

celltrack

A 2D cell tracking algorithm

v0.9.2

K. Lochbihler

November 16, 2020

1 What is celltrack?

celltrack is software that finds continuous cells in 2D fields and tracks them in time. The basic concept is inspired by Moseley et al. (2013). celltrack consists of several parts which are explained in detail in the following sections. If you use celltrack in scientific publications, please cite:

Lochbihler, K., Lenderink, G., and Siebesma, A. P. (2017), The spatial extent of rainfall events and its relation to precipitation scaling, Geophys. Res. Lett., 44, 8629–8636, doi:10.1002/2017GL074857.

1.1 Definitions

Cell

A cell is a continuous area of grid points which exceed a certain threshold. Two grid points are adjacent if their coordinates differ either in one x or one y step. Diagonal adjacency is not allowed. This definition is in accordance with Moseley et al. (2013).

Subcell

A subcell is part of a cell. A subcell can only be part of one cell and is fully covered by it. Multiple subcells can be part of the same cell. In a nutshell, a subcell is a part of a cell that belongs to a local maximum. Thus, if a cell has only one local maximum (which then is the global maximum of the cell), it has only one subcell which covers the whole cell. As soon as a cell has more than one local maximum, it will have more than one subcell. For a detailed description of the algorithm, see section 1.4.

Track

A track is a time series of cells. Its definition is in accordance with Moseley et al. (2013). Two cells are members of the same track if they (partly) overlap. The difference in time steps where the two cells occur is $(\pm)1$.

This means that a track can not have more than one cell at each time step. A track is initiated by one of the following circumstances:

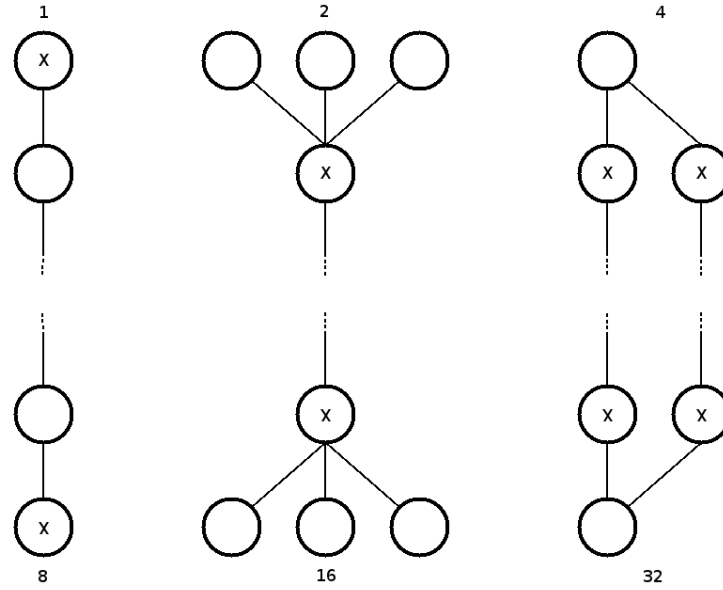


Figure 1: The track taxonomy. There are 6 types of track initiation/termination. In total 9 combinations are possible. Adding up the numbers for initiation and termination a unique track type can be described. A X marks the responsible cell/beginning/end of track.

- a cell has no overlap with cells from the previous time step
- a cell overlaps with more than one cell from the previous time step
- in both cases the cell must not overlap with more than one cell in the next time step

Cells which have overlaps with only one cell from the previous and next time step need to be distinguished further:

- if the backward link cell splits, a new track is splitting from an existing one
- if not, this cell is in the interior of a track.

A cell with the following properties ends a track:

- there is no overlap with cells in the next time step
- merges with other cells in the next time step
- splits into several cells in the next time step

The structure of these 6 different ways of initiating and terminating a track are illustrated in figure 1. Each possibility has a number. Adding up the numbers for track initiation and termination type a unique identifier for the track type can be derived. A track with type 33 for example has a beginning and ending of 1 and 32.

