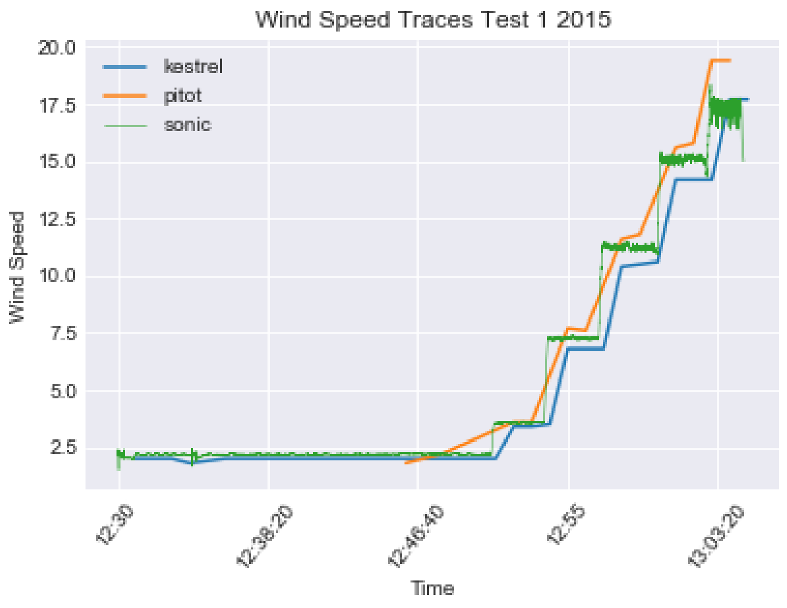
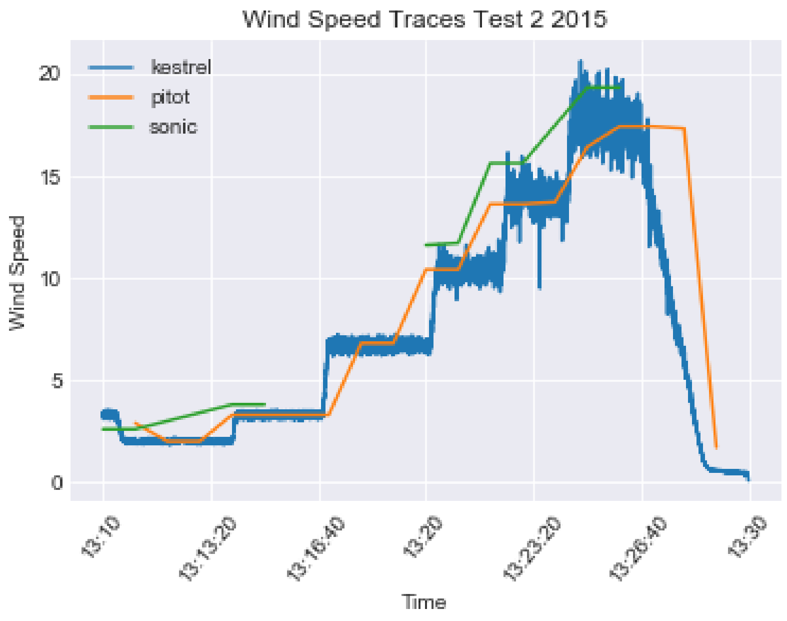
Pearl Ayem – 34404160

ATSC 303 LAB 8 – ANEMOMETRY

Part 1:

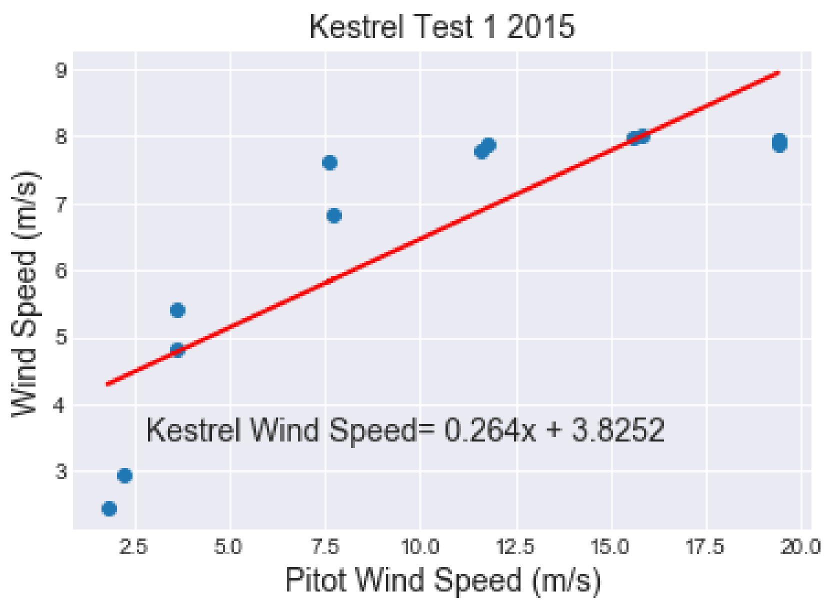
1. Wind traces for test 1 and 2 in 2015

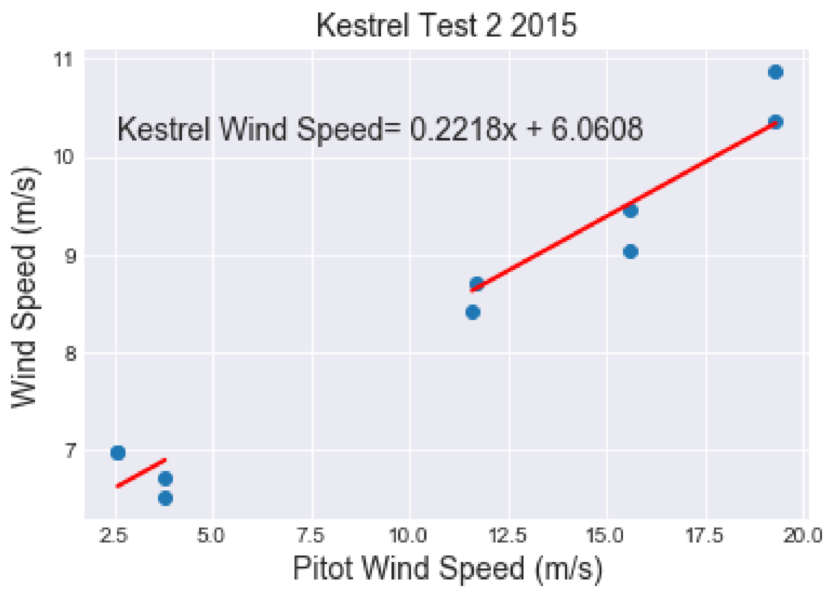




2. Calibration with Pitot wind speed

   a. Kestrel:





Transfer equation for test 1:

$$Kestrel\ wind\ speed = 0.246 * Pitot\ Wind\ Speed + 3.8252$$

Calibration equation for test 1:

$$Pitot\ Wind\ Speed\ =\ -0.069\ +\ 0.2614\ *\ Kestrel\ Wind\ Speed$$

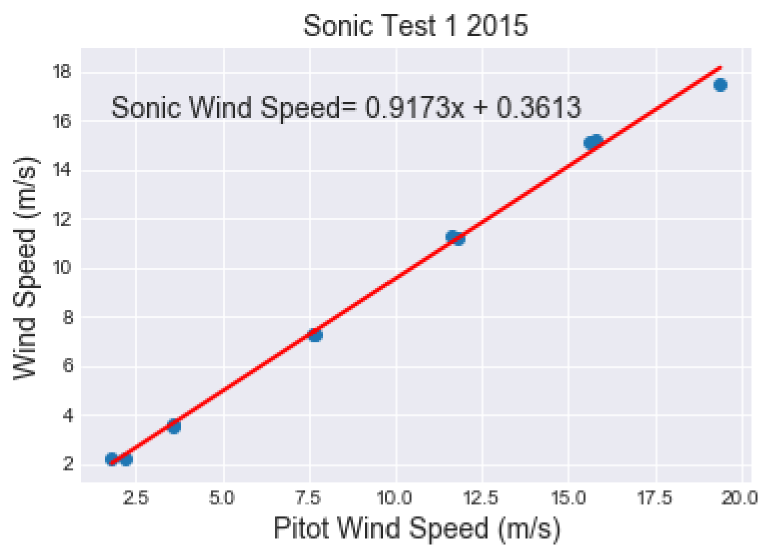
Transfer equation for test 2:

$$Kestrel\ Wind\ Speed = 0.2218 + 6.0608$$

Calibration equation for test 2:

$$Pitot\ Wind\ Speed = -0.0366 + 0.165 * Kestrel\ Wind\ Speed$$

b. Sonic Test 1:



