Institut für Physik der Atmosphäre
German Aerospace Center (DLR)

Internship Report

Author
Ross Knapman

Supervisor
Jonas Herbst

August 31, 2017

1 Determining the Peak of an Intensity Profile

1.1 Introduction

The fringe pattern from the interferometer falls onto an array of PMTs resulting in a light distribution similar to a skewed cosine. By determining the position of the peak of this intensity pattern, we can determine the Doppler shift of the backscattered light from aerosols and hence the windspeed.

1.2 Model Fitting

The most promising way to determine the peak of the light distribution would be to fit an analytical model to the intensity data. This process entails three main steps:

1. Define a model.
2. Stepwise integration over each of the pixels.
3. Apply a minimisation algorithm to determine the optimum parameters of the model.

Step 1 involves trying many different fit models and determining the quality of the fit. This can be done by taking the percentage error between the fitted curve and the data point for each pixel.

Step 2 corrects for the fact that the light distribution cannot be considered to be uniform over each of the pixels.

Although the scipy Python library can carry out step 3, I wrote a function to perform downhill simplex optimisation, mainly for the purpose of obtaining a better understanding of the task at hand. To demonstrate this, let's consider fitting the function $y = \cos(ax + b)$ to a normalised intensity profile. As can be seen in Fig. 1, the initial guess parameters were $a = 0$ and $b = 5$, with a starting simplex defined by $\Delta a = 0.3$ and $\Delta b = 3$. In the plot, one can see how the simplexes converge towards a better fit (darker region), resulting in the fit of the (blue) line to the (orange) data after 100 iterations.

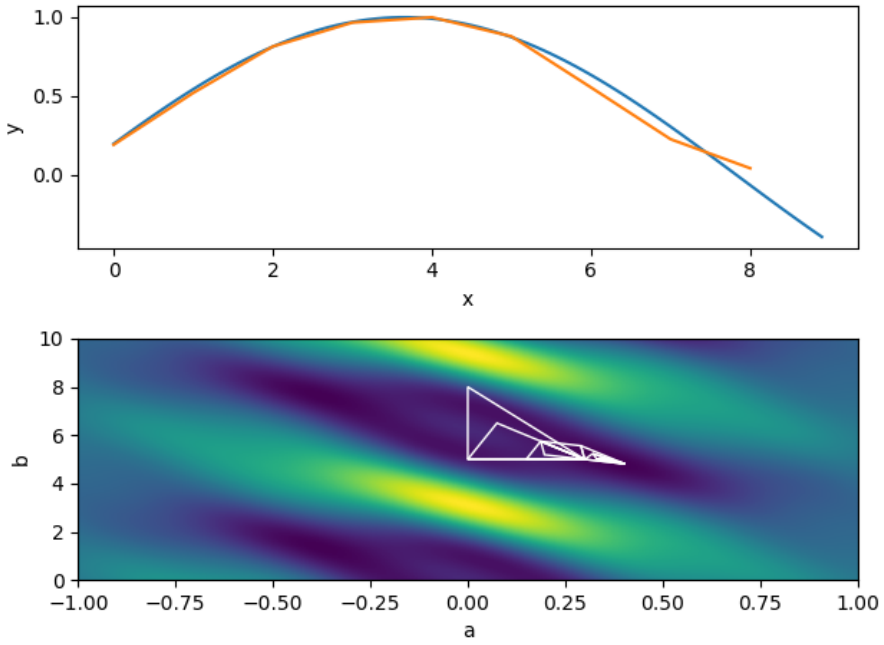


Figure 1: Top: The result of using the downhill simplex algorithm to fit data. Bottom: Illustration of the downhill simplex search in parameter space.

A lot of time was dedicated to trying out different fit models, and automatically writing the results of each to a spreadsheet in order to compare their qualities of fit. As of now, the best model seems to be:

$$f(x, w) = w_1 + w_2 \cos(\sin^2(w_4(x - w_0)) + w_5(x - w_0) + w_6)$$

which can be seen in Fig. 2.

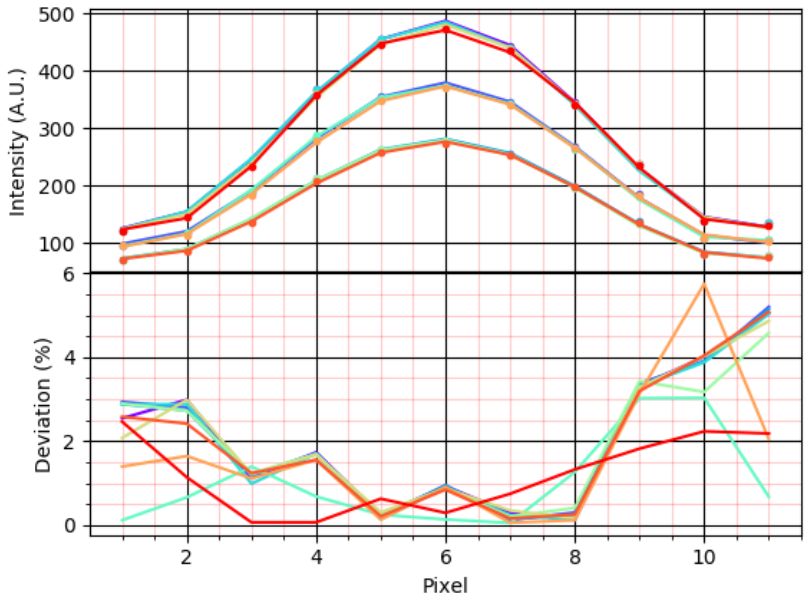


Figure 2: The best fit model so far and its percentage deviation from the measured intensity distribution.

However, this model could be problematic in that both w_0 and w_6 contribute to the phase shift, making it difficult to read out after parameter optimisation. Additionally, determination of initial guess parameters that converge well and allow the peak value to be read has so far proven to be a challenge.

