Intro

In this file, we will discuss the pre-processing used on the Sirris dataset. This dataset consists of four files: data_description.csv, la-haute-borne-data-2013-2016, la-haute-borne-data-2017-2020 and static-information.csv. The data runs from 2013-01-01 to 2018-01-12. The dataset is open source and can be downloaded at: https://opendata-renewables.engie.com/explore/index

Step 1: reading data files and selecting columns

First, we will read in both data files and join 2013-2016 together with 2017-2020. At 2017-05-15, a second wind vane was installed. Therefore, from this point on, Va will be missing and Va1 and Va2 will be present. To deal with this, the average of Va1 and Va2 is taken and inserted as Va. This allows us to keep the same set of variables throughout the analysis.

The first two columns are 'turbine' and 'timestamp'. These are extracted and stored into a list (non-numerical variables). The variables that are empty are dropped (Va1, Va2, Pas, Wa_c, Na_c) along with variables Na_c_avg and Wa_c_avg. These two variables are missing between 2017-01-01 and 2017-02-01 and between 2017-04-01 and 2018-01-12. With nine months of data missing, the reliability of these variables becomes questionable. Therefore, they are dropped. The remaining data is now entirely numerical and has 116 variables. We keep a list with the names of these variables.

Step 2: chronological ordering

The list of timestamps is sorted and the indexes are returned. We then go through these indexes and check which turbine they belong to (using the turbine list). They are then stored into an array one by one. This results in four arrays: WT1, WT2, WT3 and WT4. They have a size of 264 528 by 116. WT4 has a size of 265 **384** by 116. Closer inspection shows that the day 2017-06-10 is written out twice for the first three wind turbines but not for the last one. This is dealt with in the next step.

Step 3: imputing of missing timestamps and double timestamps

Some timestamps will be missing from the data. These can't be left out, because we might want to join up our data with a different time series. Therefore, we will interpolate these values. The missing values for all four wind turbines are: two days (2013-03-04 and 2013-05-14) and one timestamp (2013-01-27T00:00). These periods are interpolated linearly.

There is also one day present twice in three of the four turbines. As all datasets end on the exact same day, we will assume that the turbines accidentally wrote

out twice within a ten minute period instead of once. This seems very likely, especially since from around 8:00 onwards, the turbines write out one set of real data and one empty row with the same timestamps. To fix this, we toss out half of the points leaving a normal 144 points per day. If one row is empty, we select this row to delete. If both rows are filled, we select the first one (random).

Apart from this, daylight savings time (DST) is wrongly applied. Normally, one hour should be missing in March when entering DST and one hour should be present twice when leaving DST in October. Instead, we find that in March, there is first an hour missing, immediately followed by an hour doubled. In October, no hour is doubled. Therefore, the data effectively never entered DST and all timestamps from March to October are running an hour ahead. However, there is no need to adjust the timestamps, as we will not use them further on. Only their order is important (and the fact that the time between each timestamp is constant).

These operations leave us with four arrays of size 264 673 by 116. This corresponds to 5 full years (2013, 2014, 2015, 2016 and 2017 of which 2016 is a leap year) and 12 days in 2018 (+1 extra point which is 2018-01-13T00:00) or a total of 1 838 days.