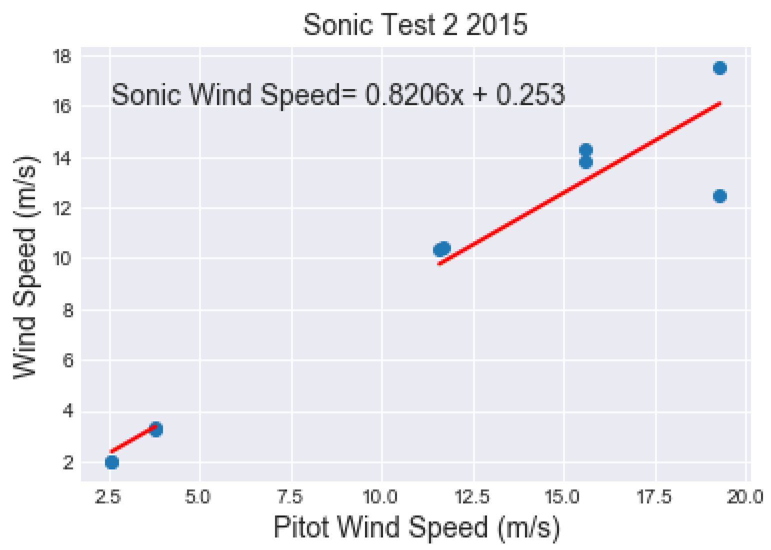
Transfer equation for test 1:

$$Sonic\ Wind\ Speed = 0.9173 * Pitot\ Wind\ Speed + 0.3613$$

Calibration equation for test 1:

$$Pitot\ Wind\ Speed = -2.5388 + 2.7677 * Sonic\ Wind\ Speed$$

c. Sonic Test 2:



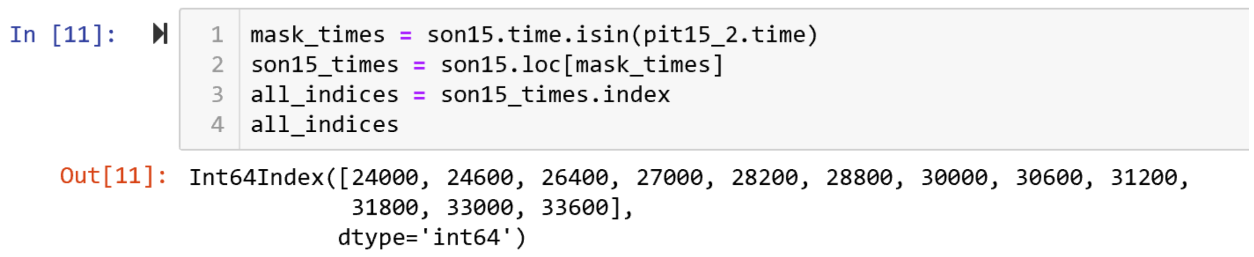
Transfer equation for test 2:

$$Sonic\ Wind\ Speed = 0.8206 * Pitot\ Wind\ Speed + 0.253$$

Calibration equation for test 2:

$$Pitot\ Wind\ Speed\ = \ -0.069\ +\ 0.2614\ *\ Kestrel\ Wind\ Speed$$

Averaging all the data: For each part (a,b, and c) the data was averaged in four steps.
(1) All instantaneous points with the same time as pitot data were selected.
(2) The indices of where these coinciding points occur in the Sonic and Kestrel datasets were stored.
(3) 10 points were picked before and after the indices found in (2)
(4) The 21 datapoints (for each index) were averaged into one datapoint for that time. This gave an average Sonic and Kestrel value of 21 datapoints around the timestamps of the Pitot dataset.



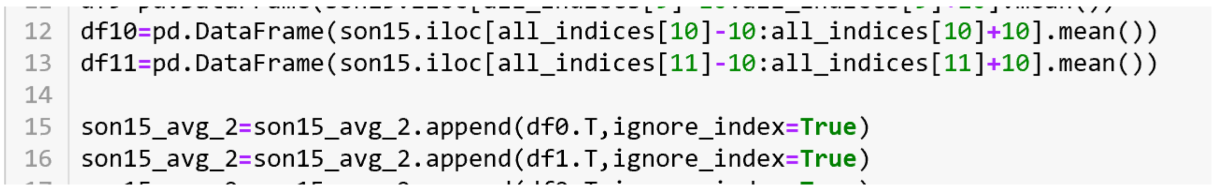
```
In [11]:  1  mask_times = son15.time.isin(pit15_2.time)
          2  son15_times = son15.loc[mask_times]
          3  all_indices = son15_times.index
          4  all_indices

Out[11]: Int64Index([24000, 24600, 26400, 27000, 28200, 28800, 30000, 30600, 31200,
                     31800, 33000, 33600],
                    dtype='int64')
```

Sample code for steps (1) and (2) with the sonic data.