1.3 Centroid Method

Another method used to try to determine the peak of the intensity distribution was to find the centroid of the peak. This uses the formula

$$\bar{x} = \frac{\sum_{i=k}^{k+m} x_i \cdot I_i}{\sum_{i=k}^{k+m} I_i}$$

to determine the position of the peak $\bar{x}$ where k is the index of the leftmost pixel, m is the number of pixels the intensity distribution lies across, x_i is the horizontal position of each pixel, and I_i is the intensity of each pixel. From this, the Doppler shift and hence the windspeed can be determined.

1.4 Gaussian Correlation Algorithm

Another method to determine the position of the intensity distribution peak is the Gaussian correlation algorithm (GCA). In this, we assume the intensity profile to be a Gaussian distribution, and the correlation function with the measured intensity is determined. We can then use an iterative procedure to find the horizontal position of the peak. Both the GCA and the centroid method have proven to be less successful than model fitting in the determination of the peak position.

1.5 Genetic Algorithm Fitting

The process of guessing models and comparing their qualities of fit was long-winded, as well as having the disadvantage that human-chosen models could potentially overlook successful models that would fit the intensity distribution well. For this reason, I wrote a program which could automate this process. The general process ran as follows:

1. Define a set of operations e.g. $+$, $\times$, cos, exp.
2. From this, create a set of starting function trees (see below).
3. For each function tree, fit the data using a minimisation algorithm and evaluate the quality of the fit, assigning a quality parameter.
4. Take the best e.g. 50% of the models and "breed" them (take pairs of two and swap some of their trees' branches to create new functions).
5. Introduce some low-probability random mutations (which attempt allow the solution to climb out of local minima of the search space).
6. Repeat steps 3-5 and terminate after some criterion has been reached (e.g. a certain number of "generations" has been reached).

The trees used in the program were defined recursively, with each node containing information about its value, as well as its child nodes. If the value is an operation which requires two variables, e.g. $+$, it has two children. Other nodes containing operations such as cos and exp have just one child. Nodes containing variables, i.e. what is passed to the function to return an output, have no children, and neither do nodes containing model parameters. The tree is traversed as shown in Fig. 3.

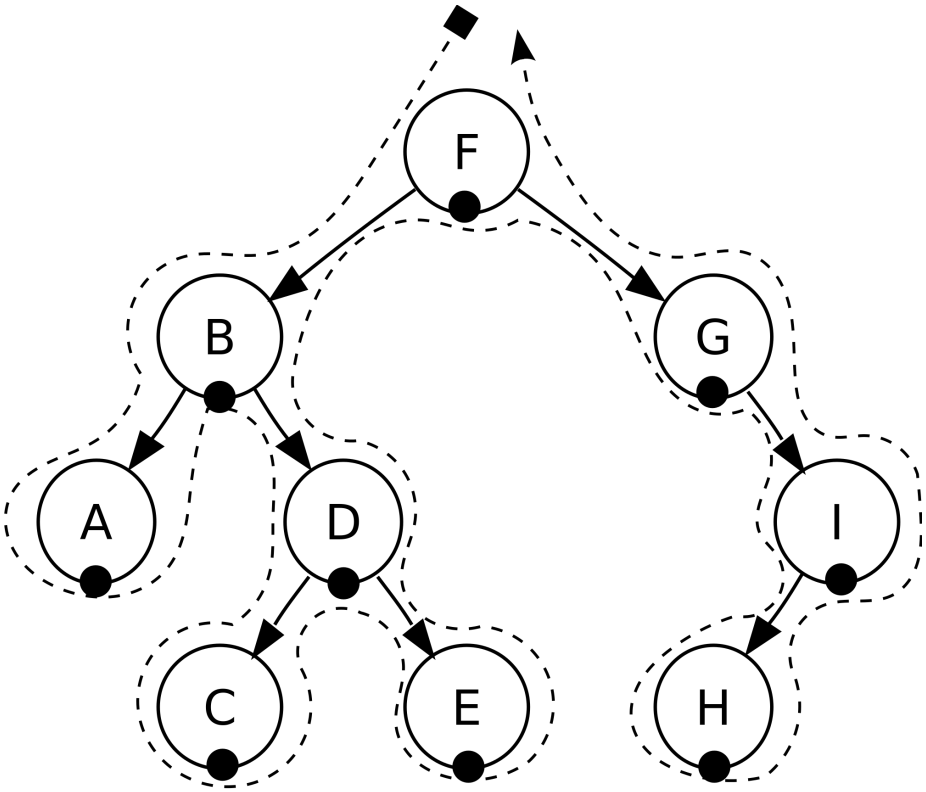


Figure 3: Illustration of the tree traversal method used in my genetic algorithm.

For example, the function $e^x \cos(ax) + \frac{x}{b}$ would be represented by as in Fig. 4.

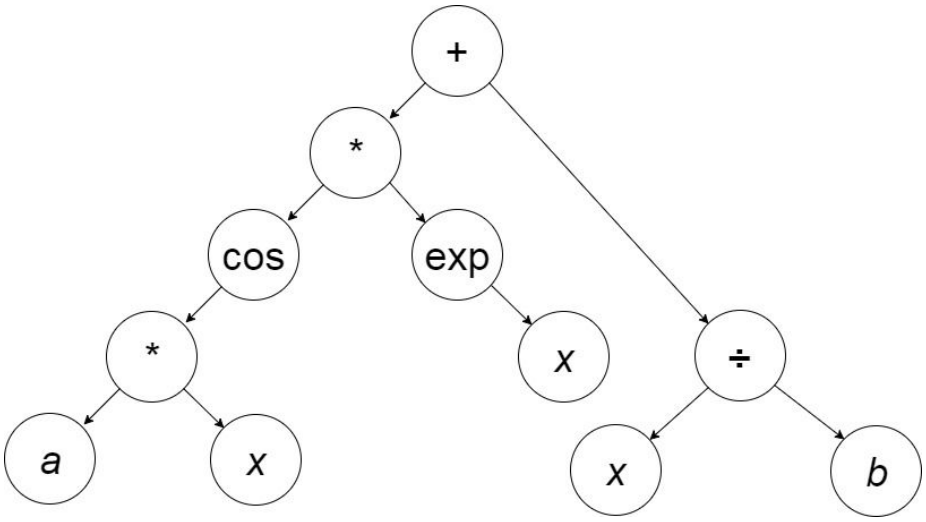


Figure 4: An example function represented as a tree.

