

UNIVERSITY OLDENBURG

WIND PHYSICS MEASUREMENT PROJECT

---

## Exercise 4 - Lidar measurements

---

*Author:*

Jan KÄMPER

Florian BÖRGEL

*Supervisor:*

Marijn Floris VAN DOOREN

Julian HIERONIMUS

July 2, 2016

## Contents

<b>1</b>	<b>Task 1: Spectral analysis: from backscatter spectrum to line-of-sight velocity</b>	<b>2</b>
1.1	Clean the spectra from their background noise . . . . .	2
1.2	Calculate the Doppler frequency that each bin of the spectra corresponds to . . . . .	3
1.3	For each spectrum, define the peak location with the centroid method . . . . .	3
1.4	Correlate your calculated line-of-sight speeds . . . . .	5
<b>2</b>	<b>VAD Scanning: from line-of-sight velocity to wind factor</b>	<b>6</b>
2.1	Investigate the LiDAR data . . . . .	6
2.2	Separate the data single 360° scans and carry out a cosine fit . . . . .	7
2.3	10 minute averages and comparison of vertical wind profiles and wind directions . .	9

## Introduction

The fourth and last exercise deals with Lidar measurements. The exercise itself was separated into three sub tasks. First we received 312 frequency spectra which span a vertical plane in front of the SpinnerLidar. For each spectra we calculated the peak and the corresponding line-of-sight velocity.

## 1 Task 1: Spectral analysis: from backscatter spectrum to line-of-sight velocity

### 1.1 Clean the spectra from their background noise

In order to clean the 312 different spectra from their individual background noise we calculated the mean values of each spectra. Keeping in mind that the goal of this exercise is to identify the peak location, we defined  $1.5 \cdot \text{mean}$  multiplied with the mean of the spectrum as noise. Therefore we subtracted  $1.5 \cdot \text{mean}$  and discarded all values below zero.

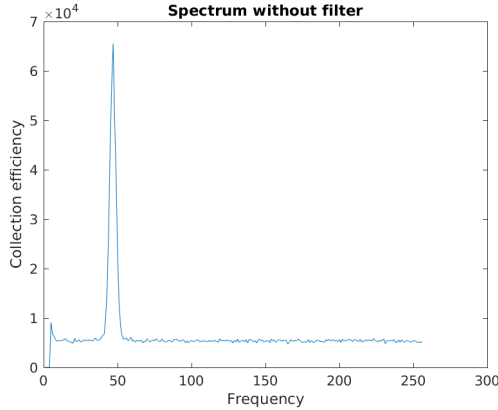
The following code-snippet shows this procedure in Matlab:

---

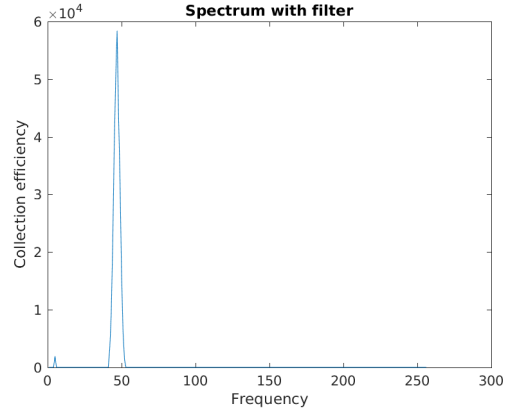
```
1 for pos = 1:312
    spinner_noiseCancelled(pos,:) = spinnerlidar_spectra(pos,:) - 1.5*mean(
        spinnerlidar_spectra(pos,:));
3     spinner_noiseCancelled(spinner_noiseCancelled < 0) = 0;
end;
```

---

The following figure shows on the left the raw spectrum and on the right the noise cleaned spectrum. As one can easily see we obtain different values for the collection efficiency, but since it is only important to find the corresponding frequency and subsequently the line-of-sight velocity of the peak, the peak value itself is not relevant for further calculations.