1.2 The clustering algorithm

This part does the cell detection. Therefore celltrack iterates the two spatial dimensions to find continuous areas and assign an unique (integer) ID to them. The following decisions are made for each grid point with a value that exceeds the threshold:

- no adjacent grid point which already has an ID: assign new ID
- one adjacent grid point which already has an ID: use this ID
- two or more adjacent grid points which already have IDs:
 - these grid points have the same ID: use this ID
 - different IDs: use the lowest ID for all grid points

This step is repeated for all time steps. At the end of this step all detected cells have unique IDs. This procedure leads to the fact that a cell A with a higher (lower) ID than cell B occurs at the same time step or later (earlier).

1.3 The linking

In this part all cells are checked for forward and backward links (one time step) with other cells. The results are stored in a array which has the size $n * L_{max}$, where n is the number of detected cells and L_{max} is the maximum number of detected links of a cell in the whole data set. Each row belongs to a particular cell ID. The columns list other cells which are linked to this cell. For an example, see figure 2. The type of link (forward or backward in time) is stored in a second matrix.

The link (and link type) matrix makes it easy to check for overlaps between two cells any time later.

1.4 The subcell detection

A cell represents a continuous area of the key variable above a certain threshold. This definition groups neighboring grid points into the same cell while ignoring the variability of the key variable within the cell. Thus, it is possible that the output statistics (e.g. average or maximum value of a cell) become less representative in the case of large and/or complex cells. To obtain a more detailed picture of such complex cells, celltrack features an algorithm to subdivide cells into separate parts, each of them belonging to one local maximum within the cell. The basic principle of the algorithm follows Senf et al. (2018). The following steps are taken to achieve this:

1. Smooth the input data field with a gaussian blur.
2. Detect local maxima in the field and give them unique IDs.
3. Seed particles at all grid points and let them ascent along the gradient of the smoothed field
4. Assign IDs to the original locations (grid points) of all particles after they have moved to a local maximum.

1.5 Linking subcells with cells

Each subcell is assigned to one cell by checking for overlaps. Due to the definition of a subcell linking multiple subcells to the same cell is possible, but a subcell can not be linked with more than one cell.

1.6 The tracking algorithm

Using the link and link type matrices the tracking part is quite simple. For each row (which corresponds to a specific ID) the algorithm checks whether there are cells that are linked. By checking the link type matrix the number of forward and backwards links in time can be determined. Based on this information and using the link matrix it is known which IDs initiate a track according to the definition given above. If an ID is at the beginning of a track the algorithm changes to the corresponding row in the link matrix and searches for forward links to cells in the next time step. If there is such a connection the algorithm changes to the corresponding row. This step is repeated until the conditions for terminating a track are met. Figure 2 shows a basic example of a link matrix. Based on this one

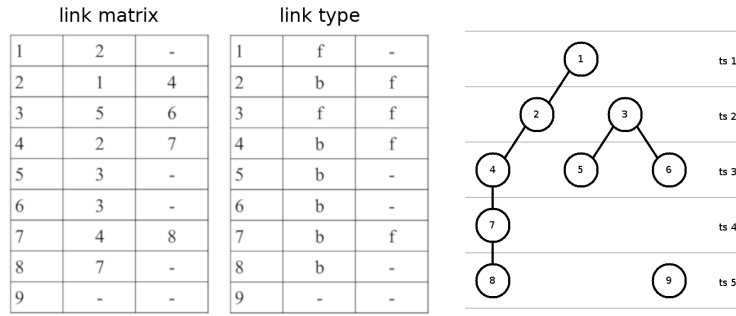


Figure 2: Example link matrix with IDs, the corresponding link types and the corresponding linking tree.

can easily generate the corresponding topology tree which shows all cells and their relationships.

1.7 Advection correction

celltrack features an iterative advection correction algorithm. Again, this follows Moseley et al. (2013). If switched on, celltrack will calculate a velocity field on a coarse grained grid using the differences in coordinates of the weighted centers of mass between linked cells. Only cells with exactly one forward/backward link are used. Depending on the chosen number of iterations this routine writes the velocity fields decomposed into x and y direction as well as the sample sizes into numbered NetCDF files. The file with the highest number i.e. the latest velocity fields are then used in the final linking process. For the advection of cells the displacement is calculated and rounded to grid points.