Line 1 → mask_times creates a Boolean mask for coinciding timestamps between the Pitot and Sonic data.

Line 2 → Applies the mask to sonic data to filter only those rows with matching timestamps

Line 3 → Finds the indies of the rows with the matching timestamps as they occur in the original dataset



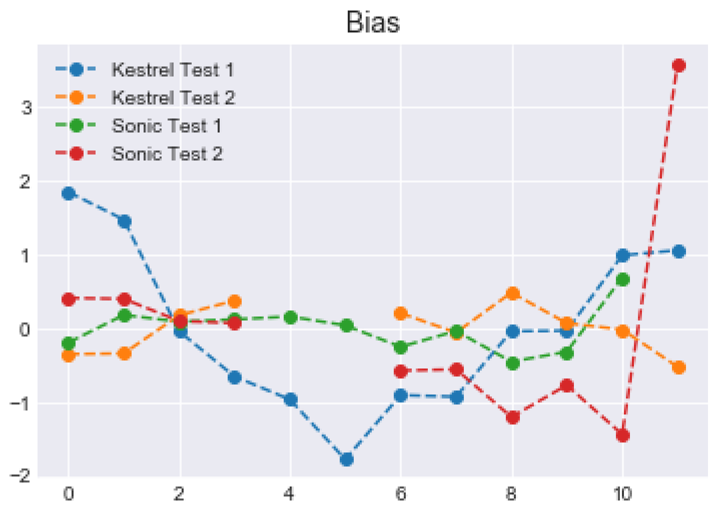
```
df10=pd.DataFrame(son15.iloc[all_indices[10]-10:all_indices[10]+10].mean())
df11=pd.DataFrame(son15.iloc[all_indices[11]-10:all_indices[11]+10].mean())

son15_avg_2=son15_avg_2.append(df0.T,ignore_index=True)
son15_avg_2=son15_avg_2.append(df1.T,ignore_index=True)
```

Sample code for steps (3) and (4) with the sonic data (Test 2)

Lines 12 and 13 → Select 10 rows before and after the index selected in (2). Find an average of the 21 rows and convert it into a row in a pandas dataframe.

Lines 15 and 16→ Append multiple such rows to a final dataframe that stores the averaged values at matching timestamps as the Pitot dataset.

3. Bias calculations

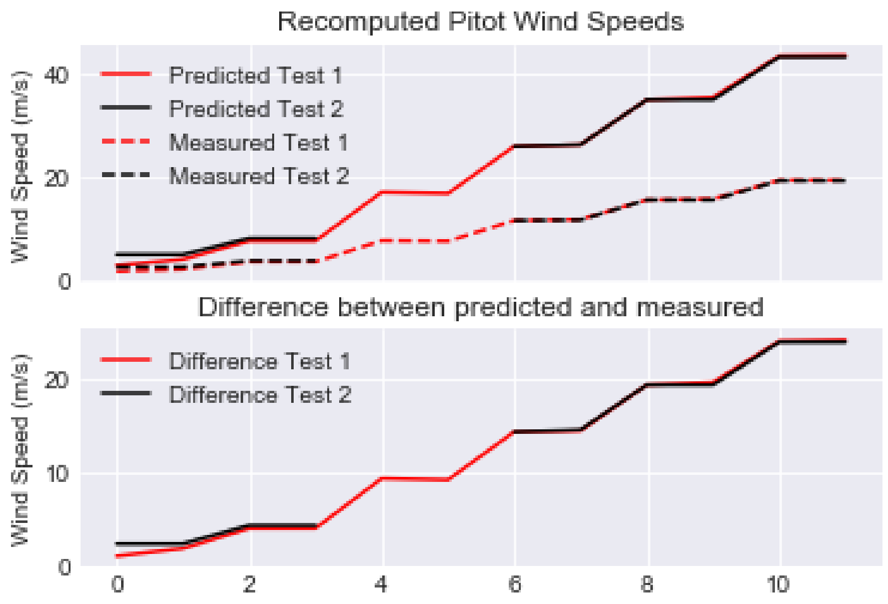


   a. Kestrel Average Bias
      i. Test 1: 0.8931 ~0.89
      ii. Test 2: 0.2597 ~0.26
   b. Sonic Average Bias Test 1
      0.2298 ~ 0.23
   c. Sonic Average Bias Test 2
      0.90998 ~ 0.91

