Data Selection Process

Selecting Days with Rain

Rain data from the nearest weather station available on the Met Office WOW site is examined to find days when it rained at the radar site. Any day with more than 1.0mm of rain is chosen for analysis.

The data don't need to be very accurate, they are just used as an indication of when it was raining overhead on any given day. I use a python notebook to read the wow files and create a csv file with time and rainfall accumulation over each day.

Processing the vertical radar scans

The SETTINGS.py file is used to set up the paths for the input data and output data, as well as the LOTUS log files.

The function that does the following processing is *process_zdr_scans* in *utilities/calib_functions.py*

The top level scripts are in *calc_calib*:
process_vert_scans_day.py and *process_vert_scans_time_series.py*
For each day with rain over the radar, the vertical scans from the radar are examined.

1. The first step is to calculate a simple average profile of Z, RhoHV, ZDR and V over the 360 degrees. I convert Z to linear units before I average, although it doesn't make a huge amount of difference.
2. I then look for data that meets the following criteria:
   rhv_prof > 0.99, uzh_prof > 0, uzh_prof < 30 and rv_prof <- 2.
3. The starting point is the lowest range gate that meets these criteria. If the starting value is above 1km, then we ignore this profile and continue to the next time step.
4. The next section of code is new since the RAINE project. It uses some of the methodology of Daniel Sanchez-Rivas' work.
   https://doi.org/10.5194/amt-14-2873-2021
   It calculates normalized profiles of RhoHV, UZH and the gradient of the vertical velocity, then combines these profiles together.
5. The scipy "find peaks" function is used to find peaks/valleys in the combined profile in order to find the base of the melting layer, and use this as an upper boundary for the presence of rain.
6. Data from the profiles are extracted below this boundary where they meet the criteria defined above.
7. The section of valid data has to exceed 250m in order to be used.
8. Median values of the ZDR profile are then calculated.
9. The ZDR and melting layer base (MLB) data for each day is saved in a file called day_ml_zdr.csv. This height is used in the Z calibration.
10. A separate function processes the daily files to calculate hourly values. There has to be at least 3 profiles in each hour to calculate an average hourly value. This is to

exclude any values being derived from bad profiles/outliers that got through the above processing/thresholding. However, I tested using all valid profiles or hourly values and there was no difference in the results.

process_hourly_zdr.py is used which calls the function *calc_hourly_ML* in *calib_functions.py*

ZDR bias from horizontal scans

We can also analyse ZDR from the low elevation PPI scans.
process_horz_zdr_day.py calls the function *horiz_zdr* from *calib_functions.py*

The script looks at low level scans, uses the melting layer height that was calculated from the analysis of the vertical scans, and only looks below this level, and also thresholds on the following:

rhohv>0.99, sqi>0.3, phidp>0, phidp<6, uzh>15, uzh<=18

CSV files are created for each day

The script *plot_horz_zdr_bias_time_series.py* is used to plot the results.

SCRIPT and OUTPUT SUMMARY

VERTICAL SCANS

STEP 1

- RUN process_vert_scans_time_series.py
  - CALLS process_vert_scans_day.py
    - CALLS process_zdr_scans in calib_functions.py in /gws/nopw/j04/ncas_obs/amf/software/ncas-mobile-x-band-radar-1/calc_calib

Produces the following files for each good rain profile:
day_ml_zdr.csv
vert_profs_YYYYMMDD_HHmm.png
vert_profs_YYYYMMDD_HHmm.txt

STEP 2

- RUN process_hourly_zdr.py
  - CALLS calc_hourly_ML in calib_functions.py

Produces hourly_ml_zdr.csv containing hourly values of ML and ZDR bias.

STEP 3
Run plot_zdr_bias_time_series.py to plot the results.

HORIZONTAL SCANS

STEP 1

- RUN process_horz_zdr_time_series.py
  - Calls process_horz_zdr_day.py
  - CALLS the function *horiz_zdr* from *calib_functions.py*

OUTPUT:
YYYYMMDD_horz_zdr.csv

STEP 2

- Run plot_horz_zdr_bias_time_series.py to plot the results