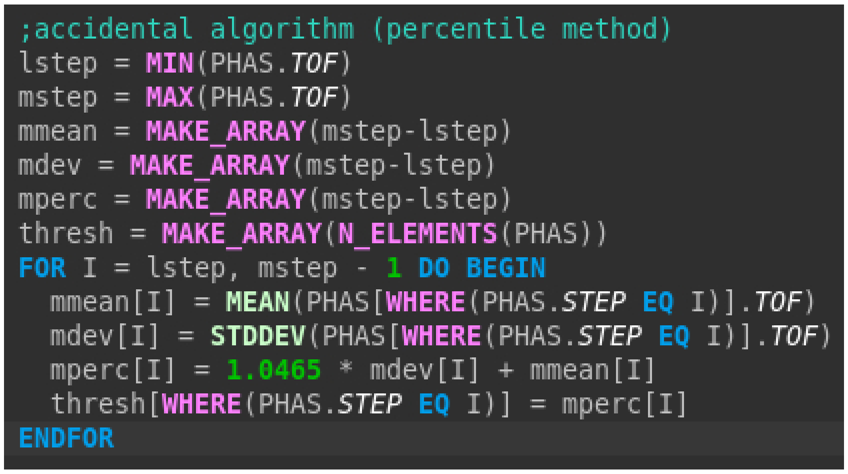
Methods

Plotting and Analysis through IDL Programming Language

For this project, all of the plotting and analysis functions were written in the IDL programming language. This is the language with which I and the rest of the SOHIS have been working with to process all the data. As such, my characterization and removal of the time of flight accidentals needed to be written in IDL as well in order to maintain the data format with which we were processing the data. This also ensured further compatibility with the rest of the programs we use to perform analysis data from the Heavy Ion Sensor.

Of importance to note before discussing methodology of eliminating time of flight accidentals is the volume of data. Each day of data recorded from the Heavy Ion Sensor contains approximately 30 million data points. As a result, some more complex and picky algorithms are not the most feasible due to the time cost in running them. Quite simply it would take far too long to run the program for them to be possible. The methods I selected were the ones I was able to ensure ran in a reasonable amount of time.

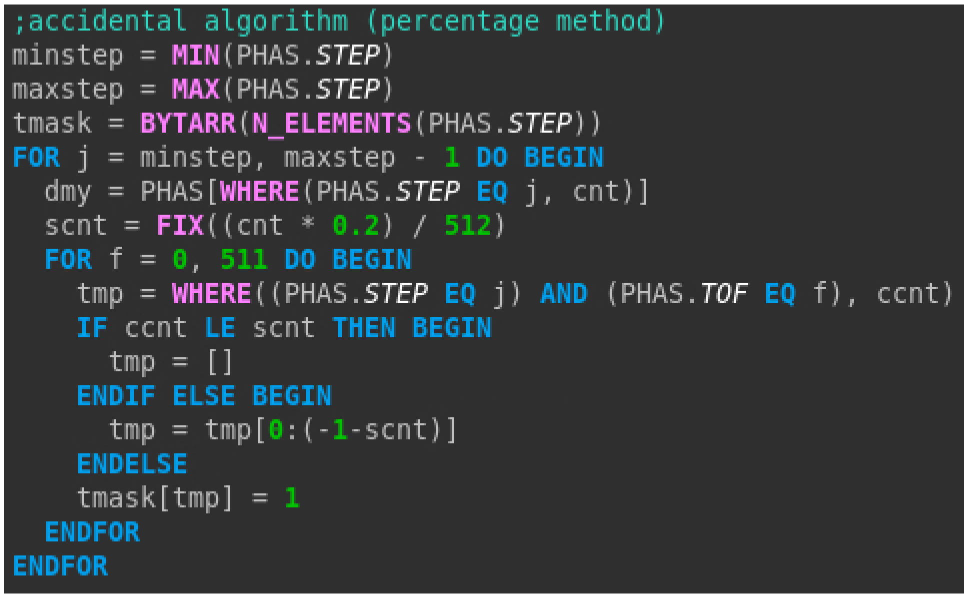
My first attempt to eliminate the time of flight accidentals involved the use of percentiles. From the plots of the energy per charge and time of flight data, I could see that at each energy per charge step, the majority of the data points were concentrated around the valid data, with the accidentals being located to the right at higher time of flight values. As such, for each energy per charge step, I determined the percentile values from all the data points at that step. Thereafter, I eliminated all the data points at that step with a time of flight greater than the selected percentile's value.



```
;accidental algorithm (percentile method)
lstep = MIN(PHAS.TOF)
mstep = MAX(PHAS.TOF)
mmean = MAKE_ARRAY(mstep-lstep)
mdev = MAKE_ARRAY(mstep-lstep)
mperc = MAKE_ARRAY(mstep-lstep)
thresh = MAKE_ARRAY(N_ELEMENTS(PHAS))
FOR I = lstep, mstep - 1 DO BEGIN
  mmean[I] = MEAN(PHAS[WHERE(PHAS.STEP EQ I)].TOF)
  mdev[I] = STDDEV(PHAS[WHERE(PHAS.STEP EQ I)].TOF)
  mperc[I] = 1.0465 * mdev[I] + mmean[I]
  thresh[WHERE(PHAS.STEP EQ I)] = mperc[I]
ENDFOR
```