1.8 Periodic boundary conditions

Sometimes, it is useful to consider the boundaries of a domain as periodic, for instance, if one wants to apply celltrack to model output. For this reason, celltrack features periodic boundary conditions. If switched on, celltrack recognizes if a cell crosses the boundaries during the cell detection and advection correction. This means, that cells of the same track can exit the domain on one side and re-enter on the other side. Without this option a new track is initiated at each boundary crossing. The detection of subcells and the linking to cells supports periodic boundaries.

1.9 Meta tracks and mainstream detection

With increasing length and/or cell sizes it becomes more and more likely that a track ends with a split or merge into/with other track(s). Although, celltrack puts those snippets into the same category as type 9 tracks one could argue that a combination of such related segments forms a complete track as well. For this reason the term *meta track* is introduced. Essentially, a meta track is a group of tracks of different types which are connected directly or through a chain of an arbitrary number of tracks. Figure 3 shows an example for a fairly large meta track. It consists of 52

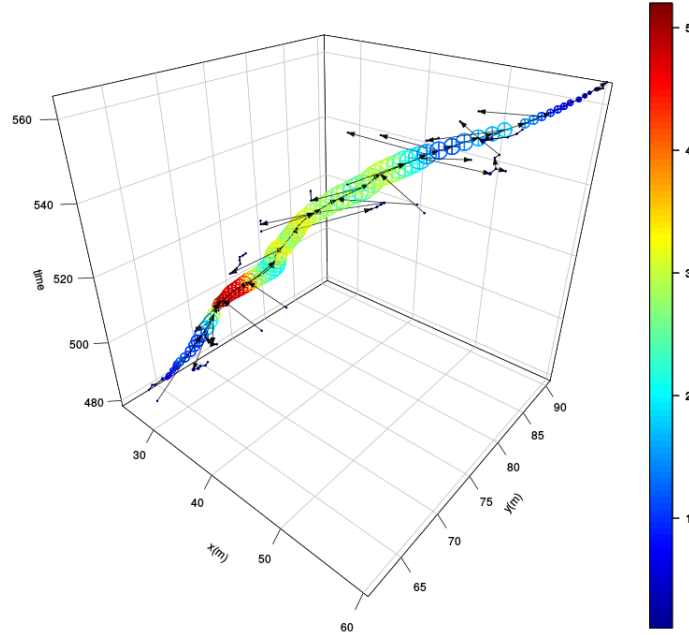


Figure 3: A rather abstract visualization of a meta track. Colors indicate the peak rainfall intensity of a cell (mm/10min). The Size of the symbols is proportional to cell area. The arrows show the direction of a connection between two tracks. The time axis is in minutes.

tracks and has a lifetime of 88 minutes. Apart from a dominant *main-*

stream there are lots of small tributaries and tracks that split away. It is obvious that they have small cell sizes as well as intensities.

To separate the dominant mainstream from dead ends and tributaries, celltrack uses a metaheuristic approach called ant colony optimization (ACO) (Dorigo, 1992). Dorigo and Stützle (2004) give an overview about theory and application of different ACO methods. ACO is inspired by the behavior of foraging ants which are able to find a minimum cost (length) path from a nest to a source of nurture through indirect communication. This is accomplished by modifications of the environment in form of pheromone deposit. To apply this principle to the problem of mainstream detection in meta tracks we have to rise our point of view to a more abstract level. As well as the possible paths from a nest to a source of nurture, a meta track can be seen as a network consisting of nodes and arcs. Using this notation a meta track can be translated into a construction graph where tracks are arcs and the connections between tracks are the nodes.

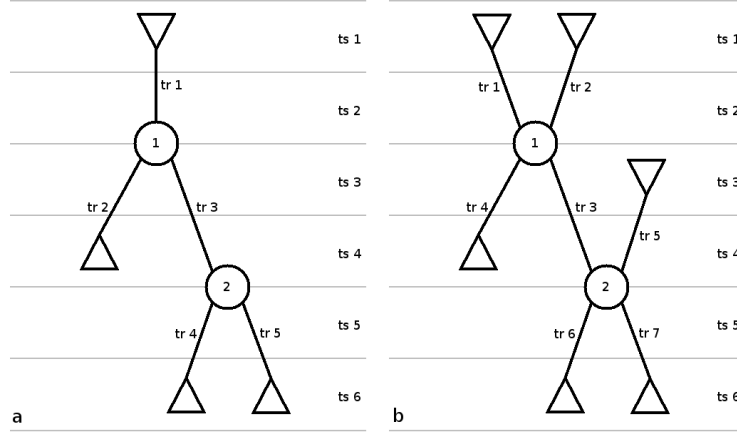


Figure 4: Two simple examples for a construction graph of a meta track. Circles represent nodes; lines show tracks. Triangles denote points of initiation and termination.