(a) Raw spectrum



(b) Spectrum cleaned from background noise

## 1.2 Calculate the Doppler frequency that each bin of the spectra corresponds to

We received the spectra as histograms which are separated in 256 bins. The measured bandwidth is  $25 \cdot 10^6$  Hz. Therefore each bin represents a frequency of

$$frequency = x * \frac{25 \cdot 10^6 Hz}{256}$$

where x is the bin number. With the equations provided during the lecture we were also able to calculate the line-of-sight velocity. The implementation in Matlab is shown in the following.

---

```

for bin=1:256
2   f_d(bin,1) = (bin-1)/bins*bandwidth;
   v(bin,1) = f_d(bin,1) *lambda_0;
4 end;

```

---

## 1.3 For each spectrum, define the peak location with the centroid method

The centroid method was introduced during the lecture and is defined as:

$$f_{peak} = \frac{\int f_d \cdot p(f) df}{\int p(f) df}$$

Before we calculated the peak values we normalized our data to get a PDF instead of absolute values. Since we are dealing with discrete values, we summed over all values in order to identify the peak

value. In order to check all 312  $f_{peak}$  values, we first calculated the index of the maximum collection efficiency for each spectra. The maximum should always represent the peak of the spectrum. Next we determined the corresponding index of  $f_{peak}$ . Now we were able to compare the distance of the calculated indices. The results are shown in figure 2.

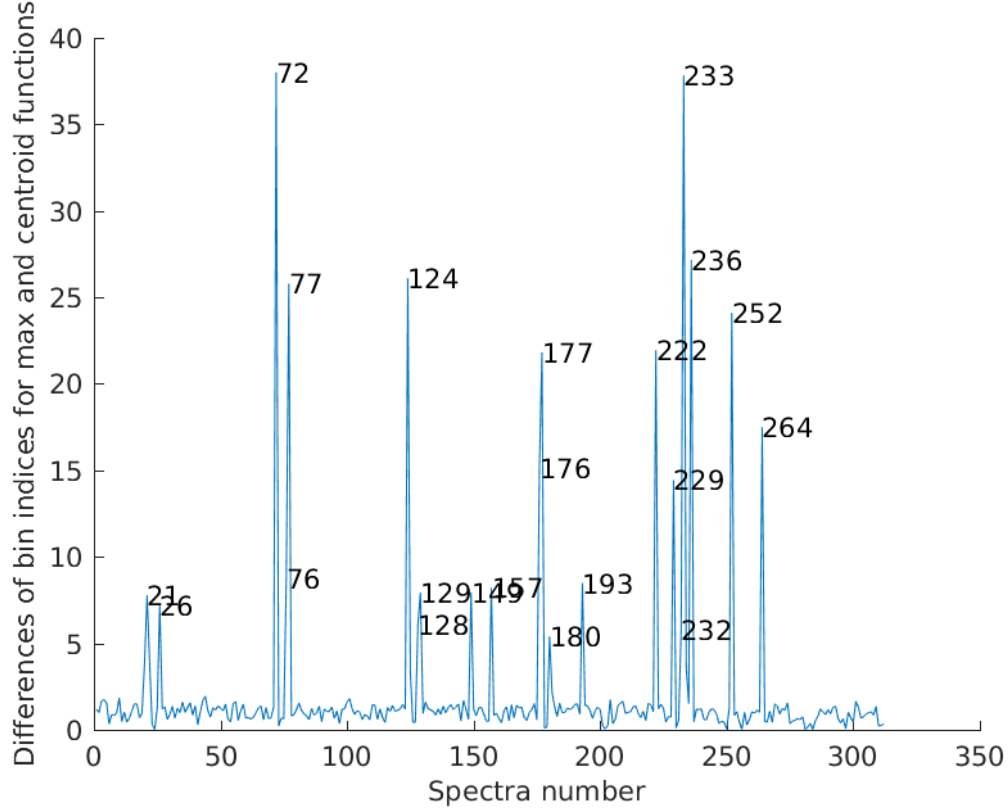


Figure 2: Centroid Errors in bin distance

The figure shows that there are several errors. The reason why the centroid method fails for some spectra is due to a measurement error. As stated in the exercise sheet the lidar was placed on the nacelle of a wind turbine. Therefore for some measurements the lidar might be blocked by the rotating blades which would result in a second frequency peak. If the spectrum contains two peaks, the centroid method returns the middle of these two peaks and is not applicable for this case. One of those cases is shown in figure 3.

