Marielle Bjerring Fournier
March 2019

# Cold pool vr tracking

## 1. Requirements

Requires the use of the Iterative Rain Cell Tracking developed by Moseley et al. 2013 and at least python 2.7.14 with the packages numpy and netCDF4.

## 2. Files included

### *Required for the tracking*

**calc_v.py**

Subroutine for calculating the magnitude and direction of the resultant 2D wind field. Includes moving the velocity half a grid point up-grid such that they align with the scalar quantities, which are defined at cell centers (Arakawa-C grid). Results in a netcdf file with the magnitude and direction of the 2D wind field at all vertical layers.

**remaketxtfile.sh**

Bash script for remaking the output file irt_tracks_output.txt to the two files irt_tracks_nohead.txt and irt_tracks_head.txt from the IRT such that it is readable for the python scripts.

**find_tracks.py**

Subroutine for finding the track IDs, time steps and center of masses (COM) for the identified rain events. Includes the disclusion of rain events with shorter lifetime than two time steps, rain events resulting from a splitting event and rain events that end because of a merging event.

**transfer_grid.py**

Subroutine for transferring the grid such that the cold pool COM is in the middle of the domain and transferring the grid back again.

**interpolation.py**

Subroutine for performing bilinear interpolation - used to calculate dvrdr.

**parameters.py**

Adjustable parameters and folder locations have to be set in this file.

### *Main tracking files*

**calc_vr+dvr+maxdvr.py**

Main tracking file to run. Imports from the other files and generates the output netCDF file and the txt file with the grid-locations of the identified edge-points.

**calc_maxdvr_first.py**

Subroutine for determining the location of the cold pool edges at first identification time for the cold pools.

**calc_maxdvr_more.py**

Subroutine for determining the location of the cold pool edges at every time step after the first identification time for the cold pools.

### *Cython files for increase of speed when calculating vr and dvrdr*

**setup.py**

Script to run when the cython files have been updated.

**calc_vr.pyx**

Cython subroutine for calculating the radial velocity.

**calc_dvr.pyx**

Cython subroutine for calculating the derivative of the radial velocity.

## 3. Setting the parameters

a. *Paths, grid size, temporal and horizontal resolution*
   Set the paths for where the input files are located and where the output files should be located. The grid size, temporal and horizontal resolution should match the simulation that is being analyzed.
b. *Maximum amount of cold pools in same time step*
   In order to initialize the cold-pool-array a finite size must be chosen. This is dependent on the simulation that is being analyzed and should be chosen such that the number is not exceeded. For the simulations analyzed when developing the tracking algorithm a maximum of 500 cold pools was sufficient.
c. *Initial size of rain search area*
   To improve the speed, the radial velocity and its derivative should not be calculated for the whole domain unless the cold pool expands that far. Therefore, a cold pool area is defined, which should enclose the cold pool and a small part of

its surroundings. The size is defined in grid points from the cold pool COM such that if e.g. an initial size of 5 is chosen, then the initial area is 10x10 grid points. The initial size should be chosen based on the grid size of the simulation that is being analyzed and the threshold set for the minimum size of the rain object - the cold pool will initially span approximately the same area.

*d. Threshold for continued cold pool tracking*

When the rain event stops precipitating, the tracking of it is accordingly finalized. The cold pool tracking can be continued based on a threshold set on the average radial velocity at the edge. This threshold is set in m/s.

*e. Travelmore thresholds*

The travelmore thresholds increases the area where the radial velocity and its derivative is calculated and the area where the algorithm looks for the edges. In the succeeding time steps following the first identification of a cold pool, the area where the radial velocity and its derivative is calculated is increased with the average distance travelled by the cold pool but in order to ensure that the area is large enough, it can be increased by adding extra grid points with the travelmore_gridgrowth variable.

The travelmore variable provides an uncertainty (in grid points) to the average distance travelled by the cold pool.

The travelmore_latetime variable is included in order to account for the cold pools near the end of the simulation being able to travel farther because there are less cold pools at this point. When this “late time” occurs is also possible to set as a variable.

*f. Slice size and neighbour constraint*

When the edges of the cold pools are identified, the cold pool area is divided into azimuthal intervals of equal size where the derivative of the radial velocity is averaged and then the location of the edge (the radius) is determined by locating the radius having minimum of the derivative. The size variable determines the amount of azimuthal intervals (minus one) such that if e.g. 33 is chosen, there will be 8 slices in each quadrant of the cold pool (32/4 = 8).

The neighbour constraint determines the radial search range for the edge point (the minimum of the derivative of the radial velocity) in a slice. The mid-point of the range is determined by the edge point (radius of minimum in the derivative) in the previous slice (or where the edge was at last time step, if it is the first slice) If e.g. a value of dr=3 is chosen and the edge point of the previous slice was determined to be at r=7, then the algorithm only locates the minimum of the derivative in the range r=[4,10].

# 4. Running the tracking

All the tracking files should be located in the same folder. The tracking is run in multiple steps

1. Run the IRT
2. Run the remaketxtfile bash script
3. Run the calc_v python script
4. Run the calc_vr+dvr+maxdvr python script - this may take several hours

If it is the first time that the tracking is being used, the cython files calc_vr.pyx and calc_dvr.pyx needs to be build. This is done by running the command:
*python setup.py build_ext --inplace*
The time-demanding parts of the tracking are the calculations of vr/dvr and the writing to the netcdf file. If one does not need the netcdf file (the radial velocity field of each cold pool) then this part can be commented out and the code will run faster through. The location (xy-coordinates) of each cold pools edges will still be known from the text file.

# 5. Output files

The output of the tracking is a netcdf file with two variables (the radial velocity for each cold pool and the xy-coordinates of the determined edges) and a text file with the grid point locations of the determined edges.

**Format of xylist.txt**

The text file contains 6 columns:

1. Time step

2. y coordinate of the edge point
3. x coordinate of the edge point
4. ID of the cold pool
5. y coordinate of the COM of the cold pool
6. x coordinate of the COM of the cold pool