Lagrangian Trajectory modeling with FLEXPART

Practical Environmental Measurement Techniques

Last change	02 Jun 2019
Supervisor	Dr. Andreas Hilboll, room S3132, e-mail hilboll@uni-bremen.de
Lab	Building NW1, room S3133 (don't be confused, on the map it's S3121)

Contents

1 Introduction

1.1 What is FLEXPART?

FLEXPART (“FLEXible PARTicle dispersion model”) (Stohl et al., 2005) is a Lagrangian transport and dispersion model (Lin et al., 2012) suitable for the simulation of atmospheric transport processes. Applications range from the dispersion of radionuclides or air pollutants, over the establishment of flow climatologies, to the analysis of Earth’s water cycle. FLEXPART also produces output suitable for inverse modeling of emission fluxes, e.g., of greenhouse gases or volcanic ash.

1.2 What can FLEXPART do?

FLEXPART can simulate air mass transport both *forward* and *backward* in time.

Forward simulations can be used to trace the dispersion of, e.g., radionucleides (after a nuclear incident), volcanic ash, or wildfire plumes. There is also the option to simulate the deposition (i.e., turbulent diffusion, sedimentation, or washout by precipitation) of said species, e.g., to determine the impact on ecosystems.

Backward simulations can be used in order to investigate the origin of a given air parcel. Knowing where an air parcel has been in the past can be useful in order to understand its composition: if, e.g., you measure an enhanced concentration of an air pollutant at a given location, you can use backward simulations in order to help understand where this pollutant was originally emitted.

1.3 What can’t FLEXPART do?

FLEXPART can only simulate the *transport*, *dispersion*, and *deposition* of particles and molecules. Therefore, it is only suitable for *long-lived* species like, e.g., carbon dioxide (CO_2), methane (CH_4), black carbon, dust, and volcanic ash aerosols.

Furthermore, FLEXPART does not account for chemical reactions between different species, so it is only suitable for inert species or those with simple, linear chemistry with reactants of (approximately) known concentration, e.g., OH.

2 Experiment formalities

2.1 On the day of the experiment

On the day of the experiment, please report to Andreas Hilboll (room S3132) by 09.30 hrs.

There is a dedicated desktop workstation prepared for your experiment work in room S3133, where you will do all your work.

Since our offices are rather new, they are not yet searchable on the interactive campus map. You can find us at the same location where the former room S3121 used to exist.

The total computational time of the experiment is less than 1 hr, so this leaves you with enough time to set up the experiments and work on the data analysis.

2.2 When coming to collecting my signature

After I have corrected and approved your reports, when you come to collect my signature, please consider the following:

- Please always come in with your study partner, so I can give you both the signature at the same time.
- Have the document pre-filled, so that I only have to sign. I will not fill out the form for you.

3 How to prepare for this experiment

In order to benefit from this experiment, you should know what a Lagrangian transport model is and which kind of scientific questions it can answer.

A good introduction can be found in Lin (2013). The basic workings of the FLEXPART model are explained in Pisso et al. (2019).

3.1 Helpful questions for preparation

- Which removal processes are relevant for
  - radionucleides
  - trace gases
  - aerosol particles?

  What is their respective underlying physical principle?
- **TBD**

4 How to write the report for this experiment

The directory `${HOME}/trajmod/analysis` contains Jupyter notebooks, one for each of tasks 1 and 2 (see below). These notebooks contain code you can use to analyze your simulation results and generate plots to include in the report.

Your report should include all information necessary to answer, i.a., the following questions:

1. What did you do?
2. Why is that interesting?
3. How did you do it?
4. What are the results of your experiment?
5. What do the results of your experiment mean?
6. . . .

It is imperative that you *interpret* your results. I expect you to not only describe what you did; you also have to describe your results and explain them.

Please submit your report as one single PDF file via e-mail to hilboll@uni-bremen.de.

5 Technical aspects

5.1 Working with the lab computer

All model runs and analysis will be conducted on the *lamoslab* computer. From the lab computer, you will connect to *lamoslab* via ssh in order to run the FLEXPART model. You can edit the FLEXPART configuration files etc. using the file manager installed on the lab computer.

5.2 Directory structure on the lab computer.

All files related to this experiment are contained in the folder `${HOME}/trajmod`:

```
~/trajmod
├── analysis              # Jupyter notebooks to conduct the data analysis
├── data
│   ├── discover-aq      # Discover-AQ measurement data
│   └── emissions        # Anthropogenic and biomass burning emission data
├── flexpart              # FLEXPART source code -> don't touch!
├── manual                # this experiment manual
└── runtime
    ├── nuclear           # runtime directory for Task 1
    └── discover-aq       # runtime directory for Task 2
```

5.3 Working with the Jupyter Notebook

There is are Jupyter Notebooks available to do the data analysis part of this experiment. These notebooks are located in the `${HOME}/trajmod/analysis` folder on *lamoslab*, and you can run them from any web browser under the URL `https://aether.uni-bremen.de/lamoslab`. You can reach this URL from anywhere within the university network, i.e., also from home (when using the ZfN VPN client).

Some information on the Jupyter Notebook can be found in the following articles:

- Jupyter Notebook for Beginners: A Tutorial
- Getting Started With Jupyter Notebook for Python
- What is the Jupyter Notebook?

Since the Jupyter Notebook uses the Python programming language, some basic understanding of Python is advisable. A good resource is the free e-book A Whirlwind Tour of Python.

6 Configuring a FLEXPART model run

A FLEXPART model run is configured using the files in the `options/` subdirectory of the runtime directory. The runtime directory contains the following files:

```
~/trajmod/runtime/nuclear
```

```
├── FLEXPART_MPI -> ../flexpart/src/FLEXPART_MPI  # link to the FLEXPART executable
├── options                  # directory for configuration files
│   ├── AGECLASSES          # configuration for age classes (NOT NEEDED HERE!)
│   ├── COMMAND             # main configuration file
│   ├── IGBP_int1.dat       # land use input data
│   ├── OH_7lev_agl.dat     # climatological OH fields (NOT NEEDED HERE!)
│   ├── OUTGRID             # configuration for output grid definition
│   ├── RECEPTORS           # configuration for receptors (NOT NEEDED HERE!)
│   ├── RELEASES            # configuration file for emission sources
│   ├── SPECIES             # directory for species definition files
│   │   ├── SPECIES_002     #
│   │   ├── [ ... ]         #
│   │   ├── SPECIES_NNN     # configuration file with parameters for a single species
│   │   ├── [ ... ]         #
│   │   ├── SPECIES_040     #
│   │   └── SPECIES.README  #
│   ├── surfdata.t          #
│   └── surfdepo.t          #
├── output                   #  directory where output files will be stored
└── pathnames                #  configuration for location of input and output
```

The most important files are:

Table 1: The most important FLEXPART configuration files

Filename	**Purpose**
COMMAND	General configuration of the model run
OUTGRID	Definition of the 3-D grid on which FLEXPART will save concentration and deposition values
RELEASES	Definition of the particle release (emission/measurement for forward/backward runs)
SPECIES/SPECIES_NNN	Definition of a species (paramterization of the different loss processes)
SPECIES/SPECIES.README	List of all pre-defined species (this is a purely informational file, do not change!)