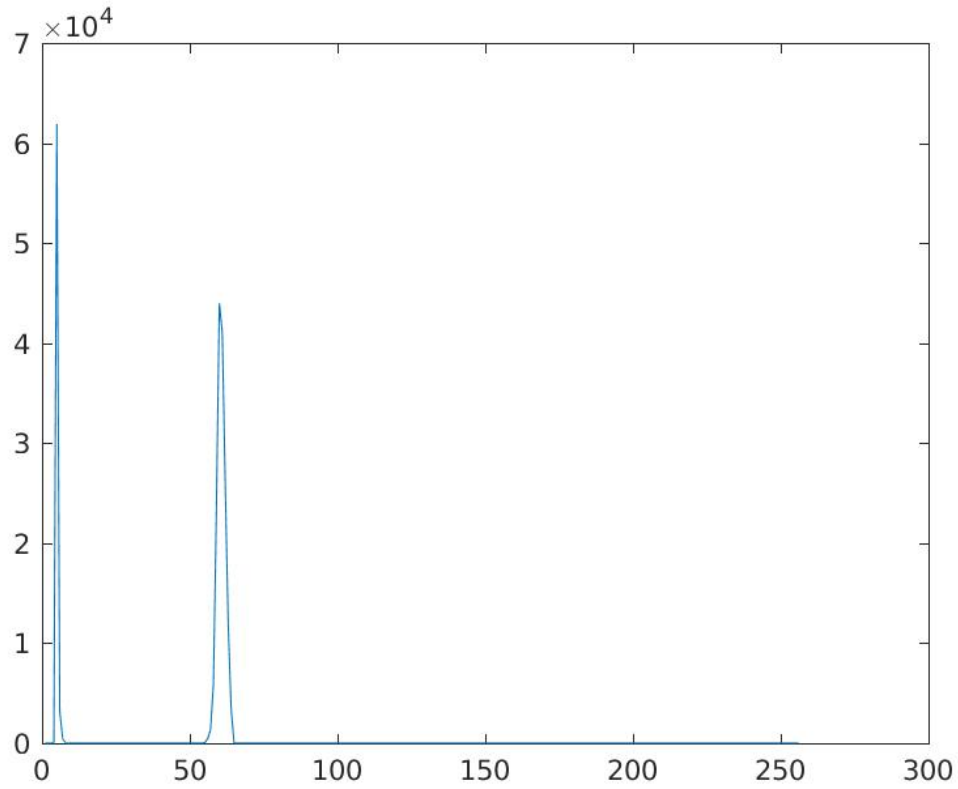


Figure 3: Measurement error in spectrum 72

#### 1.4 Correlate your calculated line-of-sight speeds

In step 4 we were asked to correlate the calculated line-of-sight speeds with the lidar data itself. The correlation method was already provided. Therefore we only needed to calculate the line-of-sight velocity by multiplying  $f_{peak}$  with  $\lambda_0$ . The resulting plot is shown in figure 4.