Figure 4: Code implementation of the percentile method in IDL. Using the mean and standard deviation of the time of flight values located at the specified step, the value of the 85th percentile is calculated using the z-score method. A threshold for use with the plotting of the spectrogram is then specified using this value.

My second attempt to eliminate the time of flight accidentals involved removing a specified percentage of the data points at each energy per charge step. This method was suggested to me by Dr. Stefano Livi. To implement the percentage removal method, I calculated the total number of data points among all the time of flight channels at each energy per charge step. I then multiplied this by the selected percentage and then divided by 512, the total number of time of flight channels. This resulted in the number of data points to remove from each time of flight channel at the energy per charge step. This method operates under the assumption that in general, for the whole set of data points, a certain percentage of the data will be accidentals. The real data will also generally be more concentrated than the accidentals, so by removing a percentage of the data points, the incorrect outliers will be eliminated.



```
;accidental algorithm (percentage method)
minstep = MIN(PHAS.STEP)
maxstep = MAX(PHAS.STEP)
tmask = BYTARR(N_ELEMENTS(PHAS.STEP))
FOR j = minstep, maxstep - 1 DO BEGIN
  dmy = PHAS[WHERE(PHAS.STEP EQ j, cnt)]
  scnt = FIX((cnt * 0.2) / 512)
  FOR f = 0, 511 DO BEGIN
    tmp = WHERE((PHAS.STEP EQ j) AND (PHAS.TOF EQ f), ccnt)
    IF ccnt LE scnt THEN BEGIN
      tmp = []
    ENDIF ELSE BEGIN
      tmp = tmp[0:(-1-scnt)]
    ENDELSE
    tmask[tmp] = 1
  ENDFOR
ENDFOR
```

Figure 5: Code implementation of the percentage removal method in IDL. Here the code iterates through each energy per charge step calculating the total number of data points at this step. This is then multiplied by a percentage, in this case 20% (or 0.2), and then divided by the total number of time of flight channels. The resultant number is the number of data points to remove from the specified energy per charge step and time of flight channel coordinate. The program iterates through each time of flight channel at this energy per charge step, removing the specified number of data points or setting the number of data points to zero. This is all done through the use of a mask which is later applied when running the plot spectrogram function to aid in increased efficiency in running the program due to the sheer number of data points which would be needed to be saved.

Results

Method 1: Percentiles

The percentile method of removing the time of flight accidentals was only partially successful. Using the 85% threshold selected, a large amount of the time of flight accidentals, including much of the time of flight accidental tail, were successfully eliminated. However, none of the accidentals at the lower time of flights were removed, and some valid data was eliminated as well. In addition, not all of the tail was eliminated, only part of it. As such, this method on its own would not be sufficient for eliminating the time of flight accidentals reliably.