6.1 COMMAND

The file `options/COMMAND` contains the main configuration options for the FLEXPART model, e.g., information about time steps, simulation period, different parameterizations for atmospheric processes, and the units to be used in the output files.

Among others, you can select if you want a forward or backward simulation (using `LDIRECT`) and the simulation time period. The temporal sampling of the air parcel locations is controlled with the parameters `LOUTSTEP`, LOUTAVER, and LOUTSAMPLE from the configuration file options/COMMAND: there will be one concentration field / trajectory output every LOUTSTEP seconds, which will be the average of the previous LOUTAVER seconds of the air parcels, sampled every LOUTSAMPLE seconds.

Other than that, some settings regarding the units to be output (IOUT, IND_SOURCE, IND_RECEPTOR) can to be specified, and simulation of some atmospheric processes can be enabled (LSUBGRID, LCONVECTION, LAGESPECTRA).

Except for the simulation period, the COMMAND files are already prepared for you, so you do not need to modify anything in them. However, you are welcome to check the file and see what options can be set.

6.2 OUTGRID

A FLEXPART model run produces two different kinds of output: the gridded *concentration fields* (written to the file output/grid_*.nc) and the *trajectories* (written to the file output/trajectories.txt.

FLEXPART simulates the transport of air parcels. At each simulation time step, every air parcel is defined by its geolocation and mass. FLEXPART then calculates gridded 4-D (time×altitude×latitude×longitude) *concentration* and *sensitivity fields* from the air parcels, for forward and backward simulations, respectively.

The other FLEXPART output, the *trajectories*, are derived from the air parcel geolocations by taking the geolocation mean at each time step. However, in order to improve the spatial representativeness of the trajectory output, FLEXPART also calculates so-called *cluster trajectories.* For this, FLEXPART groups the air parcels into five distinct *clusters* and provides, at each output time step, the geolocation of each cluster, together with the fraction of the whole plume being represented by that cluster. That way, FLEXPART provides not one but five trajectories, which gives a better representation of the plume (Stohl et al., 2002).

The trajectories are written irrespective of any special configuration settings. For the concentration / sensitivity fields (for forward / backward simulations, respectively) as well as the dry and wet deposition, you have to define the 3-D grid on which values will be output in the file options/OUTGRID.

You can change values described in Table 2.

Table 2: Configuration parameters in the options/OUTGRID file

Parameter	**Format**	**Units**	**Description**
OUTLON0	float (i.e., write *10* as 10.0)	degrees East	geographical longitude of lower left corner of output grid
OUTLAT0	float (i.e., write *10* as 10.0)	degrees North	geographical latitude of lower left corner of output grid
NUMXGRID	integer (i.e., no decimal point)	n/a	number of grid points in x direction
NUMYGRID	integer (i.e., no decimal point)	n/a	number of grid points in y direction
DXOUT	float (i.e., write *1* as 1.0)	degrees	grid distance in x direction
DYOUT	float (i.e., write *1* as 1.0)	degrees	grid distance in y direction
OUTHEIGHTS	floats separated by commas	m (above ground)	upper boudary of the vertical amospheric layers

As for the vertical layers, it is good practice to have the uppermost layer extend to high into the stratosphere (e.g., set the highest layer to 50000.0). Otherwise, when setting the upper layer too low, the output grid might not contain all air parcels (because they have risen above that layer), making interpretation of the results difficult.

6.3 RELEASES

In FLEXPART lingo, the source or destination (for forward and backward runs, respectively) of and air parcel is called a *release.* While in principle, FLEXPART can handle multiple releases within one model run, you will only have one single release in this experiment (this makes the interpretation of the results more intuitive).

A release in FLEXPART is defined by

- the species to be released,
- the coordinates (altitude, longitude, latitude) of an *air volume* of the release,
- a start and end time of the release,
- the total mass emitted (in kg).

A release "point" in FLEXPART is defined as a *volume.* This means that in all three dimensions (x, y, z), there is a lower and an upper boundary to be defined. If you want to simulate a *point release*, you can simply set lower and upper boundary to the same value. In backward runs, this will almost always be the case. In forward runs, however, it could be that you want a release to actually come from an *area*, e.g., in the case of emissions from large-scale forest fires).

The FLEXPART configuration file to control everything about the releases is appropriately called `RELEASES`.

Since we want to simulate air masses measured (or released) at a single point in space and time, you will have to define a *point release*, i.e., the upper and lower bounds of the release volume should be set to the same numbers for each of three dimensions (note that the altitude dimension is called `z1` / `z2` in the `RELEASES` file).

There is one additional parameter in the `RELEASES` file, the *total number of particles to be released.* This is a purely technical parameter, it is *not* related to the total amount of substance released. (Rather, it defines the number of *air parcels* which FLEXPART should use in the simulation; a higher number therefore leads to a higher precision in the output, at the cost of a longer simulation runtime). In particular, *particle* in this contect does not mean *aerosol particle.*

The properties of the species of the release are defined in configuration files in the directory `options/SPECIES/`. The only generic parameter to be set is the *molecular weight.* Other than that, the parameters governing the different loss processes (wet and dry deposition, radioactive half life, reaction with the *OH radical*) can be set.

7 Task 1: Transport and deposition of radioactivity from a fictitious nuclear accident

Objective

To simulate the transport and deposition of radioactive ^{137}Cs emitted from a fictitious nuclear accident.

In this task you will run FLEXPART *forward in time* to simulate the dispersion and deposition of radioactive cesium-137 from a fictious nuclear accident.

All program and data files needed for this experiment are contained in the directory `$HOME/trajmod/runtime/nuclear` on the lab computer.

7.1 Background information

Globally, more than 400 nuclear power plants are used to generate electricity. Even though the operators take great care to avoid them, accidents regularly happen — sometimes with disastrous consequences like, e.g., in

Fukushima (Japan, 2011) and Chernobyl (1986, Ukraine).

One of the major problems arising from a nuclear accident is the emission and deposition of cesium-137 (^{137}Cs). Due to its long half-life of ∼30.1 years, deposited ^{137}Cs has deleterious effects on agriculture and stock farming, thereby badly influencing human life for decades.

Using FLEXPART, we can simulate the dispersion of the plume of radioactive ^{137}Cs and its deposition. The simulated deposition can then be used to improve the knowledge on the effects of the nuclear accident, as measurements are usually sparse and costly. For example, the simulation can indicate areas where from where agricultural produce should not be used any more.

7.2 Choose a nuclear power plant

For this simulation exercise, you are free to choose any nuclear power plant in the world. You can use, e.g., the Wikipedia list of nuclear power stations to decide which one you want to study. That list only contains the largest power stations; feel free to choose any other existing power station, e.g., from your home country.