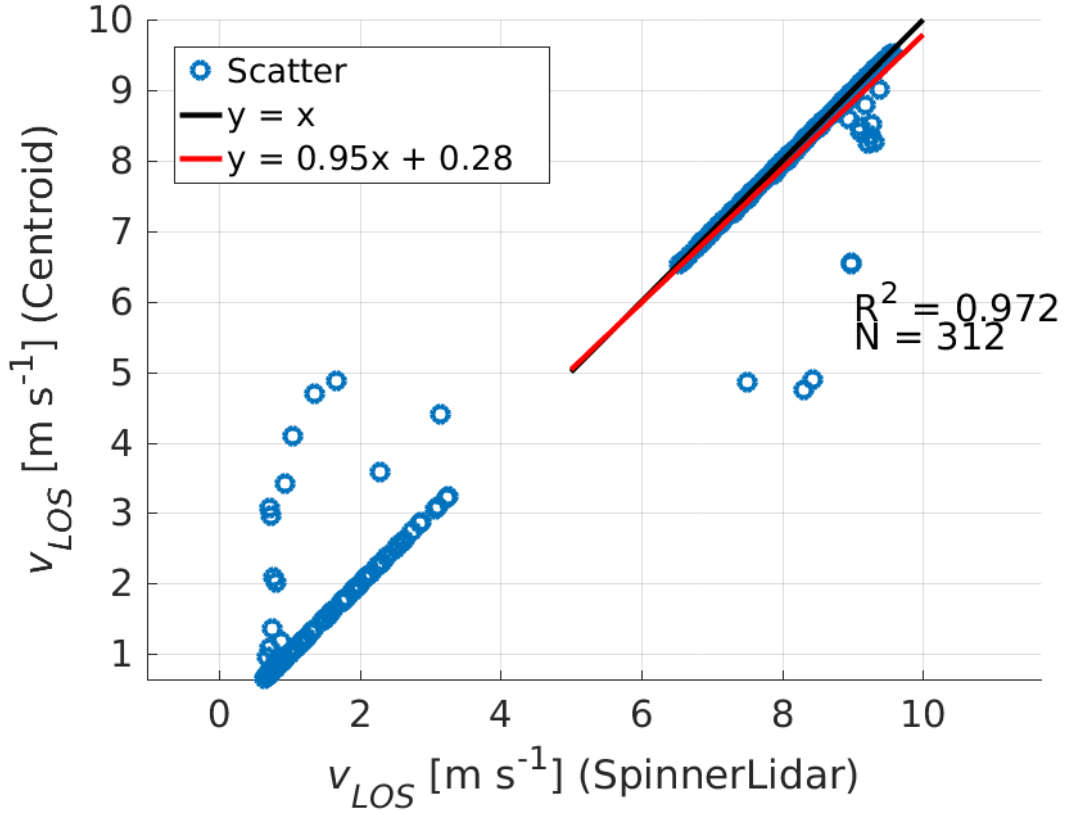


Figure 4: Correlation of the calculated line-of sight-speed and the lidar data

Figure 4 shows a strong correlation of  $R^2 = 0.972$ . As discussed in section 1.3 there are measurement errors due to the position of the lidar. Some measurements are blocked by the moving blades.

## 2 VAD Scanning: from line-of-sight velocity to wind factor

### 2.1 Investigate the LiDAR data

In exercise 2 we were working with a 6 hour dataset of VAD Lidar measurements located on the FINO1 platform. For further comparison we were also provided a 10 minute averaged dataset of FINO 1. The first step was to filter bad measurements. A common procedure is to filter bad Carrier-to-Noise-Ratios (CNR). The procedure is shown in the following code-snippet:

---

```
% Filter data by CnR, by setting to NaN and later in Step 2 discarding NaN
2 % values
```

```

    cnr_ind = find(cnr_VAD <= -20 | cnr_VAD >= -5);
4 rs_VAD(cnr_ind) = NaN;
  az_c_VAD(cnr_ind) = NaN;

```

---

After filtering the data we were asked to separate the data for the different range gates. We identified 14 different range gates, measuring at 61m, 64m, 67m, 70m, 72m, 75m, 78m, 81m, 84m, 87m, 90m, 93m, 96m and 98m. The last task of 2.1 was to calculate the time for a full 360° scan. The LiDAR has a scanner speed of 25°/s. Therefore one full scan takes:

$$time\ per\ scan = \frac{360}{25^\circ/s} = 14.4s$$

The given data is measured at a time interval of 0.4 s. A full scan represents  $14.4s/0.4s = 36$  intervals.

