CLABAUTAIR User Manual

Version 1.0, October 2006

correspond to:
Ronald Scheirer
Sveriges meteorologiska och hydrologiska institut (SMHI)
Forskning och Utveckling
Folkborgsvägen 1
601 76 Norrköping, Sverige
email: ronald.scheirer@smhi.se

Contents

1 Introduction

One of the remaining challenges in three-dimensional radiative transfer calculation is the generation of realistic cloud fields as input for sophisticated radiative transfer models. The algorithm CLABAUTAIR (**cl**oud liquid w**a**ter content and effective radius retrieval **b**y an **aut**omated use of **air**craft measurements) has been invented at the Institut für Physik der Atmosphäre, Deutsches Zentrum für Luft- und Raumfahrt, Oberpfaffenhofen, Germany, but development is still going on. This tool is designed to retrieve the structure of clouds from airborne measurements of microphysical parameters. Data from individual flight legs are scanned for characteristic patterns, and the autocorrelation functions for several directions are used to extrapolate the observations along the flight path to a full three-dimensional distribution of the cloud field. Thereby, the mean measured profiles of microphysical parameters are imposed to the cloud field by mapping the measured probability density functions onto the model layers.

1.1 Disclaimer

These programs are distributed as **freeware**. Except when otherwise stated in writing the software packet is provided **as is** without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the programs is with you. In no event unless agreed to in writing will the author or any other party who may modify and/or redistribute the programs be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the programs.

1.2 First steps

Under `http://www.smhi.se/cloud_generator/` you will find updates for both, manual and programs. To unpack the file `CLABAUTAIR.tar.gz` just copy this file to the directory of your choice in a UNIX/LINUX environment and type first `gunzip CLABAUTAIR.tar.gz` and then `tar -xvf CLABAUTAIR.tar`. This will create a directory `CLABAUTAIR` containing all the necessary files.

2 Architecture

This section give an overview about the structure of program cooperation and program flow. Central settings - the user wishes - are to be placed in one central file (`communicat.dat`) that all programs will read.

2.1 Input files

Most of the data files in use are both – input and output files, because they are produced or modified by one program and read afterwards by the next. The files discussed here are files that must exist before of the first program run.

2.1.1 `communicat.dat`

This is the main file where all user settings are stored in. The coding is position based, i.e. there are no keywords needed, the meaning of variables is defined by their position in this file. Since the requirements are concise this shouldn't be a problem. Settings are to be written in lines one to nine. This are:

1. obsfile
2. xres yres angres
3. nz corrflag
4. *vertical borders*
5. lwcres reffres
6. gridflag
7. indflag
8. vel dir slot time
9. preflag

2.1.1.1 Line 1: obsfile This is simply a part of the name of a file containing the aircraft measurements. Just enter a string 'name' there. CLABAUTAIR expects a file 'name'`.red` to exist in the same directory (don't ask for the reasons).

2.1.1.2 Line 2: xres yres angres

This is the place to provide informations on some resolutions you selected. Enter two real-values for xres and yres to specify the required horizontal resolution (xres is assigned to the East-West direction and yres to the North-South direction), both in km. The angular resolution angres specifies the angle-bin-width used to scan the measurements for straight flight legs. If it is too small, you may get short or even too few lines. On the other hand to large values result in blurred legs. We found good results with angular resolutions around 2.5 degree. By the way, an integer times angres should (!!!) result in a full circle.

2.1.1.3 Line 3: nz corrflag

The vertical resolution results from settings of nz. Place an integer there to give the number of layers. The number of vertical borders (read from line 4) is nz+1 then. If you just have equally sized vertical layers place a negative value here (find the reasons in the next paragraph). The corrflag is an integer which controls the calculations of autocorrelation functions. If this is set to 1, the calculation is switched to a periodic mode (please read Section 2.2.3 for some details).

2.1.1.4 Line 4: *vertical borders*

Enter here the nz+1 arbitrarily spaced layer-borders, real values in km. If nz is less than zero type just three real values here: The lowest border, the upper border, and the increment. Again in km.

2.1.1.5 Line 5: lwcres reffres

This is not a critical setting. Just give here the bin-width for the calculation of the probability density functions of LWC (lwcres) and R_{eff} (reffres) in real numbers. Take care, that you not just get one bin filled with measurements and not plenty of bins filled with just one measurement. Use simply the same units that are used in the measurements file.

2.1.1.6 Line 6: gridflag

This is easy. An integer that should be 0 if the horizontal positions in the measurements file are Cartesian coordinates (use distance in meter – separated in North-South and East-West direction – from a certain reference point in the South-Westerly corner) or 1, if they are given in latitude and longitude.