Although relatively conceptually sound, this technique had a number of problems. The main downfall was that it was very slow to run, and would require many generations before even a half-decent fit could be obtained. Another was the problem of "inbreeding", i.e. that the best functions were often so similar that a list of almost identical functions was returned as the end result, none of which fitted the data well. Increasing the probability of random mutations didn't seem to help with this much. However, even if the algorithm had been successful, there was a risk of overfitting to the sample data if overly-complex models had been returned.

1.6 Neural Networks

Finally, I considered using a neural network to fit the intensity distribution. I was looking at supervised machine learning, which, in brief, entails feeding in a data set with known solutions (a good example would be a set of pictures of handwritten characters, each coming with information about the correct letter), and continually updating a model (the neural network). The model can then be applied to unknown data (recognising a handwritten letter in the handwriting example). As the intensity distribution data I was using had no specified "correct" answer, this method was not applicable. Unsupervised machine learning, on the other hand, may have been applicable, but I didn't spend a lot of time looking at this as I didn't want to get too carried away.

2 Analysis of Anemometer and Windcube Data

2.1 Introduction

Four anemometers (three of which we were able to obtain data from) were placed along the line-of-sight (LOS) of the lidar beam in order to verify the windspeeds measured from the lidar. These were placed next to markers along the LOS, and the distances from the lidar measuring equipment were measured using a rangefinder.

Additionally, a Leosphere Windcube was used to measure LOS windspeeds to a higher precision than the anemometers.

2.2 Anemometer Details

Device	Height of Sensor (cm)	Laser Beam Height (cm)	Distance to Laser Beam (cm)	Distance from Telescope to Anemometer (m)	Time Offset (s)
TopSonic 2	200	97	153	67.2	-6*
Svantec 181	180	100	165	75.0	-2
Svantec 174	150	105	165	91.9	0

* ignoring time zone difference of +1 hour

2.3 Windspeed Plots

The anemometers' internal clocks had slight discrepancies between them. In order to synchronise them for the plotting, a single loud clap was made during measurements, which could be read as a spike on the anemometers' sound files. From this, we were able to calculate the time offsets of the anemometers relative to one another. An example of how this is done for the sound files is shown in Fig. 5.

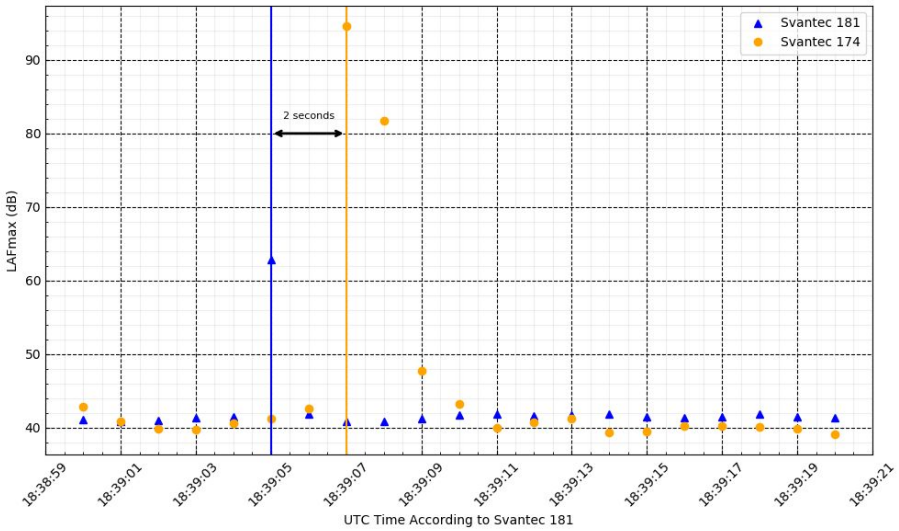


Figure 5: Demonstration of the determination of the time difference between the anemometers' internal clocks.

The anemometer data included both windspeed and wind direction. Using the wind direction, we were able to calculate the LOS windspeed component for each of the anemometers. The change in anemometer wind direction over time is shown for one day is shown in Fig. 6.

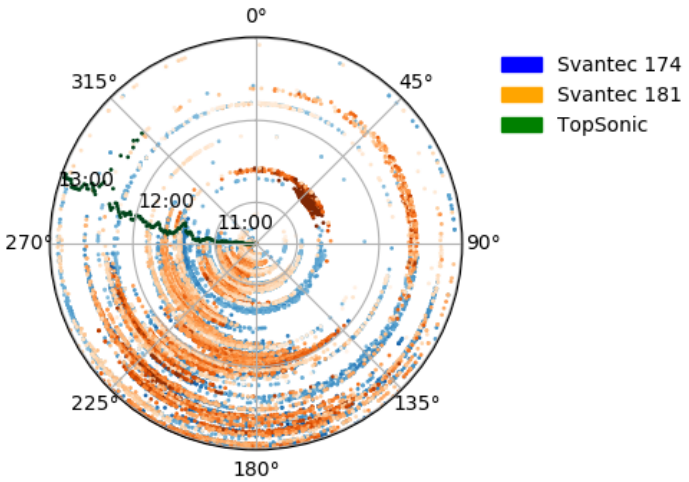


Figure 6: Variation of the windspeed (represented by colour darkness) and wind direction over time on 18th July. The angle is defined with respect to the LOS, with 0° representing the LOS.