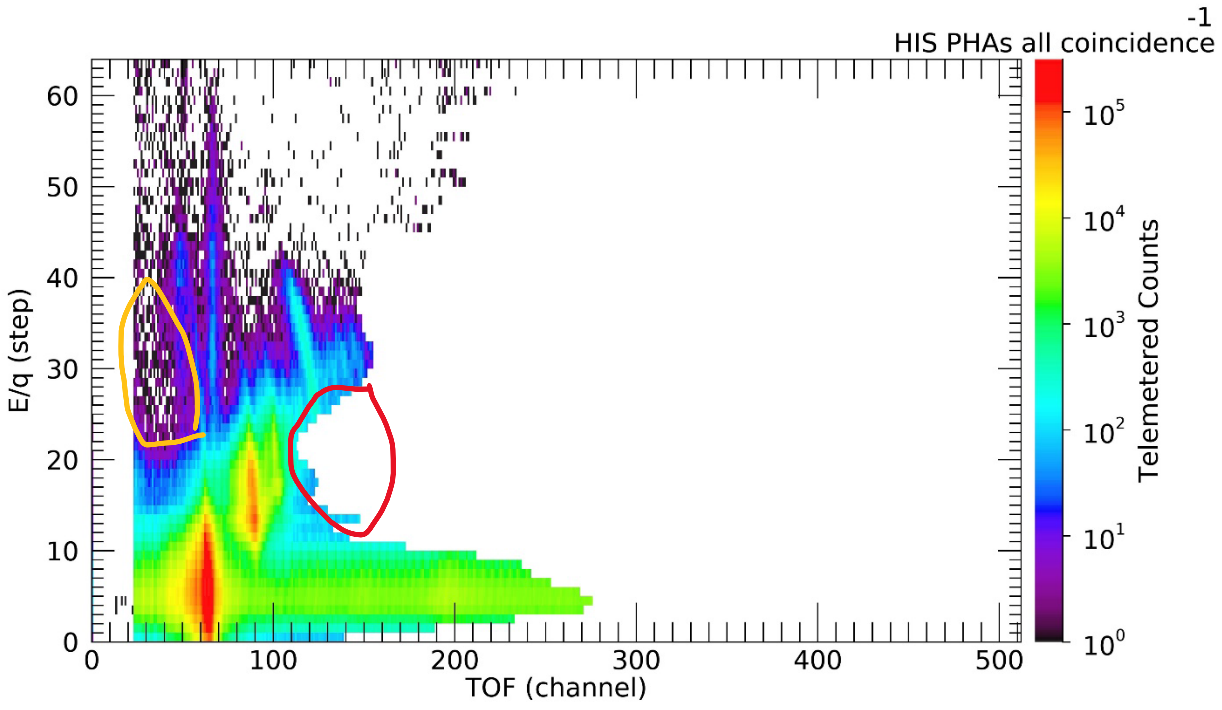


Figure 6: A spectrogram of the energy per charge data vs the time of flight data from the full day of January 2nd, 2023 with the percentile elimination method implemented at a threshold of the 85th percentile. As can be seen in the figure, much of the time of flight accidental tail has been eliminated, but a good chunk of it, from around time of flight channel 100 to 275, still remains. The red circled area highlights where real data was eliminated and the yellow circled area highlights where some of the accidentals outside of the tail remain.

Method 2: Percentage Removal

The percentage based method of removing the time of flight accidentals was also only partially successful, though in a different way than the percentile method. The percentage method was much more successful at eliminating many of the accidentals across the whole spectrogram while maintaining the real data. However, the percentage based method essentially did not effect the time of flight accidental tail, leaving the whole structure intact. As such, this method on its own is also not sufficient for eliminating the time of flight accidentals reliably.