4. It is hard to measure the dynamic performance of the Sonic because it has a very fast response and the sampling interval is likely bigger than the actual response. The Kestrel however has a propeller that has a dynamic response which is relatively easier to measure.
5. Assuming the instruments were placed inside the tunnel in 2015, this would make the experiment a full calibration since it had a reference instrument (pitot), a wind tunnel set up to restrict air flow and ensure steady flow that is uniform across the tunnel.

6. Considering an amplification factor of 5 on the barometer measurements, the following equation was used to calculate the windspeed:

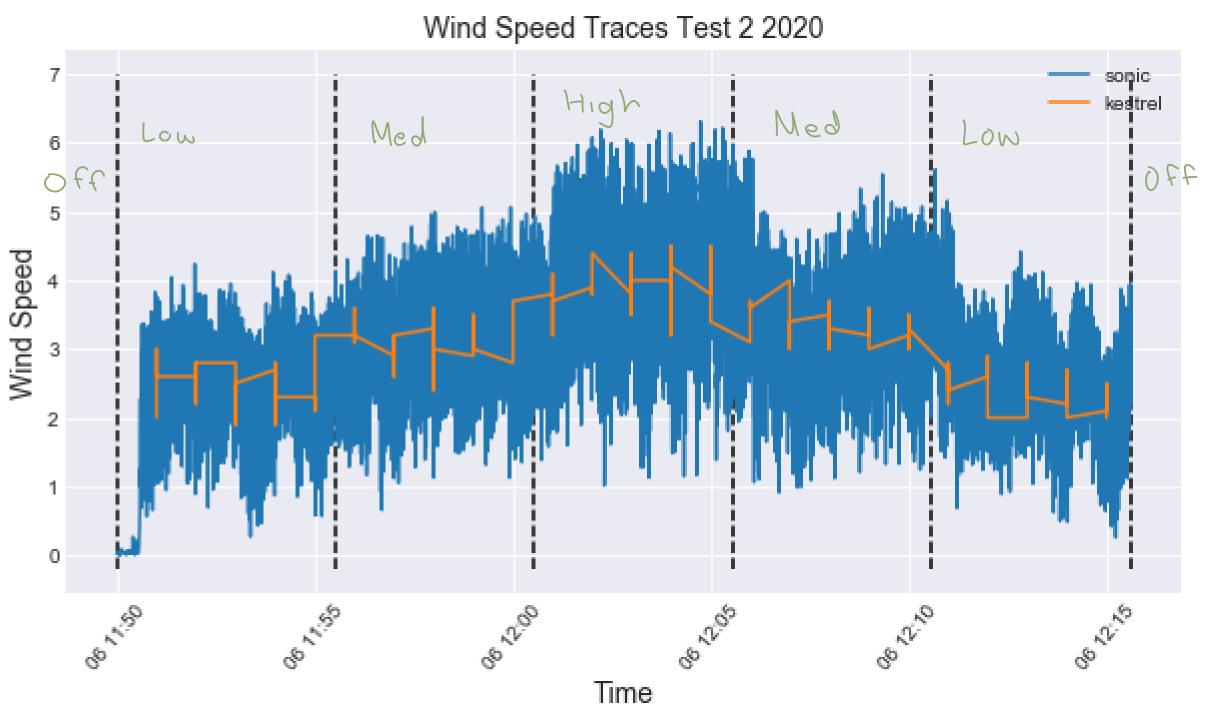
$$Wind\ Speed = \sqrt{\frac{2 \cdot (\Delta p - 5)}{\rho}}$$



These results differ from those found in the spreadsheet because air density would also depend on factors such as humidity and temperature which are not corrected for in the predicted data.