Initially, the LOS windspeed for each anemometer was calculated by taking the component of the measured windspeed with respect to the wind direction measured by that anemometer. However, this was found to produce results inconsistent with the Windcube data. We therefore only used the wind direction from one of the anemometers (the TopSonic), which can be seen to be much less variable in the plot above, most likely due to averaging over the course of a minute instead of giving a wind direction every second. The plot for each anemometer is shown in Fig. 7.

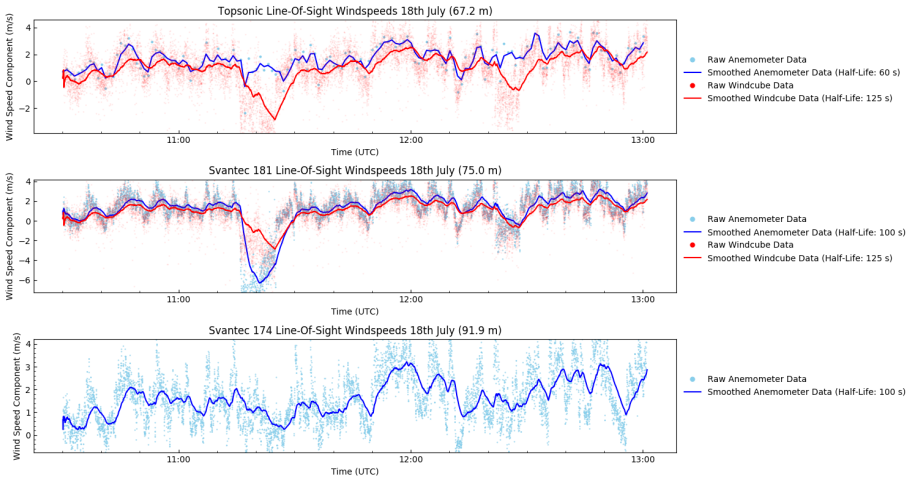


Figure 7: The variation in LOS windspeed recorded by the instruments on 18th July.

2.4 Windcube Windspeed Data

In order to extract the relevant data from the Windcube data, programs were written to obtain the LOS windspeeds and timestamps given start and end times, with a separate program to read in the data from the Windcube netCDF files. The Windcube beam ended in a beam dump, and hence the signal beyond this point (around 90m) had a very high SNR. The software that calculates the windspeeds from the raw spectra prevents windspeeds being calculated from highly noisy data, as shown in Fig. 8.

Additionally, for interest, attempts were made to manually calculate the windspeed and its variance from the raw spectra. The frequencies were determined for each pixel along the x-axis and the noise spectrum was subtracted from each spectrum. A Gaussian was fit to the resulting spectrum, giving the results shown in Fig. 9.

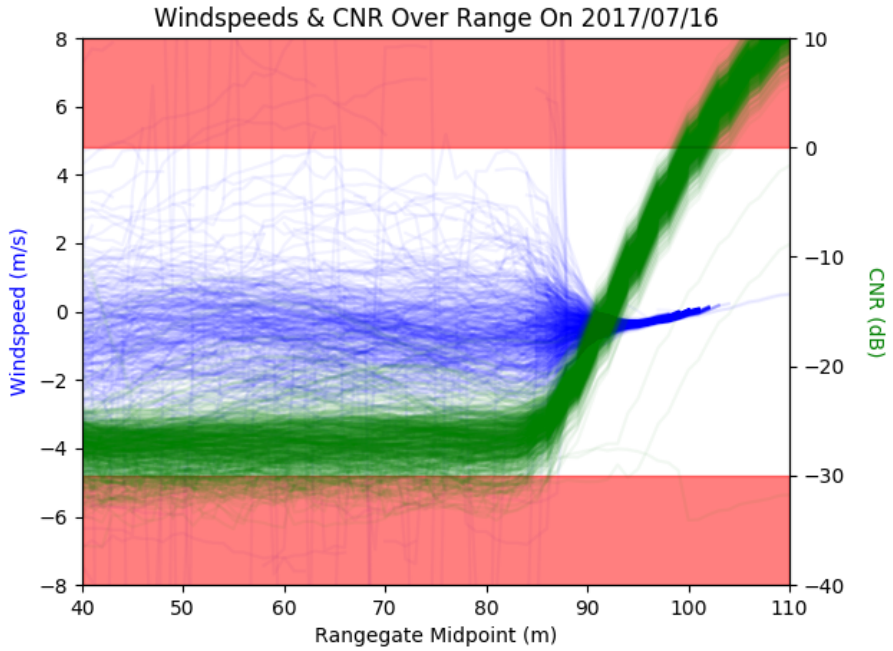


Figure 8: Plot of windspeeds and corresponding CNR over range for many times on 15th July. The red regions show the boundaries in the CNR beyond which the windspeeds are not calculated.

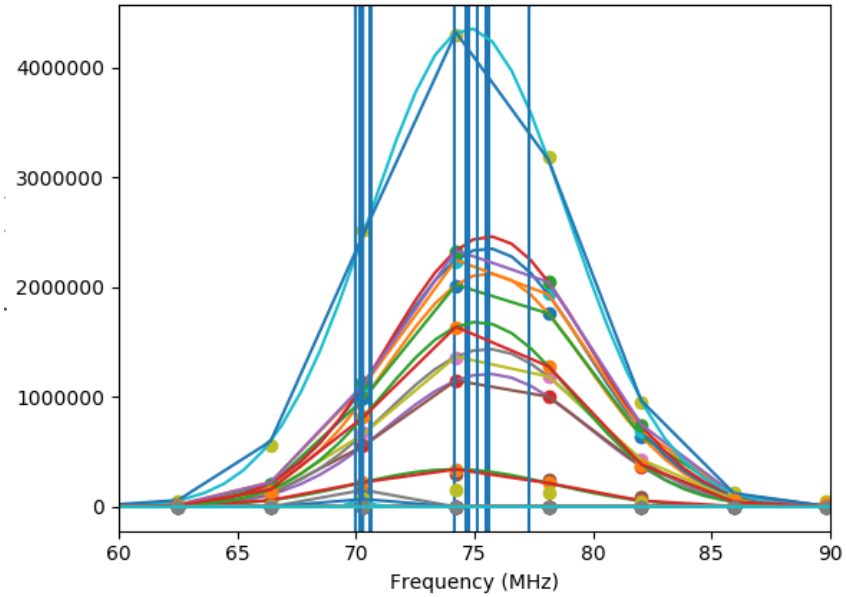


Figure 9: A sample of raw spectra from the windcube with fitted Gaussians and determined mean frequencies represented by vertical lines.