## 2.2 Separate the data single 360° scans and carry out a cosine fit

The goal of this section is to compute the horizontal wind speed, the horizontal wind direction and the vertical wind speed. In the last section we calculated that a full scan is represented by 36 intervals. So in the ideal case of a homogeneous atmosphere the line-of-sight velocity should show a sine-like behaviour. Therefore we should be able to fit this to a function of type:

$$v_{rad} = b \cdot \cos(\phi - \Theta) + a$$

where

$$v_{hor} = \frac{b}{\cos(\theta)} \quad w = \frac{-a}{\sin(\theta)} \quad D = b \pm 180$$

To implement the fitting routine in Matlab we first defined the function, depending on the three parameters  $b$ ,  $\Theta$  and  $a$  and the azimuth angle  $\phi$ . During the lecture the function `lsqcurvefit()` was recommended as the fitting routine. It should be noted, that the function is not able to deal with *NaN* values. Therefore *NaN* values had to be discarded temporally. The following code-snippet shows the Matlab implementation. The fitting was done over all full 360° scans and over all range gates.

---

```

1 for j = 1:14 % number of Range Gates
    for i = 1:numberOfScans
3         phi = az_c_VAD_filter((i-1)*intervalls_per_scan+18:(i-1)*intervalls_per_scan
            +53,j);

```



```

        v_r = rs_VAD_filter((i-1)*intervals_per_scan+18:(i-1)*intervals_per_scan
+53,j);
5     nanIndices = isnan(phi) | isnan(v_r);
        phi(nanIndices) = [];
7     v_r(nanIndices) = [];
        if length(phi)>0
9         fitparam = lsqcurvefit(VADCos, startvalues, phi, v_r);
            savedparam{i,j} = fitparam;
11        %calculate horiz and vertical speed
            v_hor = fitparam(1) / cosd(60);
13        v_ver = -fitparam(3) / sind(60);
            D = abs(fitparam(1)-180);
15        calc_times(i,j) = ts_VAD_filter((i-1)*intervals_per_scan+18,j);
            calc_vHor(i,j) = v_hor;
17        calc_vVer(i,j) = v_ver;
            calc_D(i,j) = D;
19    end
21 end
end

```

---

In order to understand what the fitting routine returns, we also plotted the data and the fitted curve. The results are shown in figure 5.