Part 2:

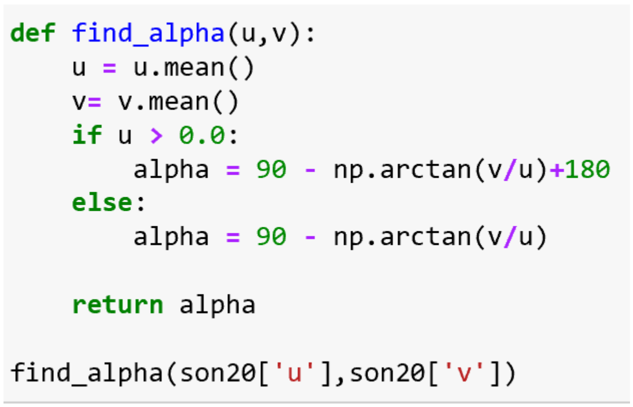
1. Wind Traces for Kestrel and Sonic (Only Test 2 plotted because Test 1 did not have good data)



2. The code used to convert Gill measurements to Meteorological ones is as follows:

```
def convert_gill_to_met(gu,gv,gw):
    """
    met u = gill -v
    met v = gill u
    met w = gill w

    """
    w = [x for x in gw]
    u = [-x for x in gv]
    v = [x for x in gu]

    return [u,v,w]
```

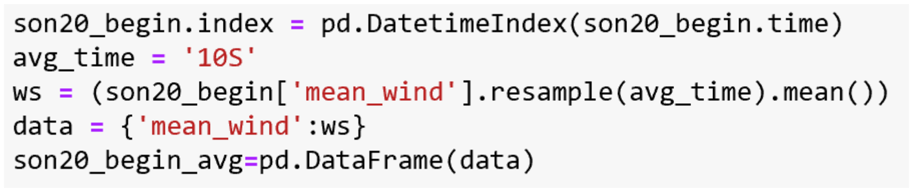
The code used to calculate the wind direction is as follows:



```python
def find_alpha(u,v):
    u = u.mean()
    v= v.mean()
    if u > 0.0:
        alpha = 90 - np.arctan(v/u)+180
    else:
        alpha = 90 - np.arctan(v/u)

    return alpha

find_alpha(son20['u'],son20['v'])
```

The final wind direction using these was 89.80°

3. To find the time constant I tried to calculate the time takes to reach 63% of the change in wind speed. I did this in the following steps:
   (1) I averaged the Sonic data for every 10 seconds, and Kestrel every minute between the times the fan went from Off to Low. This is because the Kestrel data was provided already averaged at 1 min intervals, and so I left that as is. The Sonic however was at 10Hz meaning 10 points per second, and 600 per minute which would be too many. So I chose 10 seconds to account for 100 points to average.



```python
son20_begin.index = pd.DatetimeIndex(son20_begin.time)
avg_time = '10S'
ws = (son20_begin['mean_wind'].resample(avg_time).mean())
data = {'mean_wind':ws}
son20_begin_avg=pd.DataFrame(data)
```

This code snippet shows how the sonic data was averaged for every 10s.

(2) Function was written to find the area under the curve at each timestep (10s for Sonic, 1minute for Kestrel).

```
Folding = collections.namedtuple('Folding', 'index area_covered difference')

def e_folding(df_wind):
    area_org = trapz(df_wind, dx=1000000)
    fol = area_org*0.63
    acc=[]
    for x in np.arange(len(df_wind)):
        area_x= trapz(df_wind[0:x], dx=1000000)
        area_cov=round((area_x/area_org)*100)
        diff = round(abs(fol - area_x))
        folding = Folding(index= x, area_covered=area_cov, difference = diff)
        acc.append(folding)

    return acc
```

The function written to find the area under the curve. A tuple was created that stores the index, area under the curve and difference for 63% for each timestep.