From the Gaussian fit, the mean frequency and the variance could be read out. The mean frequency is used to obtain windspeeds shown in Fig. 10.

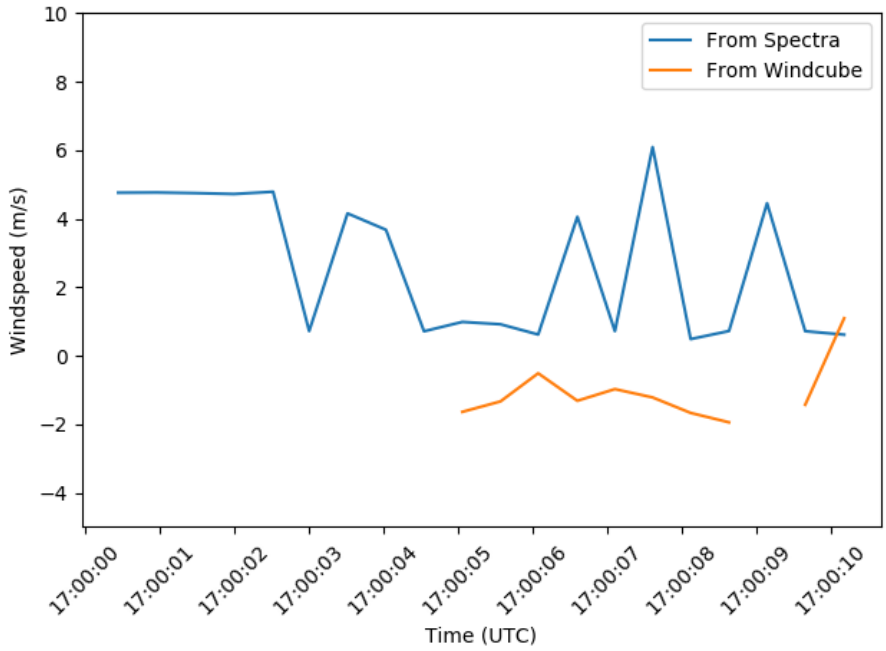


Figure 10: Windspeeds determined from the sample of raw spectra and windspeeds returned by the Windcube software for the same times (some of which not included due to low SNR).

3 Analysis of the Lidar Data

3.1 Introduction

The data from the lidar was stored in netCDF files and I was interested in looking at how the pulse intensity varied over range. The original files included the intensity data for the time between pulses, which was translated into unrealistic ranges of thousands of kilometres. By cutting down the data such that the any signal registered from less than -50 m or greater than 200 m was omitted, the data was cut down from around 100 GB to less than 0.6 GB.

3.2 Pulse Averaging

As the lidar pulses were in such rapid succession, it is advantageous to average over many pulses. This has been done in the Fig. 11 with the standard deviations shown as a filled-in region around each line.

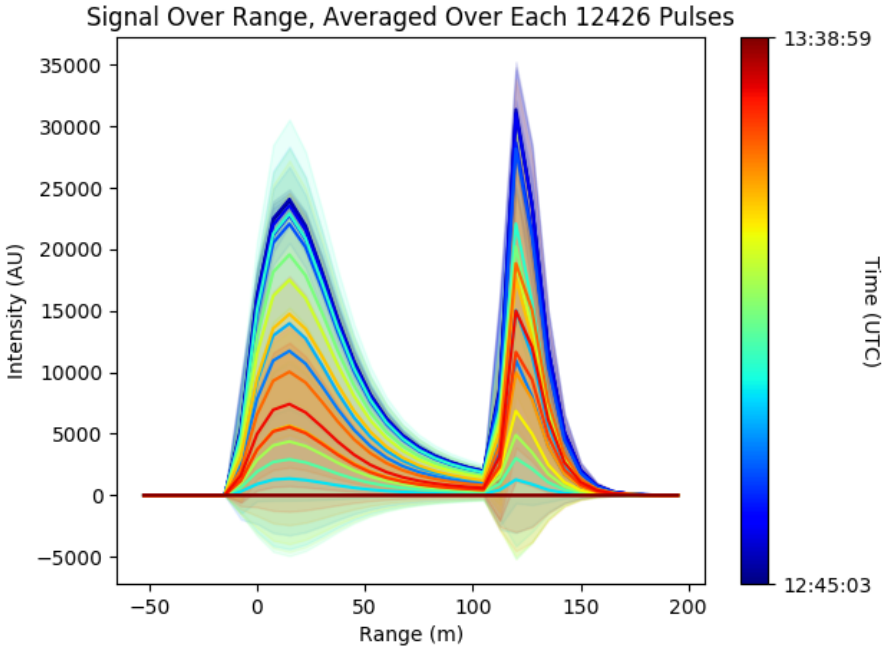


Figure 11: Lidar signal intensity over range for several times.