To run FLEXPART, you will need the following information about the accident:

1. **Geolocation** (longitude/latitude/altitude): Latitude and longitude ca be found on the Wikiepdia pages of the individual power stations. You can assume an emission altitude between 20m and 50m above ground.
2. **Time** (start and end of the emission): On the *lamoslab* computer, wind fields are available for the months January, April, and July of the year 2011. Choose any 1–2 day period for the emission of ^{137}Cs so that the end of a 10-day simulation is still covered by the wind field availability.
3. **Mass** of emitted ^{137}Cs: This parameter tells FLEXPART how much ^{137}Cs is emitted into the atmosphere. Ultimately, you can freely choose this parameter. To give you an idea of realistic values, on one day (15 Mar 2011) during the Fukuchima Daiichi incident, ∼100PBq of cesium were emitted (Arutyunyan et al., 2012). For simplicity, you can use Bq as "mass units" in the simulation.

Task

1. Choose a nuclear power plant to simulate, and note its geolocation.
2. Decide on a time when the accident should happen.
3. Decide on the mass of emitted ^{137}Cs.

7.3 Preparing the model setup

All basic settings for the FLEXPART model run are done in the configuration file `COMMAND` from the directory `options/` (inside the `$HOME/trajmod/runtime/nuclear` directory). The file `COMMAND` is already prepared for this experiment. You should make sure that `LDIRECT` is actually set correctly for a forward simulation, and you need to adapt the start and end time of the simulation. You should set the simulation to start at the half-hour before the beginning of the emission. For example, when the emission starts at *14:33:00*, set the model start time to *14:30:00*. Set the end time so that the model runs for a total of 10 days.

Task

Adapt the `COMMAND` file:

1. Set simulation start date/time (last half-hour before the emission start).
2. Set simulation end date/time (10 days after the start date/time).

3. Check that LDIRECT is set to a forward simulation.

7.4 Defining the characteristics of the ^{137}Cs emission

A release in FLEXPART is defined by the species' characteristics, the coordindates and time span of the emission, and the mass.

The species' characteristics are defined in configuration files in the directory `options/SPECIES/`. A list of species and their individual characteristics can be displayed by running the command

```
./specoverview
```

from the directory `options/SPECIES`. The *actual* characteristics of a FLEXPART species are defined in the files `options/SPECIES/SPECIES_NNN`, where `NNN` is the id number of the species which is defined in the `options/RELEASES` file (under the parameter `SPECNUM_REL`).

You should check the file `options/SPECIES/SPECIES.README` for the parameters of ^{137}Cs. In particular, you will need to know the id number (in the first column) of ^{137}Cs to identify the exact parameter file (`options/SPECIES/SPECIES_NNN`) where the ^{137}Cs characteristics are defined. Check which loss processes are implemented for ^{137}Cs (i.e., which parameters have values other than *-9.9, -09E+09*, or similar).

Now, you can adapt the `RELEASES` file from the directory `options/` to set the location and emission strength (i.e., "mass", in Bq) of the accident, and the id number of the emitted species (see above).

Task

Adapt the `RELEASES` file:

1. Set release locations (`LON1/LON2`, `LAT1/LAT2`, `Z1/Z2`) to the location / altitude of the nuclear power plant. Pay attention to the definition of altitude (`ZKIND`).
2. Set release time (`IDATE1/ITIME1/IDATE2/ITIME2`) to the period of emission.
3. Set the number of air parcels to use (`PARTS`) to *100000*.

⚠ For the coordinates, FLEXPART expects *floating point numbers*. This means that even for, e.g., *60°*, you will have to specify `60.0` in the `RELEASES` file. Simply writing `60` (i.e., without the `.0`) might lead to wrong results!

7.5 Defining the output grid

In this experiment, you are only interested in the gridded output of the FLEXPART run, namely the 4-D (time×altitude×latitude×longitude) *concentration fields* and the 3-D (time×latitude×longitude) *deposition fields*. As explained in Sect. 6.2, the output grid is defined in the configuration file `OUTGRID` from the directory `nuclear/options/`. For example, you can choose a grid of 120×100 grid cells of size 0.5°, centered on the location of the power plant. You should make sure that the lower left corner of your domain has coordinates which are divisable by 0.5°.

Task

Adapt the `OUTGRID` file:

1. Decide how large your output grid should be.
2. Set the coordinates of the lower left corner of your output grid (`OUTLON0/OUTLAT0`).
3. Set `DXOUT/DYOUT` to 0.5°.
4. Calculate how many 0.5° grid cells your output grid will have in x and y direction and set `NUMXGRID/NUMYGRID` accordingly.

7.6 Running the model

Task

Start the FLEXPART simulation by running the command

```
mpirun -n 4 FLEXPART_MPI
```

from the directory `$HOME/trajmod/runtime/nuclear` on the *lamoslab* computer.

You have to wait until the simulation is finished. Leave the command line window in which the model is running open; otherwise, the FLEXPART run will abort. The model run will take approximately 15 minutes. When the run is finished, you will see the message

```
CONGRATULATIONS: YOU HAVE SUCCESSFULLY COMPLETED A FLEXPART MODEL RUN!
```

in the terminal.

The output files are now located in the directory `~/trajmod/runtime/nuclear/output/`.

7.7 Data analysis

A Jupyter Notebook for the data analysis is available in the folder `${HOME}/trajmod/analysis/Nuclear.ipynb`. It contains precise instructions for your analysis of this experiment.

Task

Work through the Jupyter notebook `$HOME/trajmod/analysis/Nuclear.ipynb` to analyze the simulation results.

8 Task 2: Determining the sources of measured air pollution

Objective

To simulate back trajectories for one single black carbon measurement and identify potential source regions.

In this experiment, you will use FLEXPART in *backward* mode to simulate the history of the air masses that were sampled by *one single measurement* of black carbon made on one flight during the 2011 DISCOVER-AQ campaign. These simulations can then be used in the interpretation of the measurements, e.g., to decide where the measured black carbon was potentially emitted into the atmosphere.

All program and data files needed for this experiment are available in the directory `$HOME/trajmod/runtime/discoveraq/`:

```
$ tree -d ~/trajmod/runtime/discoveraq/
trajmod/runtime/discoveraq/
├── data
├── options
│   └── SPECIES
└── output

4 directories
```

8.1 Background information

DISCOVER-AQ was a measurement campaign taking place in the area of Washington, D.C. in summer 2011, where the P-3B aircraft was deployed to sample the pollution in the Washington-Baltimore area (see Fig. 1).

One of the instruments on board the aircraft was measuring *black carbon* aerosol concentrations. The data are freely available via internet. You can see the strong variability of the measured black carbon in Fig. 2.

For your reference, the measurement data are available in the data files `discoveraq_bc_YYYYMMDD.csv` located in the directory `$HOME/trajmod/runtime/discoveraq/data/` on the lab computer.

8.2 Choosing which measurement you want to simulate

First, you need to choose which measurement you want to investigate. You can look at Figs. 1 and 2 to select the day and time of measurement you want to look into.

In a real scientific project, this step would usually not be carried out, as this type of analysis should be routinely done for every single measurement made during the flight.

In FLEXPART, you have to specify the location and time of the measurement. Therefore, you will have to extract this information (date, time, longitude, latitude, altitude above sea level, concentration) from the appropriate data file from the directory `$HOME/trajmod/runtime/discoveraq/data`.