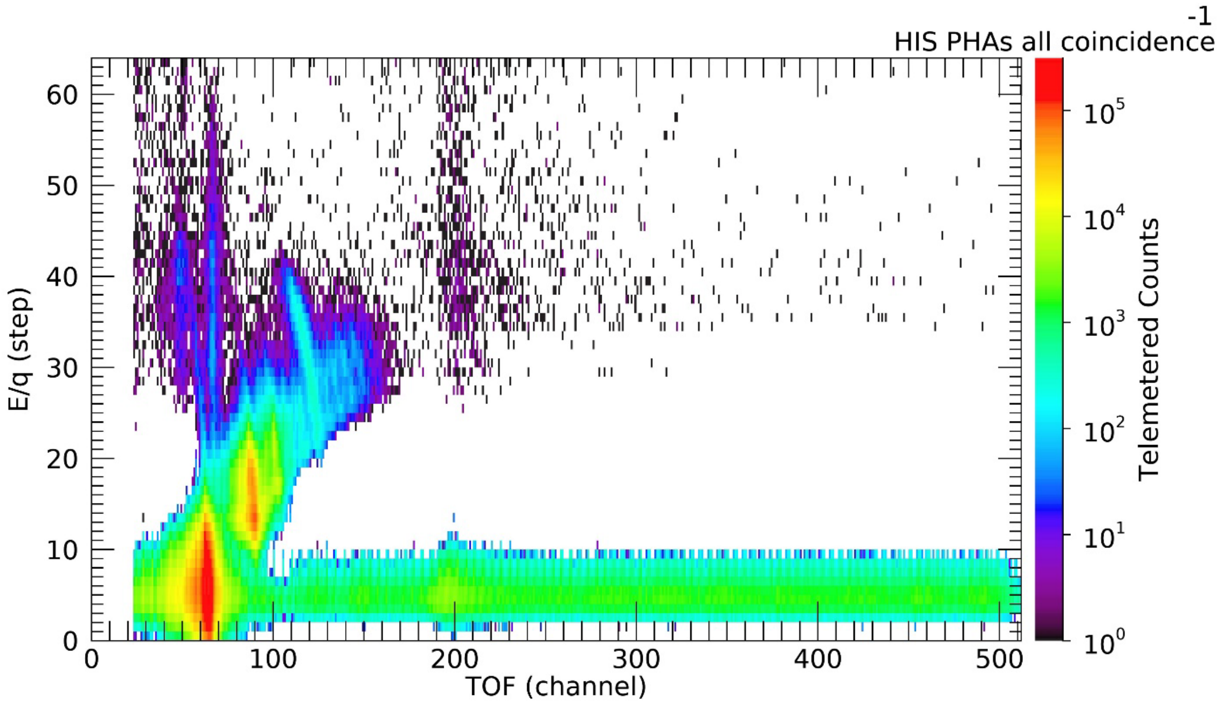


Figure 7: A spectrogram of the energy per charge data vs the time of flight data from the full day of January 2nd, 2023 with 20% of the data points eliminated from each energy per charge step. As can be seen in the figure, many of the time of flight accidentals throughout the figure are eliminated with the valid data points preserved with only minor loss. However, the time of flight accidentals tail is still distinct.

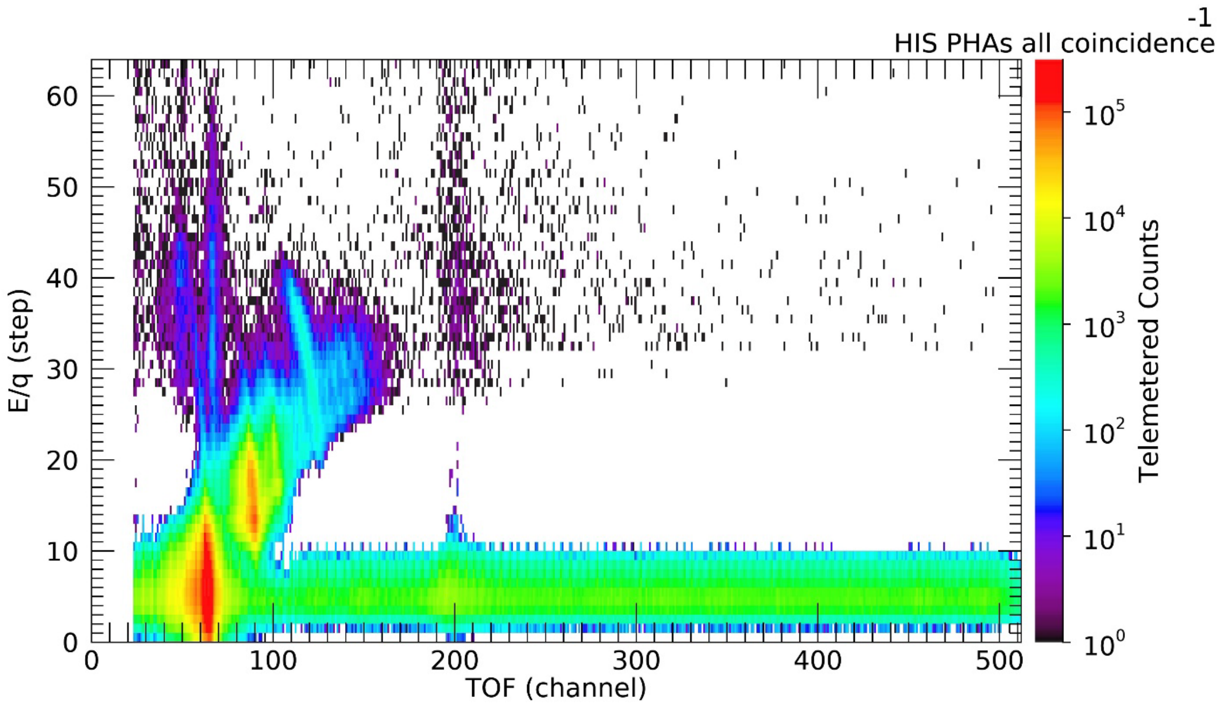


Figure 8: A spectrogram of the energy per charge data vs the time of flight data from the full day of January 2nd, 2023 with 15% of the data points eliminated from each energy per charge step. In comparison with Figure 7, the shape of the valid data points is preserved better, though at the cost of more of the time of flight accidentals remaining.