Fig. 11 shows two curves that rise sharply and fall as $1/r^2$. The first is due to the light emitted from the lidar; the second is due to small amounts of light

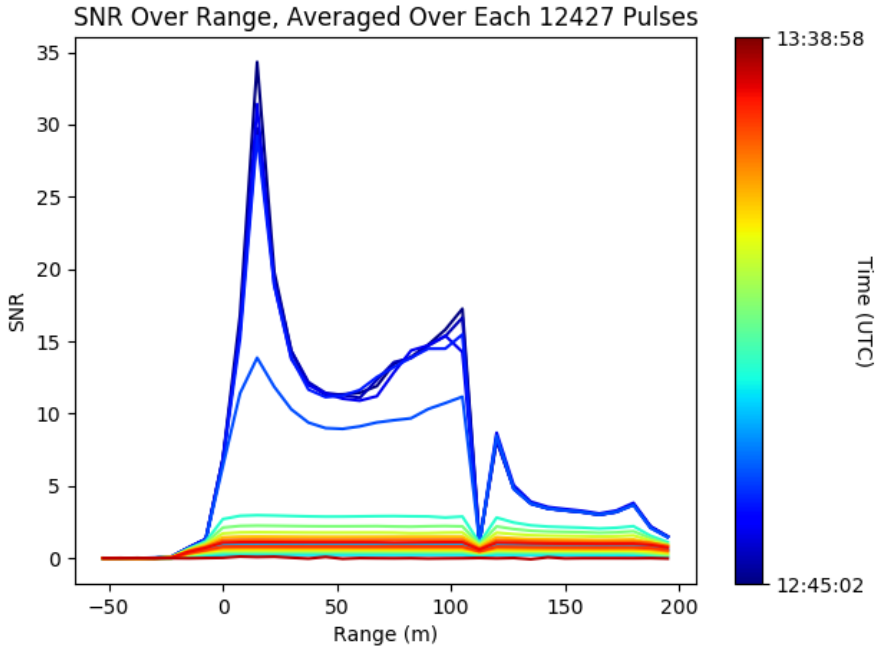


Figure 12: Lidar SNR over range for many times.

reflected from the beam dump. We can also look at the SNR of the signal over range, as shown in Fig. 12.

4 Other Aspects of the Internship

4.1 Timestamp Buffer Transfer Program

The concept of memory buffers was new to me, so I spent a significant amount of time trying to understand how the data was structured on the memory cards. I also spent a fair amount of time getting to grips with the pyspcm library.

The main struggle with this task was in obtaining the timestamps from the buffer (I managed to obtain the signal information from the 16 channels). In attempting to read out the timestamps, the program returned zero. I emailed the manufacturers and they sent me an example program for reading out the timestamps. This produced an error message when I tried to implement it, but I was unable to figure out what was causing it as I had to do something else shortly after it came up, and I didn't get the chance to come back to it. With luck, the error is trivial and the program will function correctly once it is fixed.

4.2 Conversion of Messung.py to a Python Library

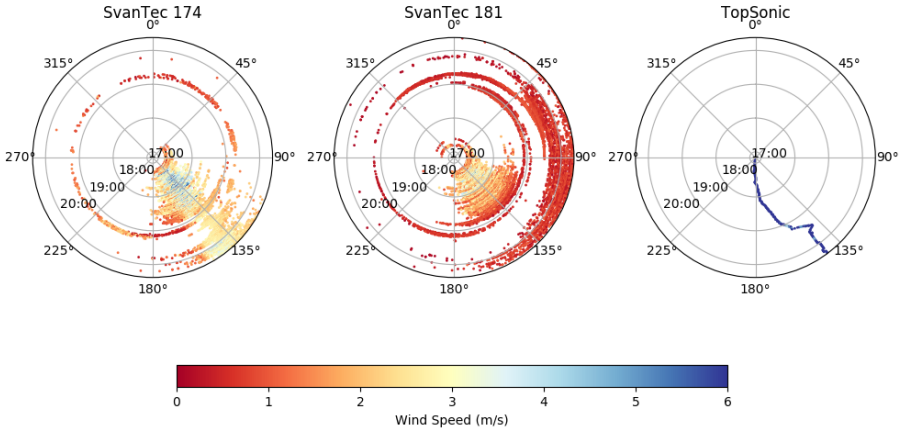
The program used for the bulk of the analysis, "Messung.py", serves many purposes. The control flow is determined by activating/deactivating flags for certain pieces of code to run, which can make the program confusing to read and use. I attempted to remedy this by taking the various aspects of the program and converting them to functions that can be imported, such that the required pieces of code are run by simply calling the relevant function. However, it turned out that the same flags cropped up in various, seemingly unconnected regions of code which rendered building a sensibly-structured library difficult.

As of now, I have included some of the more basic functions in the library, but completing it will likely be a significant time investment and will require careful planning.