Task

Looking at Figs. 1 and 2, decide for the single point in time for which you want to simulate the back trajectories. Note down the day and time, and then look into the data file for the selected day and identify latitude/longitude coordinates and flight altitude of the selcted measurement.
Usually, you will want to select a measurement which showed enhanced black carbon values, possibly close to a city, in order to check if the measured pollution actually came from that city.

8.3 Preparing the FLEXPART model run

All basic settings for a FLEXPART model run are done in the configuration file `COMMAND` from the directory `options/` (inside the `$HOME/trajmod/runtime/discoveraq` directory).

The file `COMMAND` is already prepared for this experiment. You should make sure that `LDIRECT` is actually set correctly for a backward simulation, and you need to adapt the start and end time of the simulation. (Note that the model *start* always has to be *before* the model *end*, also in case of backward runs.) You should set

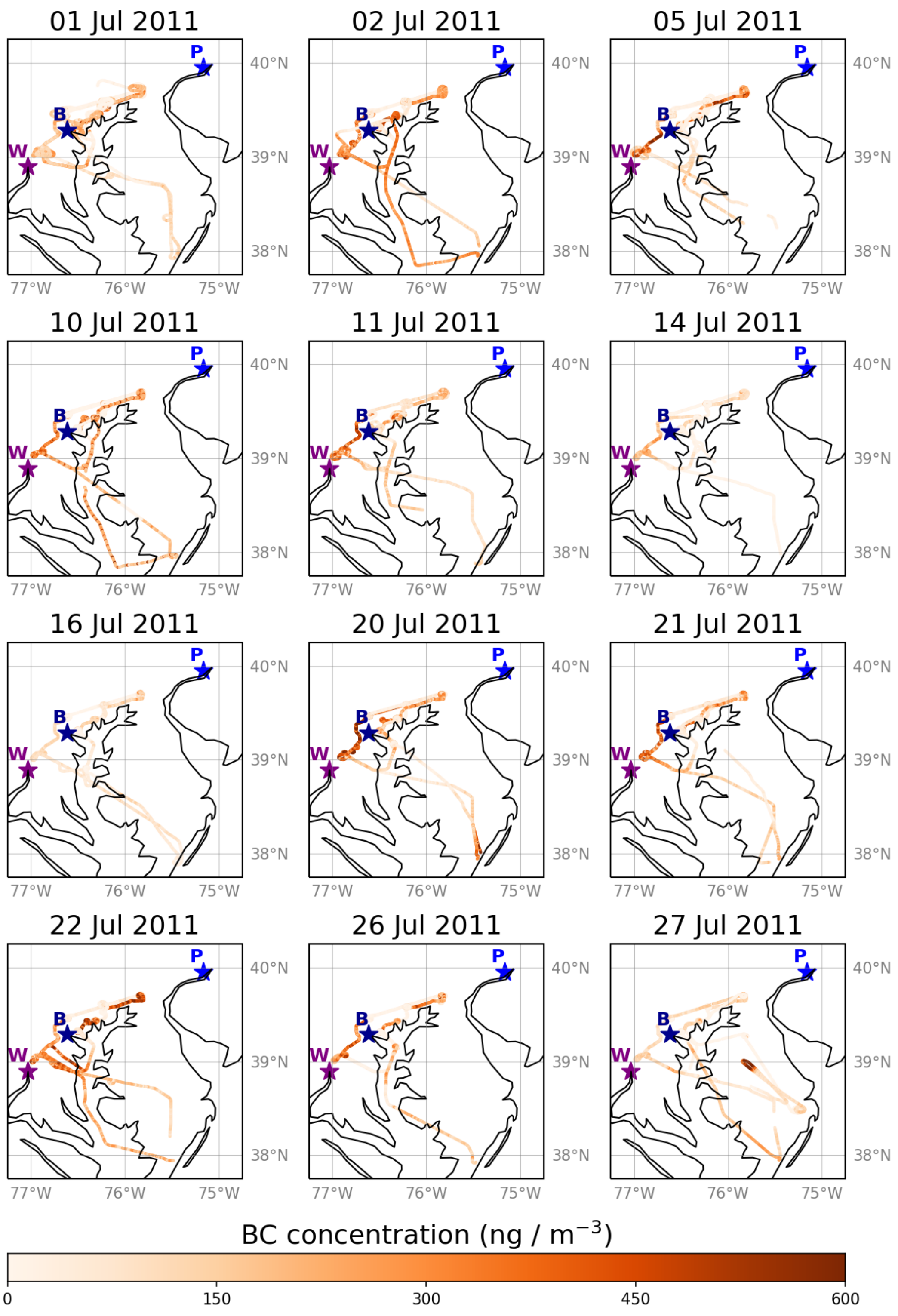

Figure 1: Flight tracks of the 2011 DISCOVER-AQ aircraft campaign. The black carbon concentration (in ng m^{-3}) is indicated by the shade of orange. The urban agglomerations of Washington DC (W), Baltimore (B), and Philadelphia (P) are marked with stars.