A very simple example for a construction graph of a meta track is depicted in figure 4a. It is clearly visible that there are three possible paths following the tracks

1. tr1 - tr2
2. tr1 - tr3 - tr4
3. tr1 - tr3 - tr5.

To rank these candidate solutions by quality, celltrack calculates the average cost of a path by $\bar{C} = \frac{1}{m} \sum_{k=1}^m c_{ij}^k$, where m is the number of nodes in this path and $c_{ij}^k = \frac{1}{2}(|\frac{A_i - A_j}{A_i}| + |\frac{P_i - P_j}{P_i}|)$. A and P are area and peak intensity of the last cell of track i and first cell of track j at the k -th node of this path. Concerning the example in figure 4a it is fairly easy to compute the cost for all paths and detect the one with the lowest cost value as mainstream. But, switching to the example in figure 4b makes

things more complicated. Although only two tracks were added to the construction graph, the number of candidate solutions increases from three to eight. But still, it is feasible to compute all solutions and chose the best. However, the assumption is that the number of possible solutions increases exponentially with the complexity of a meta track. Thus, it is necessary to introduce an optimization algorithm. ACO algorithms can reduce the computation time to find a reasonable solution to combinatorial problems. celltrack incorporates a simplified version of an ACO algorithm (Dorigo and Stützle, 2004) to find a mainstream of a meta track at low computational cost. It consists of the following steps

1. initialize the pheromone values of all arcs by $\tau_{ij} = m/\overline{C^{nn}}$, where $\overline{C^{nn}}$ is the cost value of a nearest-neighbor path of length m (number of nodes) starting at a random initiation point
2. a certain number of artificial ants construct candidate solutions
3. compute cost values for each ants path
4. lower pheromone trails of all arcs using $\Delta \tau_{ij}^k = (1 - \rho)\tau_{ij}^k$
5. increase pheromone trails by $\Delta \tau_{ij}^k = 1/\overline{C}$ for each ants path
6. After repeating steps 2 to 5 certain times, construct a nearest-neighbor path using the inverse pheromone values as a distance measure starting at the initiation point with the highest pheromone value.

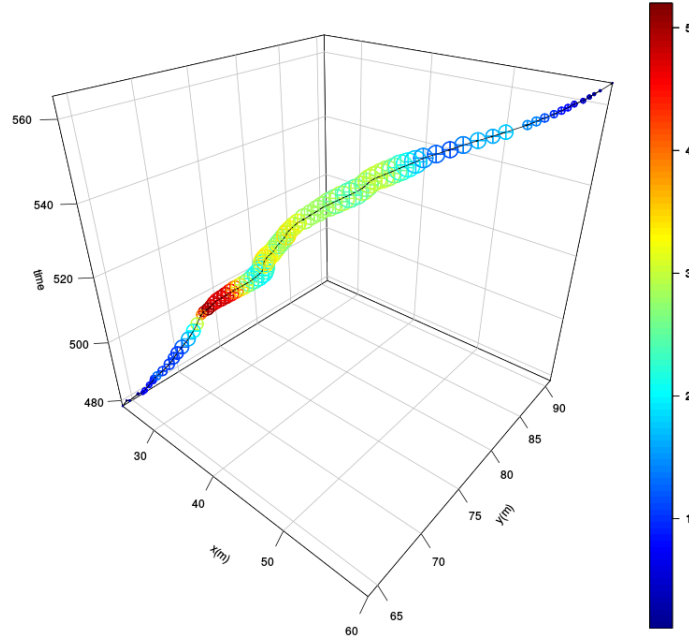


Figure 5: The mainstream of the meta track shown in figure 3; as detected by the ACO algorithm.