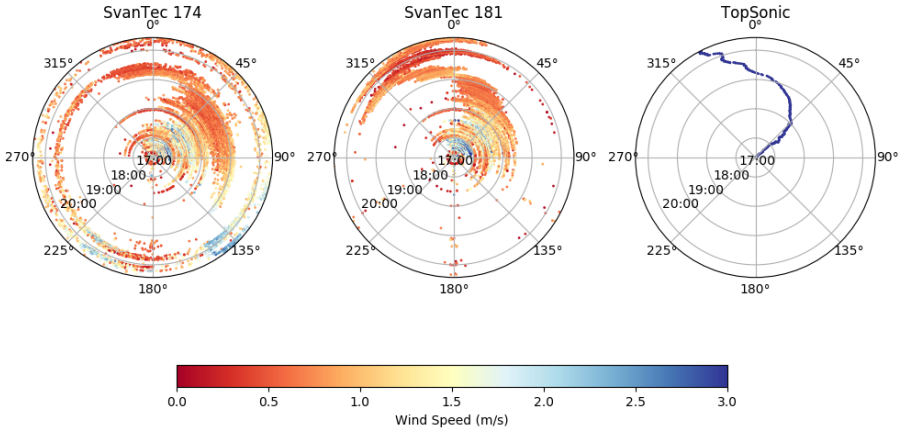
5 Other Plots

5.1 Individual Wind Direction Plots

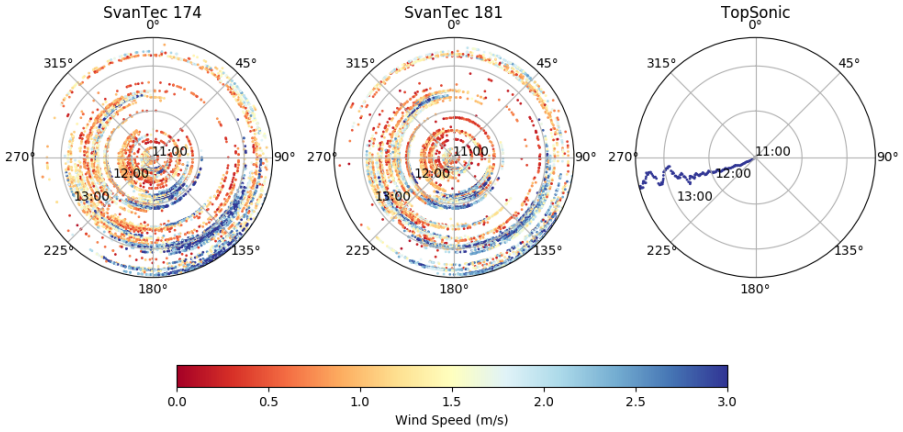
5.1.1 15th July Wind Directions



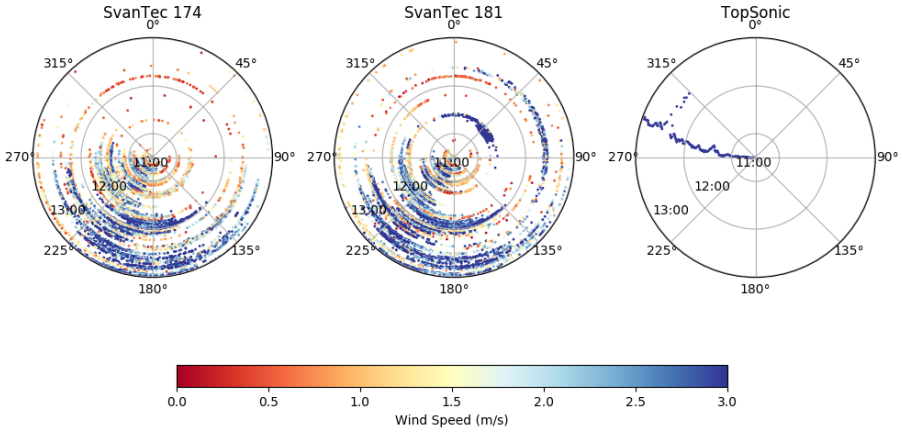
5.1.2 16th July Wind Directions



5.1.3 17th July Wind Directions

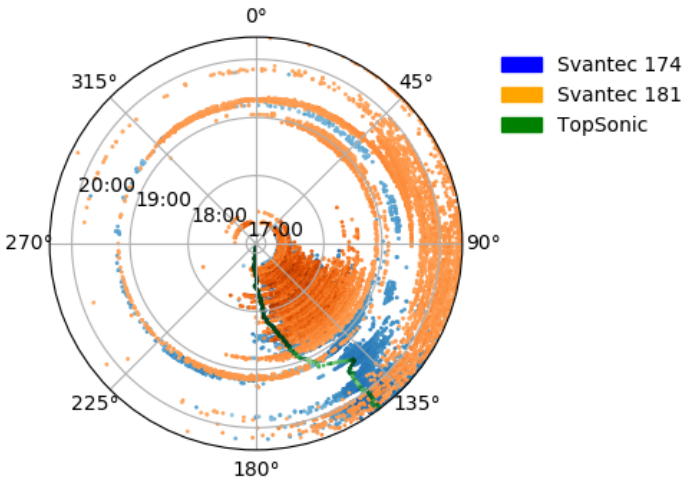


5.1.4 18th July Wind Directions

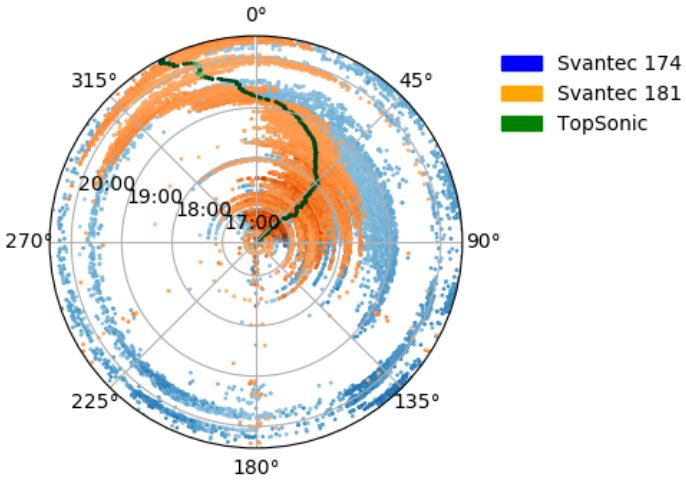


5.2 Combined Wind Direction Plots

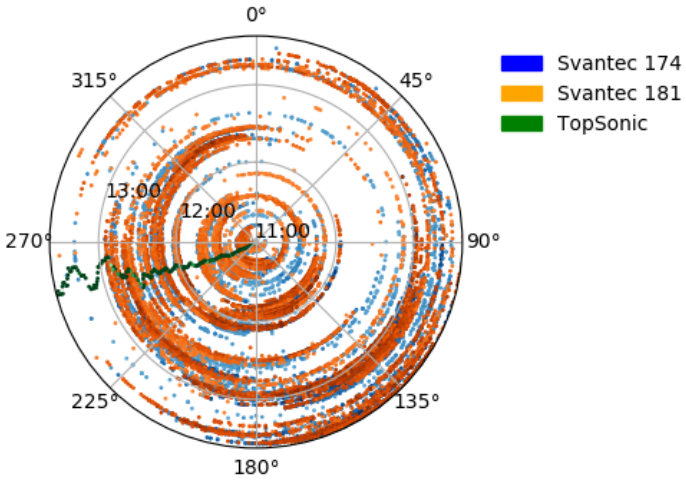