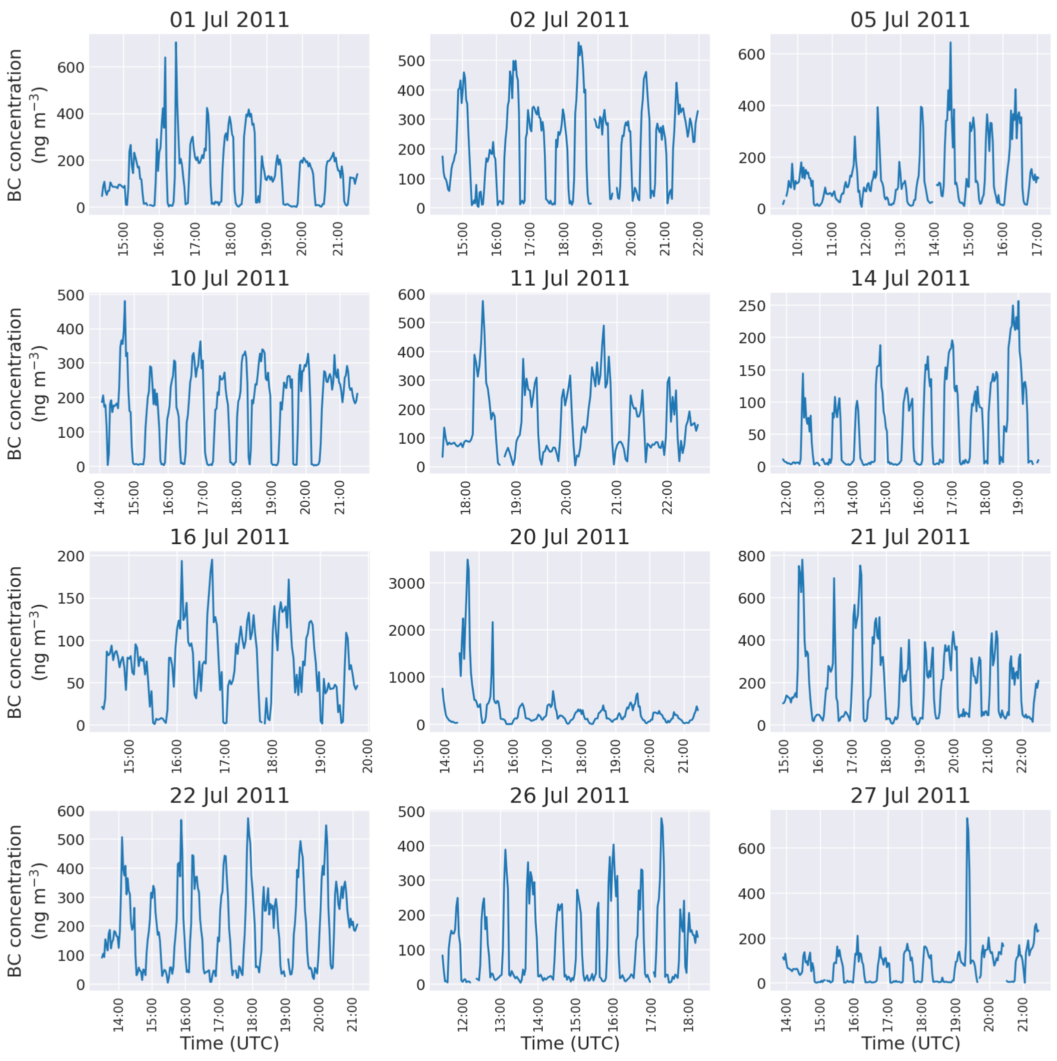

Figure 2: Time-series of black black carbon measurements during the 2011 DISCOVER-AQ aircraft campaign. Measurements have been resampled from the original 10-second interval to 2-minute intervals to yield clearer plots.

the simulation to an end at the next half-hour after the chosen measurement's time. For example, when your measurement is at *14:02:00*, set the model end time to *14:30:00*. Set the start time to 4 days before your end time, so that effectively the model runs backwards for 4 days.

Task

Adapt the `COMMAND` file:

1. Set simulation end date/time (next half-hour after the measurement).
2. Set simulation start date/time (4 days before the end date/time).
3. Check that `LDIRECT` is set to a backward simulation.

8.4 Defining the *release* (i.e., the measurement location and species)

Task

Adapt the `RELEASES` file:

1. Set release locations (`LON1/LON2`, `LAT1/LAT2`, `Z1/Z2`) to the location / altitude of the measurement. Pay attention to the definition of altitude (`ZKIND`).
2. Set release time (`IDATE1/ITIME1/IDATE2/ITIME2`) to the time of measurement.
3. Set the number of air parcels to use (`PARTS`) to *50000*.

The *total mass emitted* can be set to any arbitrary (positive) number, since FLEXPART automatically scales the results using this number — the result of a backward simulation is always a *sensitivity*.

In our case, we use a generic *black carbon tracer*, which is defined in the file `SPECIES_040`. The parameters set in this file are already selected to be correct for this experiment, so you do not need to modify this file.

8.5 Defining the FLEXPART output options

For this experiment, you can use the gridded output to estimate the *sensitivity* of "your" measurement to emissions at the Earth's surface. Since in the data analysis, you will compare these sensitivities with a gridded emission inventory of black carbon, you should choose an output grid which is "compatible" to the inventory used. The emission inventory uses a spatial resolution of 0.5°, so you should use an output grid of size 0.5° as well. That means that in the `OUTGRID` configuration file, you should set both `DXOUT` and `DYOUT` to the same value (`0.50`). You further have to define the location of the lower left corner of the output grid, using `OUTLON0` and `OUTLAT0`, and its size in grid boxes, using `NUMXGRID` and `NUMYGRID`. You can find the coordinates of the campaign area from Fig. 1; make sure that your output grid is significantly larger (i.e., covers most of North America).

Task

Adapt the `OUTGRID` file:

1. Decide how large your output grid should be.
2. Set the coordinates of the lower left corner of your output grid (`OUTLON0/OUTLAT0`).
3. Set `DXOUT/DYOUT` to 0.5°.
4. Calculate how many 0.5° grid cells your output grid will have in x and y direction and set

`NUMXGRID/NUMYGRID` accordingly.

⚠ As above, FLEXPART expects *floating point numbers* for the *coordinates* (i.e., altitude, longitude, latitude — this does not apply to the number of gridboxes `NUMXGRID` and `NUMYGRID`). This means that even for, e.g., *60°*, you will have to specify `60.0` in the `OUTGRID` file. Simply writing `60` (i.e., without the `.0`) might lead to wrong results!

8.6 Running the FLEXPART model

Task

Start the FLEXPART simulation by running the command

```
FLEXPART
```

from the directory `$HOME/trajmod/runtime/discoveraq` on the *lamoslab* computer.

You have to wait until the simulation is finished. Leave the command line window in which the model is running open; otherwise, the FLEXPART run will abort. The model run will take approximately 12 minutes. When the run is finished, you will see the message

```
CONGRATULATIONS: YOU HAVE SUCCESSFULLY COMPLETED A FLEXPART MODEL RUN!
```

in the terminal.

8.7 Data analysis

A Jupyter Notebook for the data analysis is available in the folder `$HOME/trajmod/analysis/DiscoverAQ.ipynb`. It contains precise instructions for your analysis of this experiment.

Task

Work through the Jupyter notebook `$HOME/trajmod/analysis/DiscoverAQ.ipynb` to analyze the simulation results.

Your protocol for this experiment should contain a discussion of the tasks from the notebook.

9 References

Bibliography

R. V. Arutyunyan, L. A. Bolshov, D. A. Pripachkin, V. N. Semyonov, O. S. Sorokovikova, A. L. Fokin, K. G. Rubinstein, R. Y. Ignatov, and M. M. Smirnova. Estimation of radionuclide emission during the march 15, 2011 accident at the fukushima-1 npp (japan). *Atomic Energy*, 112(3):188–193, 2012. DOI:10.1007/s10512-012-9541-6.

J. Lin, D. Brunner, C. Gerbig, A. Stohl, A. Luhar, and P. Webley, editors. *Lagrangian Modeling of the Atmosphere*. Geophysical Monograph Series. American Geophysical Union, 2012. DOI:10.1029/gm200.

J. C. Lin. *Lagrangian Modeling of the Atmosphre: An Introduction*, chapter 1, pages 1–11. American Geophysical Union (AGU), 2013. ISBN 9781118704578. DOI:10.1029/2012GM001376.

I. Pisso, E. Sollum, H. Grythe, N. Kristiansen, M. Cassiani, S. Eckhardt, D. Arnold, D. Morton, R. L. Thompson, C. D. G. Zwaaftink, N. Evangeliou, H. Sodemann, L. Haimberger, S. Henne, D. Brunner, J. F. Burkhart, A. Fouilloux, J. Brioude, A. Philipp, P. Seibert, and A. Stohl. The Lagrangian particle dispersion model FLEXPART version 10.3. *Geoscientific Model Development Discussions*, pages 1–67, 2019. DOI:10.5194/gmd-2018-333.