Step 2 needs further explanation; this is how an ant constructs a solution:

1. randomly select an initiation/termination point
2. walk to the next node
3. calculate the probabilities for all possible choices at this node with $p_{ij} = \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{l \in N_i} \tau_{il}^\alpha \eta_{il}^\beta}$ with $j \in N_i$. N_i is the group of possible tracks to follow when coming from track i , $\eta_{ij} = 1/c_{ij}$ is the heuristic value. α and β determine the influence of the pheromone and heuristic value. Decide with these probabilities using a random number.
4. repeat steps 2 and 3 until a termination/initiation point is reached.

The reason for randomly switching between a forward and backward in time direction is that only randomly starting at initiation points leads to equal probabilities because this decision is not based on the formula of step 3.

Figure 5 shows the mainstream of the meta track which was previously depicted in figure 3.

1.10 The other stuff

Apart from clustering and tracking there are also routines that calculate cell and track statistics. For this purpose, celltrack uses values from the input file. For detailed descriptions see section 5.

2 Building from source

2.1 Dependencies

The following dependencies are mandatory to compile and run celltrack:

- cdi
- all dependencies of cdi, e.g. netCDF, grib_api

2.2 configure and make

This project uses cmake to configure and create the Makefiles. The following commands should do the job (run within the celltrack directory):

```
mkdir build
cd build
cmake ..
make
```

If you want to install the celltrack binary run

```
make install
```

Remember that this most likely requires root privileges. For a custom path run cmake with

```
cmake .. -DCMAKE_INSTALL_PREFIX=/custom/path
```


2.3 Troubleshooting

2.3.1 cdi includes or library not found

If the cdi include and library are not in your environment variables you can run cmake with the following options to specify the paths explicitly

```
cmake .. -DCDI_INCLUDE=/path/to/include/dir/ \
        -DCDI_LIB=/path/to/lib/dir/libcdi.so
```

or change the values of the according keys in CMakeCache.txt in the build directory.

Another way to tell cmake where to find the cdi include and library is setting the environment variables CDI_INCLUDE_PATH and CDI_LIB_PATH. In bash this can be done running

```
export CDI_INCLUDE_PATH=/path/to/include/dir/
export CDI_LIB_PATH=/path/to/lib/dir/
```

then run

```
cmake ..
make
```

To run celltrack, the path to the cdi library must be in the environment, for example using the LD_LIBRARY_PATH variable on Linux platforms.

2.3.2 Problems with finding gfortran

As this project needs GNU gfortran to compile, the gfortran executable must be detected by cmake. Otherwise cmake reports an error and aborts the configuration process. If this happens and you are sure that gfortran is installed you can force the fortran compiler with

```
cmake -DCMAKE_Fortran_COMPILER=gfortran ..
```

2.3.3 Compile with debugging symbols

In order to compile celltrack with debugging options use:

```
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

3 Usage

A basic example to run celltrack without advection correction:

```
celltrack -i <filename> -var <char> -thres <float>
```

Run celltrack with advection correction:

```
celltrack -i <filename> -var <char> -thres <float> \
        -advcor -nadviter <int> -cx <int> -cy <int> -tstep <int>
```

Example

Run celltrack with a minimum threshold of 0.5. The + suffix for the -thres option can be omitted.

```
celltrack -i surfprec.nc -var prec -thres 0.5+
```

If you want to use a maximum threshold, that means that all values greater than a certain value are ignored, suffix the value with a -. E.g.:

```
celltrack -i surfprec.nc -var prec -thres 0.5-
```

Options

-i <char>	Input file name.
-var <int>	NetCDF/grib variable name.
-lev <int>	NetCDF/grib level ID. Default is 0.
-thres <float>+	A minimum (suffix: +, default and can be omitted) or maximum (suffix: -) threshold for the clustering routine.
-nants <int>	Number of ants/agents for mainstream detection. Default value is the number of nodes of a meta track.
-maxnants <int>	The maximum number of ants/agents for mainstream detection. Setting this value can be useful to avoid long computation times for large meta track. Nevertheless it may decrease the validity of the detected mainstream.
-nruns <int>	Number of iterations for mainstream detection. Default is 300.
-rho <float>	Pheromone evaporation rate. Default is 0.5.
-rseed <int>	If set, the random number generator will use this seed.
-lout	Write the links matrix and link type matrix to file cell_links.txt.
-advcor	If set, celltrack performs an advection correction.
-nadviter <int>	The number of iterations for advection correction. Default is 6.
-maxv <float>	The maximum allowed velocity of cells. This prevents that errors in the advection correction (unrealistically high velocities) are taken into account. Default is 50 distance units/per second.
-nometa	Completely switch off all meta track routines.

-nometamstr	Switch off only mainstream detection of meta tracks.
-perbound	Switch on periodic boundary conditions.
-cx <int>	Definition of the coarse graining of the grid for advection correction. Example: Given that there are 50 grid points in x direction, a value of 25 would lead to 2 grid points for the advection correction grid.
-cy <int>	Same as -cx but for y direction.
-tstep <float>	The time step of the input file in seconds. Mandatory for advection correction!
-tracknc	Output of tracks to netCDF.
-metanc	Output of meta tracks to netCDF.
-minarea <int>	The minimum area of a cluster in grid points. Smaller clusters will be omitted. Default is 0.
-subc	Activate subcell routines.
-sigma <float>	Preprocessing for subcell detection: The standard deviation for the gaussian low-pass filter. Higher values cause a stronger smoothing of the input field. Default is 2.
-trunc <float>	Preprocessing for subcell detection: Truncation of the filter weights. Higher values cause a wider filter span. Default is 4.
-v	Verbose output to stdout. Use with caution! Massively slows down code execution!
-h	Show help.

4 Input

celltrack should be able to read all file formats which are supported by cdi. However, currently only netCDF has been tested. celltrack expects 3D data:

1. x axis
2. y axis
3. time axis

5 Output

5.1 The cells files

5.1.1 cells.nc

This is the first file celltrack will create. It has the same structure as the input file and contains the unique cell IDs. The default file format is netCDF.

5.1.2 cell_stats.txt

This file contains some simple statistics about the detected cells. The columns are:

clID	The cell ID.
tsclID	The time step this cell occurs.
clarea	The area.
grd_clarea	The area in grid points.
clcmassX	The x coordinate of the center of mass.
clcmassY	The y coordinate of the center of mass.
wclcmassX	The x coordinate of the weighted center of mass.
wclcmassY	The y coordinate of the weighted center of mass.
grd_clcmassX	The x coordinate of the center of mass (in grid point index).
grd_clcmassY	The y coordinate of the center of mass (in grid point index).
grd_wclcmassX	The x coordinate of the weighted center of mass (in grid point index).
grd_wclcmassY	The y coordinate of the weighted center of mass (in grid point index).
peakVal	The maximum value.
avVal	The average value.
touchb	TRUE if the cell touches the boundaries.
date	The date for tsclID.
time	The time for tsclID.

5.1.3 cell_advstats_percentiles.txt

This file contains percentiles of the key variable within each cell. The columns are:

clID	The cell ID.
MIN	The minimum value.
0.XX	Percentile of probability 0.XX.
MAX	The maximum value

5.2 The links files

5.2.1 cell_links.txt

This file is rather intended for development and debugging. It contains the link matrix and the link type matrix for each cell.

5.2.2 links_stats.txt

This file contains the number of forward and backward links for each cell. The columns are:

clID	The cell ID.
nbw	The number of backward links.
nfw	The number of forward links.

5.3 The subcells files

These files contain information about each of the detected subcells. The following subsections contain also information about the linking with cells since the linking is not in time in this case; Cells can only contain subcells from the same time step. There is no linking of subcells between time steps.

5.3.1 subcells.nc

This NetCDF file has the same structure as the input file and the cells.nc file. It contains the unique subcell IDs.

5.3.2 subcell_stats.txt

Statistics and characteristics for each subcell, similar to cell_stats.txt. The columns are:

subclID	The cell ID.
subtsclID	The time step this subcell occurs.
subclarea	The area in unit grid points.
subclcmassX	The x coordinate of the center of mass.