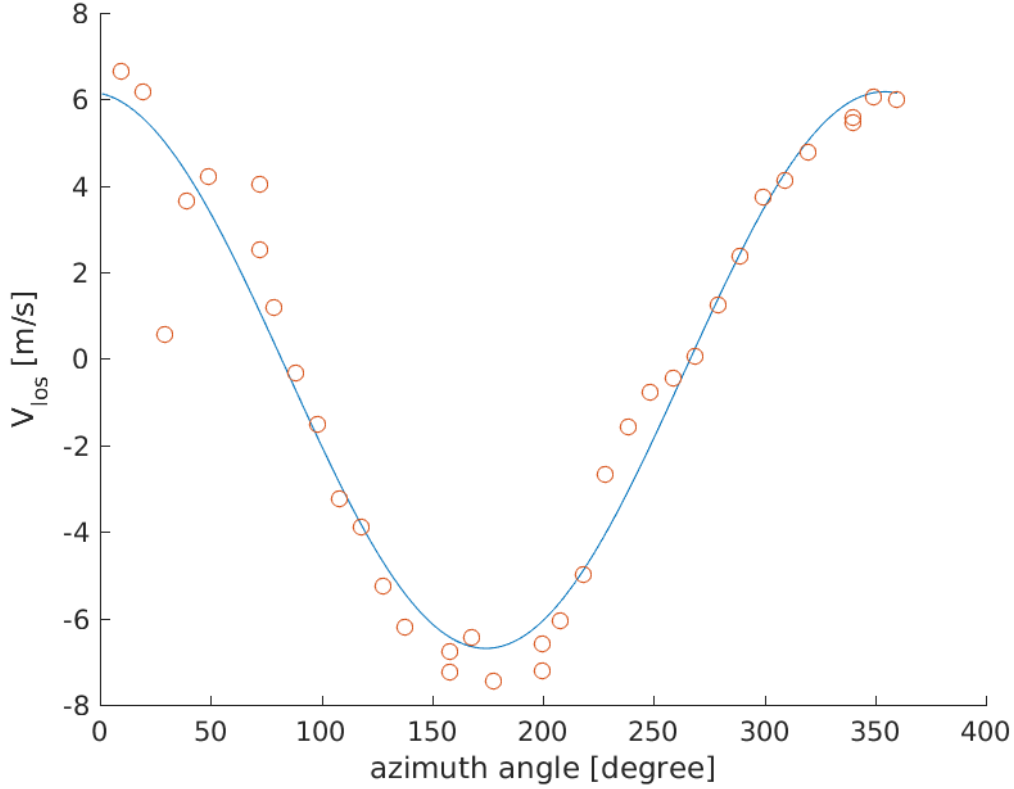


Figure 5: Full 360° for Range Gate = 61 m with fit

### 2.3 10 minute averages and comparison of vertical wind profiles and wind directions

The FINO 1 data includes wind vanes at heights of *33m, 40m, 50m, 60m, 70m, 80m, 90m* and eight anemometers at heights *33m, 40m, 50m, 60m, 70m, 80m, 90m and 100m*. The measurements are in 10 minute intervals. The LiDAR data was measured at heights *67m, 70m, 72m, 75m, 77m, 80m, 83m and 85m*. To compare the vertical profiles we had to compute 10 minute averages of the LiDAR wind speeds and wind directions. Since we only had a six hour dataset we searched for the corresponding time stamp in the FINO 1 data. After averaging the LiDAR data and extracting the correct FINO 1. We were now able to compare the datasets. In Exercise 2 - Energy Meteorology we identified the logarithmic wind speed profile as a good method to predict the vertical wind speed profile. Therefore we fitted the calculated LiDAR data and the given FINO 1 data with the help of logarithmic wind speed profile. The procedure for FINO 1 is shown following code-snippet:

---

```

1 logProfileModel = @(b,z) b(1)/0.4 * (log(z/b(2)));
  opts = statset('nlinfit');
3 opts.RobustWgtFun = 'bisquare';
  logProfileCoeffsFino = real(nlinfit([33,40,50,60,70,80,90,100], ...
5 avg_horSpeed_Fino, logProfileModel, [0.1, 10^-5], opts));

```

---

The results are shown in figure 6 :