A. Stohl, S. Eckhardt, C. Forster, P. James, N. Spichtinger, and P. Seibert. A replacement for simple back trajectory calculations in the interpretation of atmospheric trace substance measurements. *Atmospheric Environment*, 36(29):4635–4648, 2002. DOI:10.1016/s1352-2310(02)00416-8.

A. Stohl, C. Forster, A. Frank, P. Seibert, and G. Wotawa. Technical note: The lagrangian particle dispersion model flexpart version 6.2. *Atmospheric Chemistry and Physics*, 5(9):2461–2474, 2005. DOI:10.5194/acp-5-2461-2005.

Data Analysis Part I — Transport and deposition of radioactivity from a fictious nuclear accident

Practical Measurement Techniques: Lagrangian Transport Modeling

This document is meant to aid in the analysis of *Task 1* of the trajectory modeling experiment. It is available on the lab computer in Jupyter Notebook (`*.ipynb`) format.

Some information on the Jupyter Notebook in general can be found in the following articles:

- What is the Jupyter Notebook?
- Jupyter Notebook for Beginners: A Tutorial
- Getting Started With Jupyter Notebook for Python

A more thorough introduction into working with Jupyter can be found in Chapter 1 of the excellent (and free!) Python Data Science Handbook.

Since the Jupyter Notebook uses the Python programming language, some basic understanding of Python is advisable. A good resource is the free e-book A Whirlwind Tour of Python.

1 Module imports

In the beginning, you need to import some Python modules:

```
%matplotlib inline

import datetime
import os.path

import cartopy.crs as ccrs
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import xarray as xr
```

2 Task 1: Reading FLEXPART output

The main output of this model run is a file `grid_conc_YYYYMMDDHHMMSS.nc`, located in the folder `output/`. The file is in netCDF format.

The easiest way to work with netCDF data in Python is the xarray library:

```
ds = xr.open_dataset('../runtime/nuclear/output/grid_conc_20110101000000.nc')
print(ds)
```

There are three different variables of interest in that file:

1. `spec001_mr` contains the concentration fields for each time step, altitude, longitude, and latitude. Note that vertically, the output is structured in layers; each layer's *upper bound* is contained in the variable `ds.height`
2. `WD_spec001` contains the wet deposition fields for each time step, longitude, and latitude.
3. `DD_spec001` contains the dry deposition fields for each time step, longitude, and latitude.

```
print(ds.spec001_mr)
print()
print(ds.WD_spec001)
print()
print(ds.DD_spec001)
```

Since the variables contain several not needed dimensions of size `1`, it is a good idea to *squeeze* these from the dataset:

```
ds = ds.squeeze()
```

Finally, you probably want to have the three fields (concentration, wet deposition, dry deposition) easily available in variables:

```
conc = ds.spec001_mr
wdep = ds.WD_spec001
ddep = ds.DD_spec001
```

3 Basic usage of *xarray*

3.1 Indexing / selecting

In Python, you can access individual elements / slices using square brackets (`[]`). For example, to access the first time step of the concentration field, you can

```
conc = ds.spec001_mr

print(conc[0])
```

For more details, see the xarray documentation.

3.2 Aggregation

You can aggregate a grid along one axis. For example, to sum along the `time` axis, you can

```
wetdep = ds.WD_spec001.sum('time', keep_attrs=True)

print(wetdep)
```

For more details, see the xarray documentation.

3.3 Plotting

There are a lot of examples on how to use the plotting functionality from *xarray* directly, the xarray documentation contains many examples.

For example, in order to plot a time series of the dry deposition at the grid location 10/10, you can simply

```
drydepts = ds.DD_spec001[:, 10, 10]
drydepts.plot()
```

Plotting maps is also not too complicated. For example, to plot a map of the wet deposition at time step 10:

```
ds.WD_spec001[10].plot(vmin=0)
```

You can also include coastlines (see the xarray documentation for details):

```
ax = plt.axes(projection=ccrs.PlateCarree())
ds.WD_spec001[10].plot(ax=ax, vmin=0)
ax.coastlines(color='white', resolution='50m')
```

As always, you can save a plot using

```
plt.gcf().savefig('name_of_output_file.png')
```

4 Task 2: Calculate atmospheric Cesium-137 load

Create a 1D array which contains the total atmospheric Cesium-137 load for each timestep. In order to do this, you will need to use the `.sum()` method of the `conc` array to sum over the dimensions `height`, `longitude`, and `latitude` so that you end up with a 1D array in dimension *time*.

Plot this time series. You can easily do this using the `.plot()` method of the time series object.

5 Task 3: Calculate the average altitude of the radioactive cloud

Create a 1D array which contains the average height (above ground level) of the atmospheric Cesium-137 load for each time step.

First, calculate the sum over all longitudes and latitudes, so that you have a 2D array in dimension *time* × *height*. As above, you can use the `.sum()` method of the `conc` array to do this.

```
mean_conc = ...
```

Plot this array. You can adapt the `vmax` value to set the upper limit of the color map.

```
mean_conc.plot.pcolormesh(
    x='time', y='height', vmin=0.0, vmax=10.0)
```

Next, you will need to know the altitude (agl) of each model layer. You can do this using the coordinate variable `height` from the `conc` array, which contains the *upper boundary* of each model layer. So first, we calculate the *thickness* of each model layer:

```
# replace thickness of the lowest model layer
thickness_list = [conc.height.values[0]]
# thickness of the other model layers
thickness_list += conc.height.diff('height').values.tolist()

# create layer thickness array with correct metadata
layer_thickness = xr.DataArray(
    thickness_list, coords=conc.height.coords, dims=conc.height.dims,
    name='layer_thickness')

print(layer_thickness)
```

Now, you need to calculate the mass of each model layer, by multiplying the `mean_conc` by the `layer_thickness`:

```
mass_ts =
```

Plot this array. Again, you can change the `vmax` value to adapt to your range of values.

```
mass_ts.plot.pcolormesh(
    x='time', y='height', vmin=0.0, vmax=10000.0)
```

In order to better visualize the range of values, you might want to plot the logarithm of the mass instead:

```
np.log10(mass_ts).plot.pcolormesh(...)
```

Next, calculate the mid-layer altitude for each model layer by subtracting half of the `layer_thickness` from each upper boundary (remember that the upper boundaries are stored as `conc.height`):

```
layer_mid =
```

Finally, calculate the average of the altitude h, weighted by the mass m:

$$h_{mean} = \frac{\sum_i m_i \cdot h_i}{\sum_i m_i}$$

```
h_mean = (mass_ts * layer_mid).sum('height') / mass_ts.sum('height')
```

Plot this time series.

```
h_mean.plot()
```

Describe the mean altitude of the radioactive cloud in your own words.

6 Task 4: Calculate Cesium-137 deposition time series

Create a 1D array of the total Cesium-137 dry deposition happening at each time step, and plot this array.