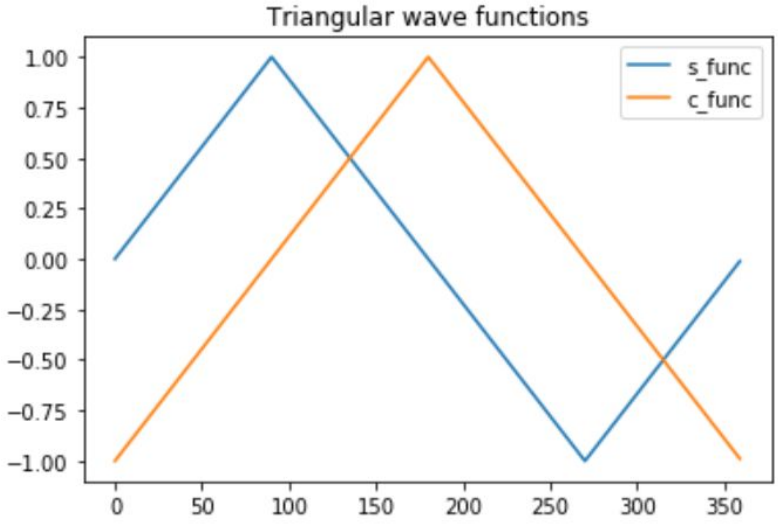
Step 4: imputing empty variables

Some values will be missing from the data. An empty variable is not the same as a missing timestamp. If the timestamp is missing, there is a row gone that should have been there. If a value is empty, it means the row is there, but the actual values are not filled in. These variables are identified and interpolated linearly between known datapoints (just like before). If datapoint 5 is missing, the interpolated value will be the average of datapoint 4 and 6. Because the data is chronological, this makes sense.

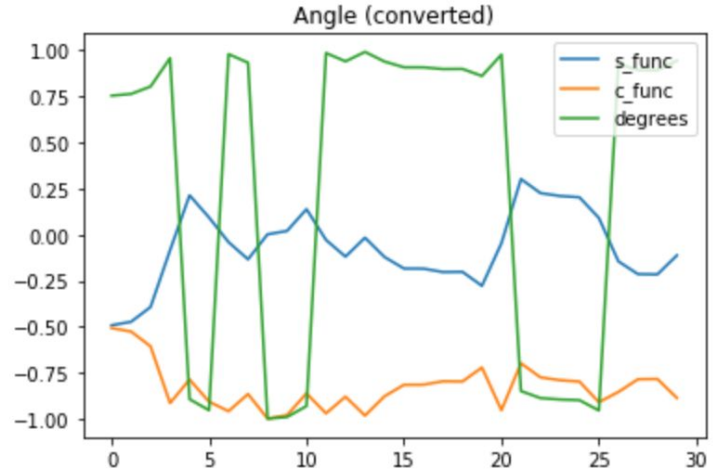
Step 5: Angle conversions

The angles used in the dataset are not usable in their current form. This is because angles are circular variables between 0 and 360. The computer will assume that the difference between 355 and 5 degrees is 350, while in reality it is 10 degrees (shortest angle). Computers will only use one direction for angles as they will assume this is a linear variable. To handle this, all angles are converted with triangular functions. This principle is similar to taking a sine and a cosine, which uniquely describe every possible angle. However, sines and cosines do not have a fixed slope. Therefore, the same difference can create larger differences in sine or cosine depending on the current angle. To fix this, we take a form with a fixed slope (a triangular function). This is illustrated in the figure below (where the horizontal axis shows degrees and the vertical axis shows the value of s and c). As is clear, any change in x (angle in degrees) will

create the same relative change in either s or c, regardless of the current angle. Even when going above 360 and back to zero, this principle applies.



The figure below shows an example of this transformation. The green line shows the original angle with 180 subtracted and divided by 180. This creates values between -1 and 1. When the line 'jumps' between 1 and -1, it means 360 degrees was reached and the angle went back to 0. As is clear from the decomposition, these 'jumps' are all gone. The higher the real change in angle, the more the s and c components change.



All angles are converted to lie within a range of 0-360 (before, Ba was in a range of -120 to 240 and Va in a range of -180 to 180). Then, the S and C functions are used to convert them into our new angle variables. The std-components are tossed out as they are completely useless (the std of 0 and 259 degrees is very high, even though the actual difference in angle is very small).
With these transformations, we are left with 124 variables (4 were deleted, 12 were doubled, a net gain of 8 variables).

Step 6: normalisation

As a final step, all variables are normalised by subtracting the mean and dividing by the std. In reality, we divide by std+1e-5 to make sure we do not divide by zero.

Step 7: write out data

These four arrays are now written out to four different files (WT1_full,...) plus an extra file with the names of the variables in order (124 variables).

There are 264673 rows which correspond to the timesteps:
2013: 1-52560
2014: 52561-105120
2015: 105121-157680
2016 (leap year): 157681-210384
2017: 210385-262944
2018 (ends on Jan 12): 262945-264673

The turbines are divided as follows:
WT1: R80736
WT2: R80721
WT3: R80711
WT4: R80790