5.2.1 15th July Wind Directions



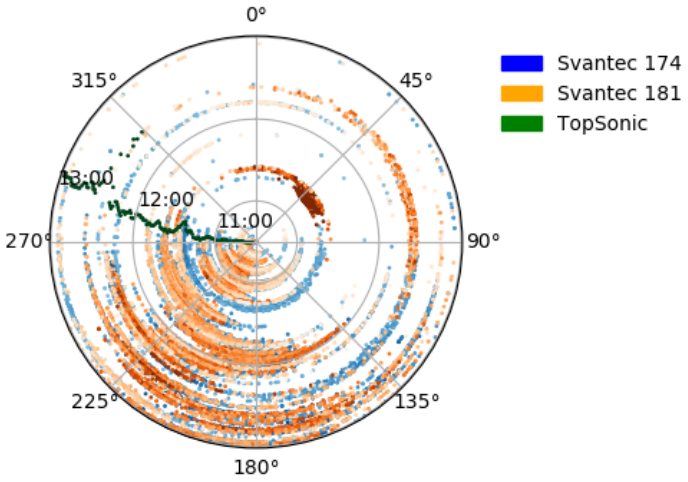
5.2.2 16th July Wind Directions



5.2.3 17th July Wind Directions

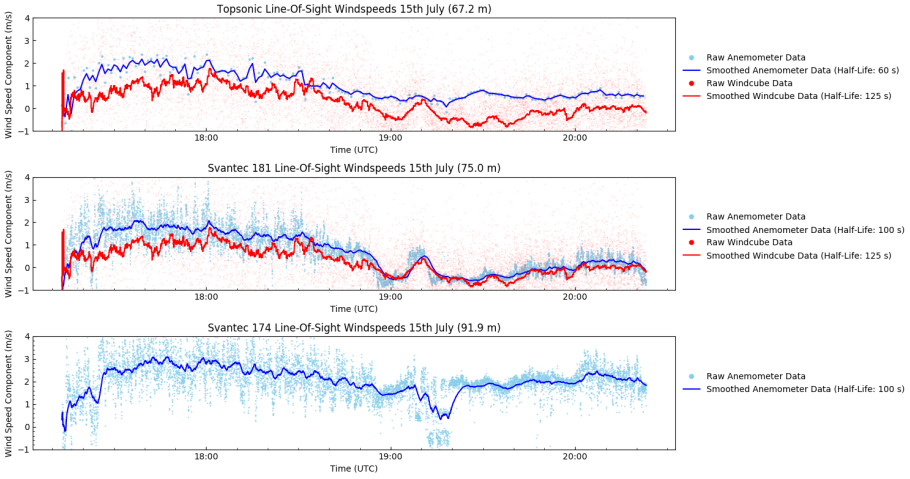


5.2.4 18th July Wind Directions

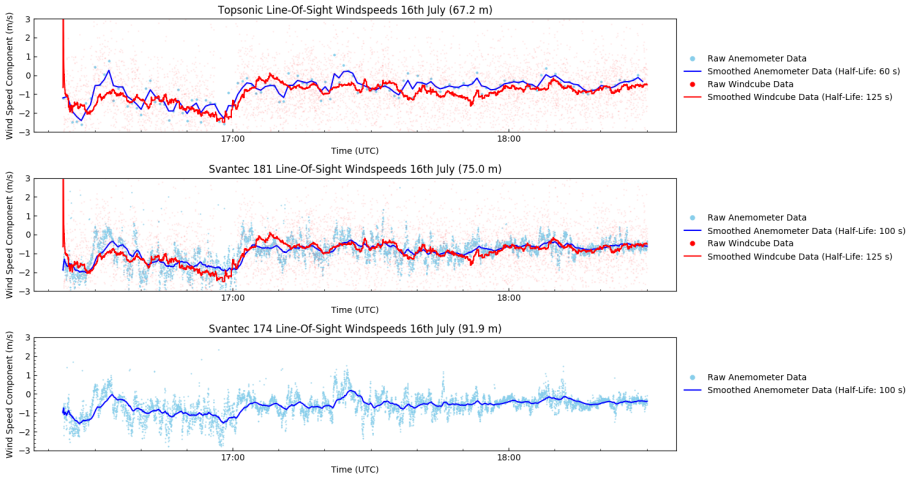


5.3 LOS Windspeed Plots

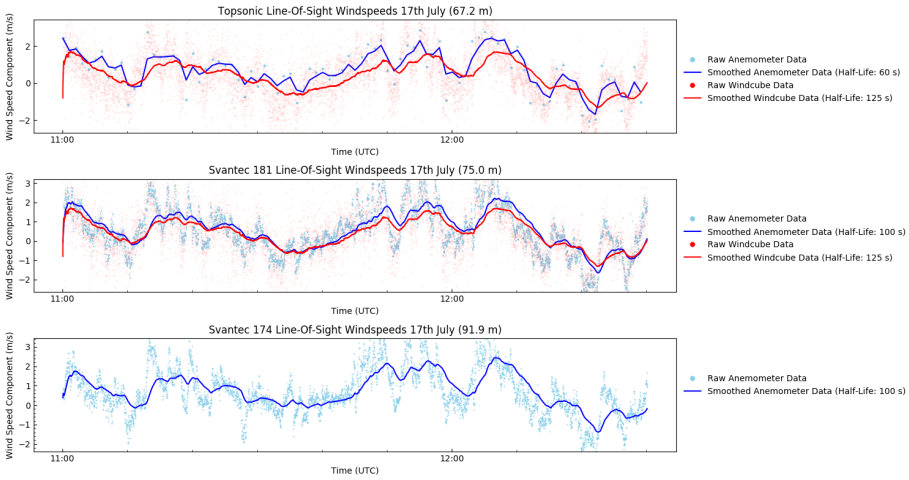
5.3.1 15th July LOS Windspeeds



5.3.2 16th July LOS Windspeeds



5.3.3 17th July LOS Windspeeds



5.3.4 18th July LOS Windspeeds