Note that the `DD_spec001` variable contains, for each longitude/latitude pair, the *total* dry deposition at that location *since the beginning of the model run.* In other words, in order to get the amount of dry deposition *at each time step*, you have to subtract from each value the previous (in time) value. You can do this using the `.diff()` method (see here).

```
ddep_map = ddep.diff('time')
```

Again, you need to sum over the horizontal extents of the model domain:

```
ddep_ts = ...
```

Finally, plot this time series.

```
ddep_ts.plot()
```

Describe the plot in your own words. Why does the dry deposition show the pattern you observe?

Now, repeat the same for wet deposition:

7 Task 5: Follow the radioactive cloud

Create daily plots of the vertically integrated atmospheric load of Cs-137.

Start by calculating the mass of each grid cell, by multiplying the concentration field `conc` with the `layer_thickness`:

```
mass = ...
```

Now, you can vertically integrate this mass, by using the `.sum()` method of the `mass` array to sum over the `height` dimension:

```
column = ...
```

Now, in order to reduce the size of the array, we choose only one time step per day:

```
column_daily = column.sel(time=datetime.time(0))
```

At this point, `column_daily` is a 3D array with dimensions *time* × *longitude* × *latitude*:

```
print(column_daily)
```

Finally, you can plot these daily maps:

```
column_daily.plot.pcolormesh(
    x='longitude', y='latitude', col='time',
    vmin=0.0, vmax=100.0,
    col_wrap=4)
```

As above, adapt `vmax` to your liking, and consider plotting the logarithm instead.

Describe the path of the radioactive cloud in your own words.

Data Analysis Part II — Analyzing potential sources of measured air pollution

Practical Measurement Techniques: Lagrangian Transport Modeling

This document is meant to aid in the analysis of *Task 2* of the trajectory modeling experiment. It is available on the lab computer in Jupyter Notebook (`*.ipynb`) format.

Some information on the Jupyter Notebook in general can be found in the following articles:

- What is the Jupyter Notebook?
- Jupyter Notebook for Beginners: A Tutorial
- Getting Started With Jupyter Notebook for Python

A more thorough introduction into working with Jupyter can be found in Chapter 1 of the excellent (and free!) Python Data Science Handbook.

Since the Jupyter Notebook uses the Python programming language, some basic understanding of Python is advisable. A good resource is the free e-book A Whirlwind Tour of Python.

1 Preamble

1.1 Module imports

In the beginning, you need to import some Python modules:

```
%matplotlib inline

import datetime
import os.path

import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import xarray as xr
```

The following import will only work on the lab computer. It is only needed to load the output file `trajectories.txt`. So after you convert the original FLEXPART output to netCDF format, you can continue analyzing the data using *xarray*.

```
%run flexpartio.py
%run flexpartplot.py
```

1.2 Plotting configuration

We set some default values for plotting:

```
mpl.rcParams['axes.titlesize'] = 'xx-large'
mpl.rcParams['axes.labelsize'] = 'x-large'
mpl.rcParams['xtick.labelsize'] = 'large'
mpl.rcParams['ytick.labelsize'] = 'large'
mpl.rcParams['lines.color'] = 'r'
```

2 Reading FLEXPART output

The back trajectories are stored in the file `output/trajectories.txt`. This is an ASCII file which can be read by `pandas.read_csv()`. However, since the file format is a bit awkward, you can use this code to read the trajectories into an `xarray.Dataset`:

```
_, _, traj = read_trajectories('../runtime/discoveraq/output/trajectories.txt')
```

`traj` is now a *xarray.Dataset* having two dimensions: *cluster* and *time*. This is because instead of getting just one trajectory, FLEXPART output contains 5 distinct trajectories (called *clusters*). For each of the clusters, the following information are relevant:

- longitude (in `traj.longitude`)
- latitude (in `traj.latitude`)
- altitude (in `traj.altitude`)
- fraction (in `traj.c_fraction`)

The latter contains the percentage of the backward plume per cluster. For for each time step, the sum of all `c_fraction` is 100.

```
print(traj)
```

3 Task 1: Plotting the back trajectories

You can create a plot of the back trajectories using the following command:

```
lf.plot.plot_clusters(
    traj, color='altitude', marker='age', cmap='inferno',
    draw_map=True, freq='3H', scale=200)
```

You can modify the `scale` parameter to adjust the size of the symbols.

The plot then shows five distinct trajectories, which are as a whole representative of the whole backward simulation plume. Symbols are scaled according to the fraction of the whole backward plume they represent. The symbol shape indicates the temporal dimension; each day has a different marker. For example, squares (labeled as *1 day*) indicate the locations of the air masses 1 day ago, circles show the locations of the air masses 3 days ago, and so on. The altitude of the air parcels is indicated by the color of the symbols.

#+ipynb-newcell

Describe, in your own words, the mean path of the back trajectories. Discuss potential source regions of the measured black carbon pollution.

4 Task 2: Altitude of the back trajectories

Create a line plot of altitude vs. time for all five clusters. The altitude is contained in the variable `tract.altitude`.

5 Task 3: Sensitivity to emissions

In addition to the back trajectories, FLEXPART creates 4-D fields (time×height×latitude×longitude) of the *sensitivity* of the release (i.e., the measurement) to individual grid cells of the atmosphere. The unit of these sensitivities is *seconds*, and they can be interpreted as the *residence time* of air parcels in a given grid cell: the more time an air parcel (which ultimately ends up at the time and location of measurement) spends in a given grid cell, the more the measurement is influenced by (i.e., sensitive to) emissions entering the atmosphere in exactly that grid cell.

The gridded output is contained in a file `grid_time_YYYYMMDDHHMMSS.nc` in the `output/` directory. As it is a regular NetCDF file, you can use the xarray library to open it.

Now, open the gridded FLEXPART output into a variable `ds`, and create a variable `sensitivity` which points to the correct field:

```
sensitivity = read_flexpart('PATH/TO/grid_time_XXXXXXXXXXXXXX.nc')
```

This variable now contains the *sensitivity* (residence time) of the *release* (measurement) to air that was at the time and location indicated by the coordinates of this array `sensitivity`:

```
print(sensitivity)
```

5.1 Sensitivity to the surface

Since emissions to the atmosphere usually occur at the surface, we are mostly interested in the lowest layers of the `sensitivity` array. As you can see above, the dimensions of the sensitivity array are time, height, lat, lon. For now, you should create a new array of the surface sensitivity only:

```
surface_sens = sensitivity.BC_sens[:, 0, :, :]
```

This is now a 3-D array with dimensions time×lat×lon:

```
print(surface_sens)
```

Now, if we ignore the age of the air parcels, we can *sum* over the time dimension:

```
total_surface_sens = surface_sens.sum('time')
```

You should plot a map of this total surface sensitivity. To which areas was your chosen measurements most sensitive?

5.2 Dependence of Sensitivity on altitude

Now, the height at which emissions enter the atmosphere depends very much on the process which leads to the emission. For example, road traffic emissions are very local to the surface (the lowest ~10m), power plants usually have a rather high stack (100–300m). Wildfires, on the other hand, are different: while the burning biomass is located at the surface, the very high temperature of these fires leads to the emissions rising a significant height very quickly. Therefore, models usually distribute biomass burning emissions along the lowest 1000–2000m.