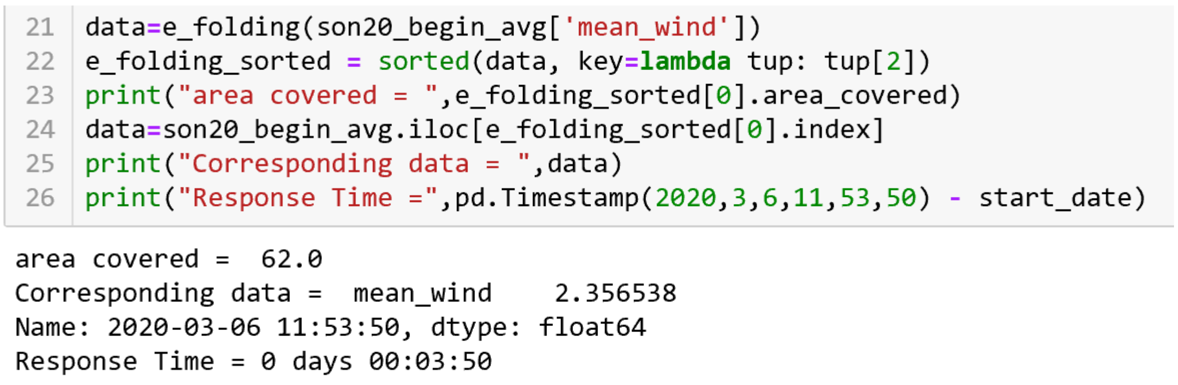
(3) The areas under the curve were sorted from the biggest to smallest difference from 63% of the total area. That is, the times at which the area under the curve was closest to 63% was shown first in the list.

```
data=e_folding(kes20_begin_avg['ws'])
e_folding_sorted = sorted(data, key=lambda tup: tup[2])
e_folding_sorted
```

```
[Folding(index=3, area_covered=51.0, difference=1142542.0),
 Folding(index=4, area_covered=76.0, difference=1240792.0),
 Folding(index=2, area_covered=26.0, difference=3638375.0),
 Folding(index=0, area_covered=0.0, difference=6213375.0),
 Folding(index=1, area_covered=0.0, difference=6213375.0)]
```

The output list shows the areas sorted from lowest difference to largest difference. Here the index refers to the row index (which is analogous to the timestep), the area_covered is the area under the curve covered by this timestep, and difference is the difference from 63%. Since 63-51 is the smallest absolute difference, index 3 shows up first in the list.

(4) The times were retrieved using the index, and the start time was subtracted from it to find the response time.



```
21 data=e_folding(son20_begin_avg['mean_wind'])
22 e_folding_sorted = sorted(data, key=lambda tup: tup[2])
23 print("area covered = ",e_folding_sorted[0].area_covered)
24 data=son20_begin_avg.iloc[e_folding_sorted[0].index]
25 print("Corresponding data = ",data)
26 print("Response Time =",pd.Timestamp(2020,3,6,11,53,50) - start_date)
```

```
area covered =  62.0
Corresponding data =  mean_wind    2.356538
Name: 2020-03-06 11:53:50, dtype: float64
Response Time = 0 days 00:03:50
```

Here line 23 prints out the area covered by a timestep. In this case it is 62%. The index of this output is used to find the corresponding time. In the case of Sonic data, it is at 11:53:50. The time difference is calculated and to be found at 3 minutes and 50 seconds for the response time.

(5) Steps 1 to 5 were repeated for the fan setting from Low to Off.

Using these steps, the following response times were found:

Off to Low:

- Kestrel: The Kestrel data did not have a perfect time that was close to 63% of the data. This could be because it was averaged at a much larger interval. At 11:53 51% of the change was accounted for and at 11:54 76%. So I approximated 63% to be at a halfway time at 11:54:30. The response time is thus 4 mins 30 secs
- Sonic: 3 mins 50 sec at about 62%

Low to Off:

- Kestrel: Similar to the Kestrel data from Off to Low, the Kestrel at 12:14 accounted for 52% of the change, and at 12:15 76%. So I approximated the time at 12:14:30 and the response time would be 3 mins 55 secs
- Sonic: 02 mins 55 sec at about 62%

4. The kestrel contains some level of hysteresis due to which even after stopping the air flow the propeller will keep spinning and therefore recording before it comes to the stop. The same is true for the beginning when the propeller initially at rest needs to start recording. To overcome it's initial state at rest has some time lag due to which the time constants differ. As for the Sonic the time constant differ due to eddies and turbulence generated by the fan when it first starts. The response time in the beginning is greater (as also observed in a wind vane) at the start because the initial wind gust is far more turbulent than when the fan is turned off.