2.1.1.7 Line 7: indflag

Another integer flag. If set to 0, LWC and R_{eff} are calculated independently on the same way (calculating the autocorrelation functions, etc.). This could be time consuming and may lead into trouble if, for example you find two slightly shifted fields for scattered clouds of LWC and R_{eff} due to small differences in their autocorrelation functions or if you use different instruments for deriving LWC and R_{eff}. One answer and a proper one at least for narrow distributions of R_{eff} is to correlate them with the LWC. Choose 1 if you want to do so.

2.1.1.8 Line 8: vel dir slot time Four real settings on the wind and time. The wind speed in $\frac{m}{s}$ is read from vel, the direction in degree from dir. A value of 270. means wind is blowing from the West to the East. The typical time an aircraft needs to pass one flight leg should be given in seconds here in slot. This information is necessary to avoid the assignment of earlier or later measurements (with possible different cloud microphysics that may cause discontinuities in the autocorrelation function) to a certain flightleg.
The time in seconds for the calculation of the cloud-state is specified in the variable time. If less than zero, the time centred between the first and the last measurement is taken. Note: If you use UTC, MESZ, or Lummerland-Time in the measurement file use it also here.

2.1.1.9 Line 9: preflag Set this integer flag to 1 if you want cloud-free layers between cloudy ones to be calculated first. If done so, cloudy layers separated by cloud-free layers do not affect each other. This leads to a cloud field comparable to one following a random-maximum overlap assumption.

2.1.2 'name'`.red`

This file contains an arbitrary number of aircraft measurements in an arbitrary order. Six columns are expected, that are:
lat lon alti lwc reff mtime
The meaning of lat is the latitude in decimal degree, lon is the longitude in decimal degree (if gridflag is set to 0, lat and lon are expected to contain North-South and East-West distances in meter from a certain reference point), alti the altitude in m, lwc the liquid water content in arbitrary units, reff the effective radius in arbitrary units, and mtime the measurement time in seconds. If you want to calculate an other parameter than the LWC, say the volume extinction and you think that this parameter should follow similar rules, feel free to test it. You should replace reff by the same parameter because a correlation may be questionable.

2.2 Programs and their tasks

In general CLABAUTAIR consists of 5 programs written in FORTRAN90. These programs should be started in the order given below (sections 2.2.1 to 2.2.5) to make sure that following programs are provided with all required informations from the proper source. Table 1 gives an overview on the interaction of the programs.

2.2.1 `navigat`

The program navigat fixes the general geometric extensions of the cloud field. If the aircraft measurements are not given on a Cartesian grid they are transformed from LAT/LON-

Table 1: *The information flow, or in other words the interaction between the five programs is given in brief below.*

Program	Purpose	Reads from	Writes to
`navigat`	fixes the general geometric extensions of the cloud field, transform positions from LAT/LON-notation to an orthogonal metric grid	communicat.dat 'name'.red	communicat.dat cloud.log output1.dat output1b.dat
lines	select contiguous measurements, located on a straight line for calculations of autocorrelation functions	communicat.dat output1.dat	communicat.dat cloud.log output2.dat output2b.dat
corel	calculate lag-correlation coefficients along flight legs by shifting a measurement series against a copy of itself	communicat.dat output2.dat	cloud.log output3.1.dat output3.2.dat
fill	shifted, measured anomalies are averaged over the respective boxes and a wheighted interpolation along flight directions over the whole 3D space will be done	communicat.dat output1b.dat output3.1.dat output3.2.dat	cloud.log random.seed output4.1.dat output4.2.dat
rescale	fit the resulting probability density functions of LWCs and R_{eff} to the measurements	communicat.dat output1.dat output2b.dat output4.1.dat output4.2.dat	cloud.log random.seed wcloud3D.dat

notation to an orthogonal metric grid. The most south-westerly coordinate is selected for marking the reference corner. All later given coordinates are relative coordinates. A listing of 'X = 500 m, Y = 700 m' means 700 m north and 500 m to the east of the reference point.
Input: The general settings are read from `communicat.dat`. Aircraft data are taken from 'name'`.red` ('name' is specified in `communicat.dat`).
Output: Some general informations are written to `communicat.dat` and all activities are protocoled to `cloud.log`. In `output1.dat` the measurements in Cartesian coordinates, re-sorted in layers are stored. The same, but shifted by the wind-vector is found in `output1b.dat`.