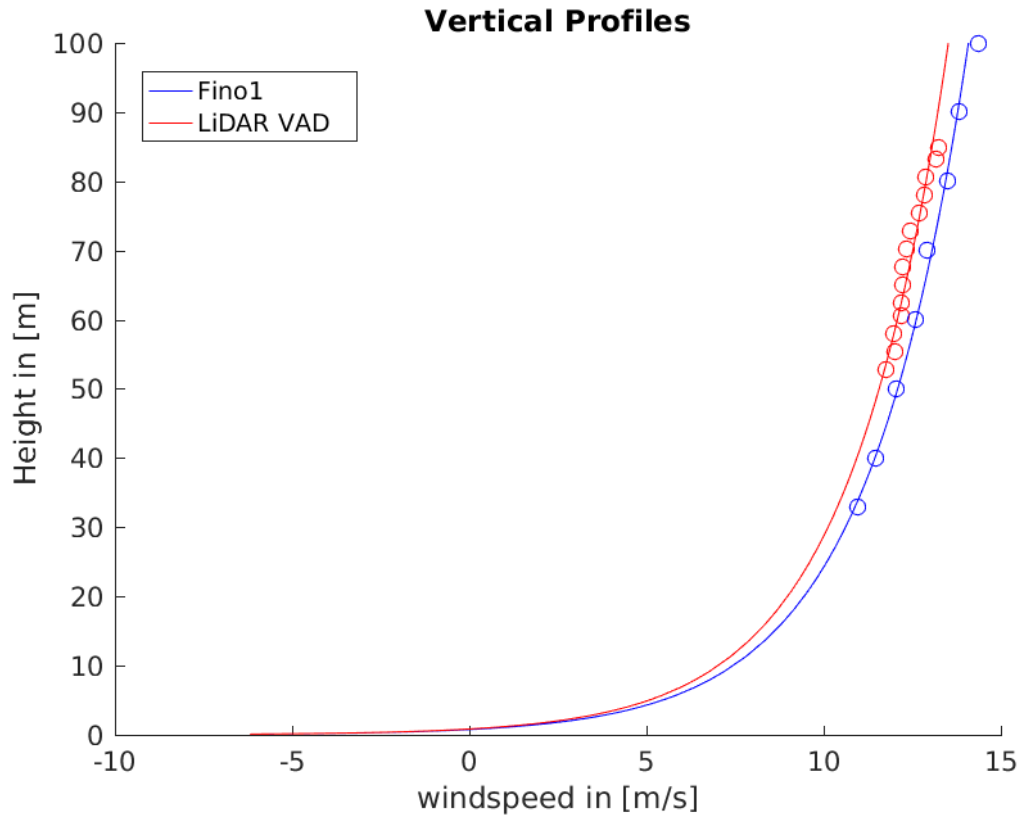


Figure 6: Vertical profile of FINO 1 and LiDAR measurement

The figures shows that the vertical profile of the LiDAR is shifted to the left. As discussed in the lecture, anemometers measure more accurately and therefore the FINO 1 curve should be more representative.

Further we also evaluated the wind directions. Here we also used the already existing WinRose.m routine. The wind rose helps to evaluate and compare the wind directions.

In order to obtain correct wind directions we used the following plot routine:

---

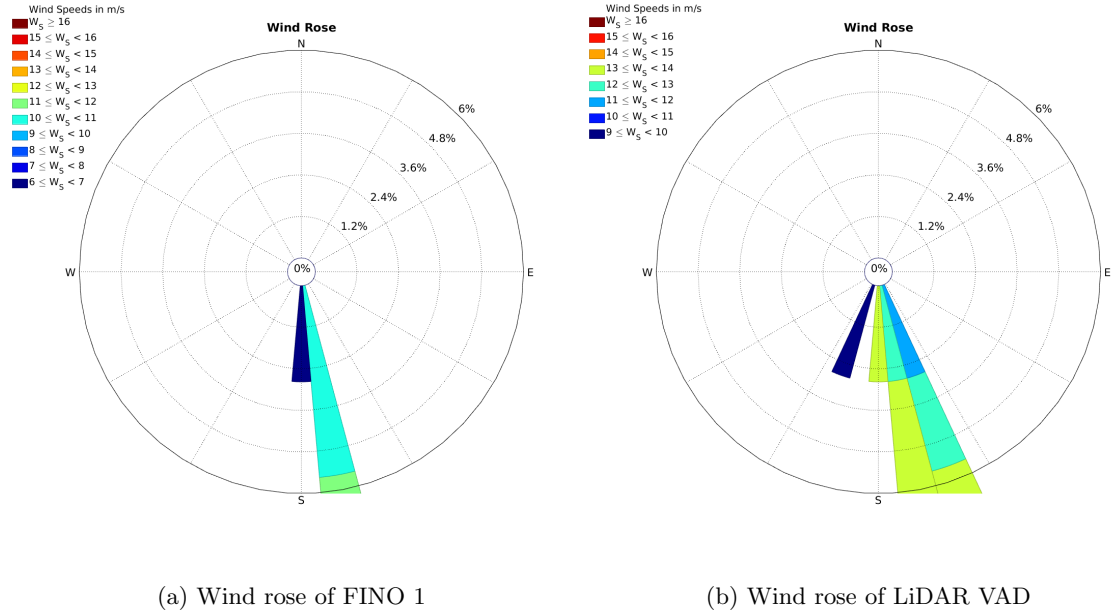
```

1 WindRose(fino_90_dir_interval ,fino_speeds (:,7) , 'AngleNorth' ,0 , 'AngleEast' ,90 , '
    freqlabelangle' ,45 , 'MaxFrequency' ,6) ;
WindRose(tenMinAvg_direction (:,7) , tenMinAvg_horSpeed (:,7) , 'AngleNorth' ,0 , 'AngleEast'
    ,90 , 'freqlabelangle' ,45 , 'MaxFrequency' ,6) ;

```

---

The comparison between the two wind roses are shown in figure 7.



(a) Wind rose of FINO 1

(b) Wind rose of LiDAR VAD

Figure 7: Comparison of wind roses