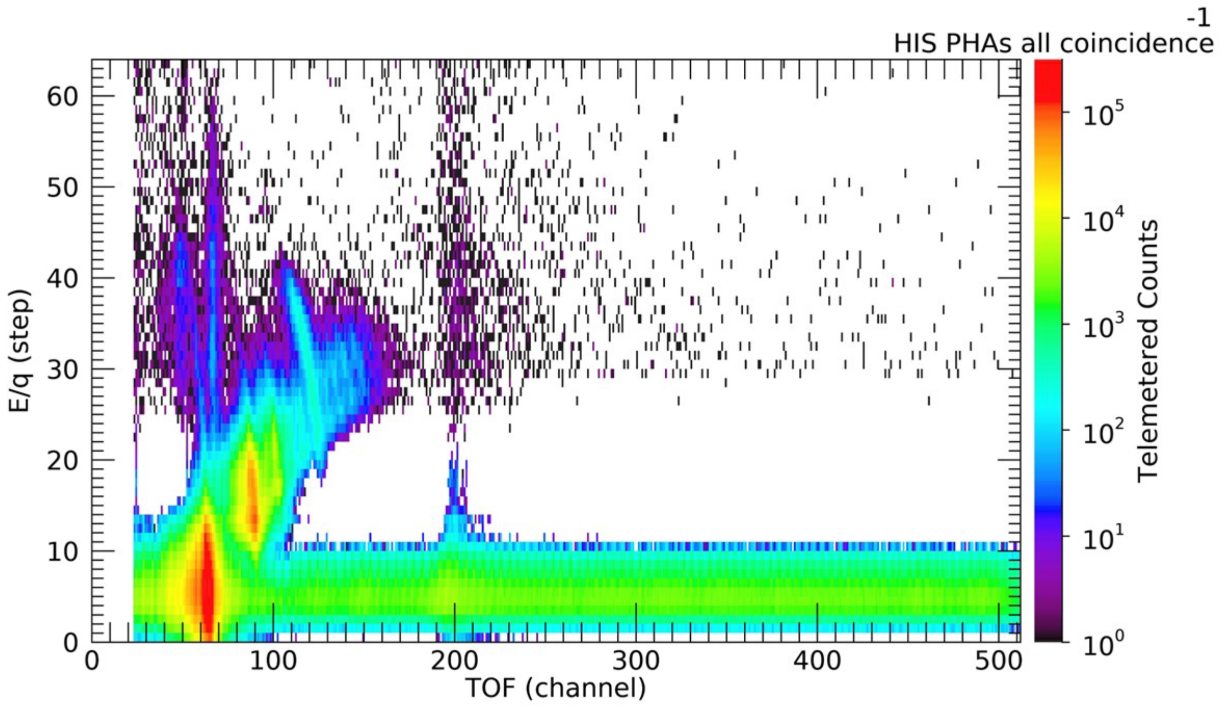


Figure 9: A spectrogram of the energy per charge data vs the time of flight data from the full day of January 2nd, 2023 with 10% of the data points eliminated from each energy per charge step. In comparison with Figure 8, the shape of the valid data points is approximately the same, though a greater number of time of flight accidentals remain.

Future Plans

As neither currently implemented method of eliminating the time of flight accidentals fully succeeded, more work is needed to be done to determine the best way to remove them in their entirety. I will be continuing this research as a part of the SOHIS team over the summer of 2023. Currently, I already have a plan in place for my next method to attempt to eliminate the time of flight accidentals. This new method will compare the recorded velocity of the individual ions, which is taken from the time of flight data, to the recorded bulk wind speed of the solar wind. The bulk wind speed is how fast the solar wind moves as a whole, so in essence, this method will be checking to see if ions are moving either extremely fast or extremely slowly in comparison to this wind speed. The ions that are the outliers are most likely time of flight accidentals and can therefore be eliminated. This method is based upon work previously done by Dr. Jason Gilbert for the Solar Wind Ion Composition Spectrometer, a different ion sensor which was a part of the Advanced Composition Explorer mission. In the case this method fails too, I will most likely need to look into potentially implementing a combination of all three methods, though the results of all of these attempts are yet to be determined.

Conclusion

This project had the goal of eliminating the time of flight accidentals from the Heavy Ion Sensor's data. Though my existing methods of attempting this were unsuccessful, I have already been planning future ideas for new methods to attempt. Furthermore, this project offered me extremely valuable experience in working with extraordinarily large quantities of data, something which poses a unique challenge when performing such data analysis tasks. Going forward, such knowledge will be made good use of in both finishing this project and in future work.