2.2.2 `lines`

The main purpose of the program `lines` is to select contiguous measurements located on a straight line for calculations of lag-correlation coefficients. To this end, all measurement-coordinates (at each level) are connected among one another. For all data points the found directions are combined to angular bins and measurements outside a preselected time slot are sorted out. The eight flight legs found this way providing the most measurements and are directionally separated by angles of at least 15° are selected for the calculation of autocorrelation coefficient.
Input: The general settings are read from `communicat.dat`. The re-sorted measurement field is read from `output1.dat`.
Output: Some general informations are written to `communicat.dat`. All activities are protocoled to `cloud.log`. The points of measurement, sorted by flight leg and distance from starting point are stored in `output2.dat`. Some cryptic information on the correlation of liquid water content and effective radius can be found in `output2b.dat`.

2.2.3 `corel`

Autocorrelation function (or lag-correlation coefficients) along flight legs are calculated by this program. This is done by shifting a measurement series against a copy of itself and storing the calculated correlation coefficient as a function of the total shifted distance. Since the overlap of both series decreases with increasing lag, the user can switch between a periodic shift or just filling empty locations with zeros. Except for the unlikely case that a flight leg has exactly the length of an integer multiple of a cloud-cloud-gap distance, the autocorrelation function decreases with displacement from its origin. This is not satisfying. A possible solution for the future could be a sledge-like part of the measurement series to be shifted along almost the whole distance.
Input: The general settings are read from `communicat.dat` and the selected flight legs are taken from `output2.dat`.
Output: All relevant activities together with some useful results are protocoled to to the file `cloud.log`. The autocorrelation functions of LWCs are written to file `output3.1.dat` and for R_{eff} to `output3.2.dat`.

2.2.4 `fill`

This is the most time consuming program. First of all the shifted (by the wind-vector) anomalies are averaged over the respective box sizes where available. The starting field initialised this way provides the basis for ongoing calculations. If you have strong winds and a small domain or unfavourable distributed measurements, severe errors may occur if no or deficient data are left inside the selected domain.
Empty boxes are selected by random next. From the centre of the current box filled boxes

are searched along the directions determined by `lines`, section 2.2.2. Then a weighted average over all anomalies found by this means is calculated. The autocorrelation coefficients are fixed as weighting factors here. If the weighting sum fails to reach a certain threshold (initially set to 3) the calculation for this box is postponed. Additionally a success-rate (sr, $sr = \frac{calculated\ boxes}{tries}$) is calculated periodically. In case of small srs over 3 cycles, the threshold is reduced temporarily. On the other hand, if the threshold is reduced and srs are large the threshold is increased again. The program ends if all boxes are assigned with calculated anomalies or the threshold has to be reduced too dramatically to expect reasonable results.
Input: The general settings are read from `communicat.dat` and informations on shifted measurements are read from `output1b.dat`. The autocorrelation functions of LWCs and R_{eff} are read from `output3.1.dat` and `output3.2.dat` respectively. **Output:** All relevant activities together with some useful results are protocoled to to the file `cloud.log`. Since a random generator is used here, the initialising value is stored in `random.seed`. The anomaly field of LWCs is written to `output4.1.dat` and for R_{eff} to `output4.2.dat`.

2.2.5 `rescale`

The polish – or more precise – the roughened finish is performed by this program. The aim is to fit the resulting probability density functions of LWCs and R_{eff} to the measurements. To this end the calculated anomalies are assumed to represent the probability for a local large LWC or R_{eff}. First the cloud free fraction of each layer $(1 - p_{cf}(z))$ is assigned to the fraction of boxes with the smallest values. A random number generator 'throws the dice' for each of the remaining boxes $(p_{cf}(z) \times N_{boxes})$. This random number generator is prepared to transform from an equal distribution of numbers to the measured PDFs. Finally the largest anomalies in boxes are replaced by the largest values produced by the random number generator.
Input: The general settings are read from `communicat.dat` and original measurements (needed for the PDFs) are read from `output1.dat`. Foe the connection of LWCs with R_{eff} the informations stored in `output2b.dat` are required. Of course the anomaly fields are read from `output4.1.dat` and `output4.2.dat`.
Output: All relevant activities together with some useful results are protocoled to to the file `cloud.log`. The initialising value of the random number generator is stored in `random.seed`. The final three-dimensional cloud field is written to `wcloud3D.dat`

2.3 Output files

There are several files produced during the program run. Normally there is no need to keep all of them.

2.3.1 `wcloud3D.dat`

This file follows the requirements for three-dimensional cloud files from the libRadtran packages running with the MYSTIC solver. There is just one known problem, depending on the chosen compiler and its setting. Line-breaks may appear in the line reserved for initialising vertical borders if it is quite long. They have to be removed manually.