subclcmassY The y coordinate of the center of mass.

subwclcmassX The x coordinate of the weighted center of mass.

subwclcmassY The y coordinate of the weighted center of mass.

subpeakVal The maximum value.

subavVal The average value.

subtouchb TRUE if the subcell touches the boundaries.

date The date for subtsclID.

time The time for subtsclID.

5.3.3 sublinks.txt

This file contains the cell ID for each subcell, meaning that subcell A is completely covered by cell B.

subclID The subcell ID.

clID The cell this subcell is part of.

5.4 The tracks files

5.4.1 tracks.nc

This file is similar to cells.nc but with track IDs instead of cell IDs. The default file format is netCDF.

5.4.2 tracks_all.txt / tracks_clean.txt

These files list tracks with an object-oriented structure. The first line of each block (beginning with ###) gives important information about the track. The first integer is the track ID. Then, a logical value tells if the track is free of any boundary contact. The last number is the track type as explained in figure 1. Following the header, all cells of a track are listed. The only column is:

clID The cell ID.

The difference between the two files is that tracks_clean.txt only lists tracks which are of type 9 and do not touch the boundaries during their lifetime.

5.4.3 tracks_all_stats.txt / tracks_clean_stats.txt

The structure of this files is the same as in tracks_all.txt / tracks_clean.txt. However, all cells of a track are listed with their characteristics from the cell_stats.txt file.

The difference between the two files is that tracks_clean_stats.txt only lists tracks which are of type 9 and do not touch the boundaries during their lifetime.

5.4.4 tracks_all_summary.txt / tracks_clean_summary.txt

The summary files contain overall statistics for all tracks/tracks of type 9 which do not touch the boundaries. The columns represent:

trackID The cell ID.

trType The track type according to figure 1.

peakVal The maximum value.

pValtime The time step of the maximum value.

avVal The average value.

start The time step at which a track starts.

dur The duration/life time.

5.5 The meta tracks files

5.5.1 meta.nc

This file is similar to tracks.nc but with meta track IDs instead of track IDs. The default file format is netCDF.

5.5.2 meta_mainstream.nc

This file is similar to meta.nc but contains only the mainstream of each meta track. The default file format is netCDF.

5.5.3 meta_all.txt

This file lists meta tracks with an object-oriented structure. The first line of each block (beginning with ###) gives important information about the meta track. The first integer is the meta track ID. Then, a logical value tells if the track is free of boundary contact. Following the header, all tracks of a meta track are listed. The only column is:

trackID The track ID.

5.5.4 meta_summary.txt

This file contains summary statistics about the different track types in a meta track. The columns are

metaID The ID of the meta track a mainstream belongs to.

dur The duration of the meta track in time steps (from first cell to last cell).

nXX The number of tracks of a certain type (e.g. n10 for track type 10)

5.5.5 meta_mainstream.txt

Similar to the previous file this one lists all tracks of a meta track that are part of the mainstream. The logical value in the header of each block indicates whether the mainstream is free of boundary contact. The columns are the same as in tracks_all_summary.txt / tracks_clean_summary.txt.

5.5.6 meta_stats.txt

Same as the previous file but including all tracks of a meta track.

5.5.7 meta_con.txt / meta_con_pher.txt

These files contain information about the connections between tracks within a meta track. The header of a block is the same as in meta_stats.txt. The columns of meta_con.txt are:

trackID1 First track of a pair of connected tracks.

trackID2 Second track of a pair of connected tracks.

In meta_con_pher.txt there are two additional columns:

pher The final pheromone value of this connection.

dist The distance between the connecting cells of the two tracks. See section 1.9.

5.5.8 meta_mainstream_summary.txt

This file contains some summary statistics about the mainstream. The columns are:

metaID The ID of the meta track a mainstream belongs to.

dur The duration of the mainstream in time steps.

MetaValSum The total sum of the key variable (used for cell detection) for the whole meta track.

MstrValSum The total sum of the key variable (used for cell detection) for the mainstream.

valFrac The fraction of the key variable included in the mainstream (compared to the whole meta track).

References

- Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.
- Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge, MA.
- Moseley, C., Berg, P., and Haerter, J. O. (2013). Probing the precipitation life cycle by iterative rain cell tracking. *Journal of Geophysical Research: Atmospheres*, 118(24):13,361–13,370.
- Senf, F., Klocke, D., and Brueck, M. (2018). Size-resolved evaluation of simulated deep tropical convection. *Monthly Weather Review*, 146(7):2161–2182.