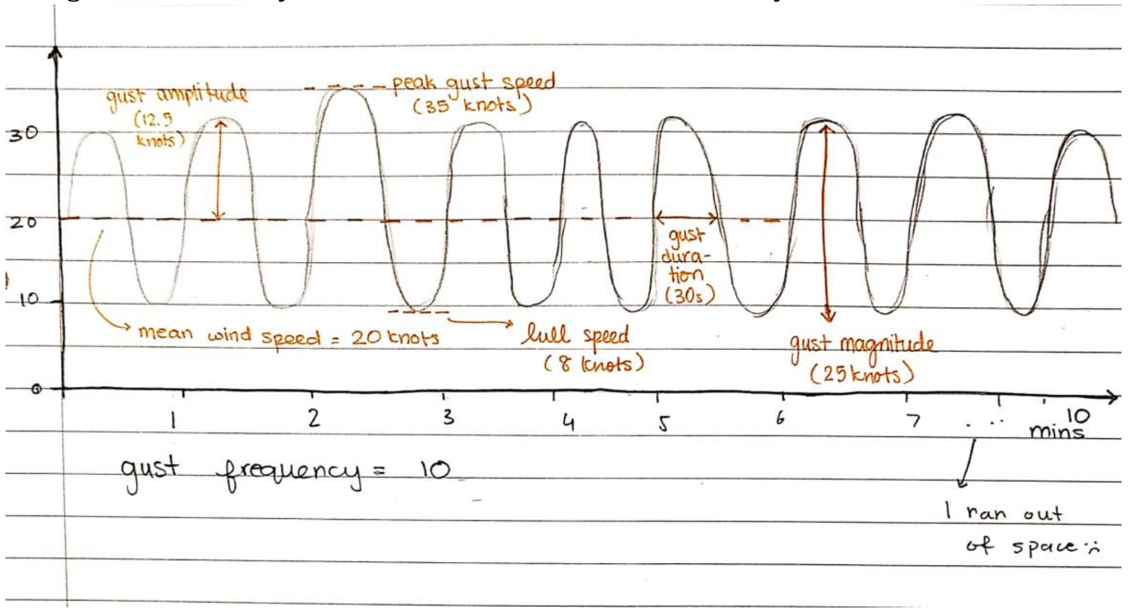
5. I think the Sonic is more accurate because it depends on the speed of sound and calculates wind from all the 3 directions.
6. I think the Kestrel has higher resolution since it can be used to measure wind further out. Additionally the sonic anemometer has a space resolution limitation and measures wind speed averaged over the path length that is much shorter than the equivalent space resolution limit of a cup or propeller anemometer.
7. This set up does not have the pitot to measure pressure or air density and does not have a wind tunnel. The wind tunnel stabilizes flow to a more steady and uniform flow regime. This reduced ambient air flow and turbulence. Because these properties could not be eliminated, a calibration was not done in 2020.
8. The non-zero data values can be present when the fan is off due to eddies and turbulence in the ambient air. There is airflow in the room and when it hits surface it creates small eddies and/or turbulence which could cause the non-zero values.
9. Rearranging equation 8.9 from Harrison, the sonic temperature Ts can be defined as:

$$T_s = \frac{C_s^2 M_r}{\gamma R^*} \approx \frac{C_s^2}{403}$$

where R^* is the universal gas constant,
M_r is the relative molecular mass of air,
and γ is the the ratio of the specific heat capacities
of air at constant pressure and volume

Further Questions:

1. The distance from the anemometer to an obstruction should be at least 20 times the height of the obstruction. For a 9m tall tree, the anemometer needs to be placed at least 180m away from the tree. The anemometer should be placed 10m above the ground. If these conditions are not satisfied the sensor can still be set up at a site with minimal interference. It is very difficult to find measuring sites that conform to these standards; most sites are compromised to some extent, in some directions. It is a good practice to take photographs at each measuring site of the surrounding terrain in all directions to document the fetch. Ideally, this should be done in winter and in summer to show the seasonal effect, and the site characterization should be repeated whenever there are major changes in the vicinity of the station and at intervals of five years or less.



2.
3. Yes because for a gust with a vertical component the cup anemometer has a poor cosine response and the shielding by the upwind cup is reduced. The propeller has better response so may not always overestimate.
4. Yes a cup anemometer or propeller anemometer can overestimate the gust magnitude because they respond quickly to speed increases slowly to speed decreases. As a result the
5. The cup anemometer has a larger drag coefficient that slows down the rotation rate.
6. Let's set V_T to be 1.

$\Delta V = 1 - 0.28 = 0.72$

$63\%\ of\ \Delta V = (0.63)(0.72) = 0.4536$

$Time\ taken\ for\ 63\%\ of\ the\ change = \tau$

$0.28V_T + 63\%\ of\ \Delta V = 0.28 + 0.4536 = 0.7336$

$Since\ this\ is\ close\ to\ 0.74\ we\ can\ see\ that\ it\ takes\ 2.5\ s\ to\ go\ from\ 0.28\ to\ 0.74$

$\therefore\ \tau = 2.5s$

Response length = $\tau \times V = 2.5 \times 6 = 15m$