In order to evaluate the effect this has on the FLEXPART measurement sensitivities, we define the following function which vertically integrates the 4-D sensitivity fields:

```
def integrate_sensitivity(data, zmax, zmin=0.0):
    """Vertically integrate FLEXPART sensitivity output

    This function is used to prepare the sensitivity output from FLEXPART for multiplication with e

    Parameters
    ----------
    data : xarray.Dataset

    The sensitivity data to integrate.  This will usually come from a call to ‘read_flexpart‘.

    zmax : float
The upper limit of integration.

    zmin : float, optional
The lower limit of integration.  Defaults to 0.0 (i.e., surface)

    Returns
    -------
    sens : xarray.DataArray
The vertically integrated sensitivity field

    Notes
    -----
    This function does not do interpolation, i.e., when FLEXPART output has levels 100m, 200m, 300m

    """
    # TODO optionally, interpolate to take into account partial layers

    ix_lower = data.hgt_b[:, 0].searchsorted(zmin, 'left')
    ix_upper = data.hgt_b.shape[0] if zmax >= data.hgt_b[-1, 1] else data.hgt_b[:, 1].searchsorted(
    z_lower = data.hgt_b[ix_lower, 0].values
    z_upper = float(data.hgt_b[ix_upper - 1, 1])
    dz = z_upper - z_lower

    sens = data.isel(hgt=slice(ix_lower, ix_upper)).sum('hgt')
    for var in sens.variables:
if not var.endswith('_sens'):
    continue
sens[var].attrs['dz'] = dz
sens[var].attrs['long_name'] = 'Integrated sensitivity ({:.0f}-{:.0f}m)'.format(z_lower, z_upper)
sens[var].attrs['units'] = 's'

    return sens.BC_sens
```

Now, we can use this function to integrate the sensitivities up to . different levels (remember to put the actual altitudes into the function on the last line below; these should be part of our height definition in the `OUTGRID` file):

```
def make_sensitivities_array(sens, levels):
    if len(levels) != 6:
raise ValueError("You must use 6 levels")
    data = []
    for lev in levels:
data.append(integrate_sensitivity(sens, lev).sum('time', keep_attrs=True))
    data = xr.concat(data, 'upper_bound').assign_coords(upper_bound=levels)
    data.attrs['long_name'] = 'Integrated sensitivity'
    return data

allsens = make_sensitivities_array(sensitivity, [LIST, OF, UPPER, LEVELS, YOU, WANT])
```

Finally, you can plot the different integrated sensitivities:

```
def plot_sensitivities(sens):
    states_provinces = cfeature.NaturalEarthFeature(
category='cultural',
name='admin_1_states_provinces_lines',
scale='50m',
facecolor='none')

    sens.plot.pcolormesh(
col='upper_bound', col_wrap=2, vmin=1e-3, norm=mpl.colors.LogNorm(),
figsize=(16., 12.),
subplot_kws={'projection': ccrs.PlateCarree()})

    for ax in plt.gcf().axes:
try:
    ax.coastlines('50m')
    ax.set_extent([sens.lon.min(), sens.lon.max(), sens.lat.min(), sens.lat.max()])
    ax.add_feature(cfeature.BORDERS)
    ax.add_feature(states_provinces, edgecolor='gray')
    ax.add_feature(cfeature.LAKES, color='lightgray', alpha=0.5)
    ax.add_feature(cfeature.RIVERS, edgecolor='lightgray', alpha=0.5)
    ax.add_feature(cfeature.OCEAN, color='gray')
except AttributeError:
    pass

plot_sensitivities(allsens)
```

6 Task 4: Contribution of emission sources to the measurement

Now that you have calculated the sensitivity of the chosen measurement to emissions at different altitudes, you can estimate where the measured black carbon was emitted. In order to do this, you can multiply the sensitivities with an emission inventory.

Here, we will use two different emission inventories: *MACCCity* for anthropogenic emissions (i.e., traffic, heating, industry, . . .), and *GFED* for biomass burning (e.g., forest fires).

6.1 Plotting helper function

Again, you will create several plots comparing the anthropogenic and biomass burning influence on the measurement. Here, we define a helper function which creates two maps side-by-side:

```
def plot_dual(data_anthr, data_fire, vmin, vmax, title='', lognorm=False):
    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18, 7), subplot_kw={'projection': ccrs.PlateCa

    states_provinces = cfeature.NaturalEarthFeature(
category='cultural',
name='admin_1_states_provinces_lines',
scale='50m',
facecolor='none')

    data_anthr.plot.pcolormesh(
vmin=vmin, vmax=vmax, cmap='plasma_r', norm=(mpl.colors.LogNorm() if lognorm else None), ax=ax1,
cbar_kwargs={'orientation': 'horizontal'})
    ax1.set_title('{} (anthropogenic)'.format(title), fontsize='xx-large')

    data_fire.plot.pcolormesh(
vmin=vmin, vmax=vmax, cmap='plasma_r', norm=(mpl.colors.LogNorm() if lognorm else None), ax=ax2,
cbar_kwargs={'orientation': 'horizontal'})
    ax2.set_title('{} (fires)'.format(title), fontsize='xx-large')

    for ax in [ax1, ax2]:
ax.set_extent([-140., -60., 25., 60.])
ax.coastlines('50m')
ax.add_feature(states_provinces, edgecolor='gray')
ax.add_feature(cfeature.BORDERS)
ax.add_feature(cfeature.LAKES, color='lightgray', alpha=0.5)
ax.add_feature(cfeature.RIVERS, edgecolor='lightgray', alpha=0.5)
ax.add_feature(cfeature.OCEAN, color='gray')
```

6.2 Looking at the emission data

The monthly average emission fluxes of black carbon from both datasets are already prepared at a horizontal resolution of 0.5°×0.5°, in units of $\frac{kg}{m^2 \cdot s}$. First, you should load them:

```
emis = xr.open_dataset('../data/discoveraq_emissions_2011.nc')
print(emis)
```

Now, you can plot the two emission datasets side by side:

```
plot_dual(emis.emis_bc_anthro, emis.emis_bc_fire, 1e-13, 1e-9, 'Emissions', lognorm=True)
```

TASK: Briefly describe the spatial pattern of anthropogenic and biomass burning emissions.

6.3 Choosing different sensitivities for anthropogenic and biomass burning emissions

As outlined in the previous task, the effective injection height of anthropogenic and biomass burning emissions differs a lot. Therefore, it makes sense to use different integrated sensitivities for the two.

First, we (re-)create the sensitivitiy arrays for the layers you choose (remember to use the correct upper bounds instead of the `ANTHRO` and `FIRES`). We directly sum over the *time* dimension: since we are using a *monthly* emission inventory anyways, we can apply the same emission flux to the full, integrated (in time) sensitivities.

```
sens_anthro = integrate_sensitivity(sensitivity, ANTHRO).sum('time', keep_attrs=True)
sens_fire = integrate_sensitivity(sensitivity, FIRES).sum('time', keep_attrs=True)
```

TASK: Describe which integrated sensitivities you want to use for anthropogenic and biomass burning emissions, respectively, and explain why.

You should plot the two chosen integrated sensitivities side-by-side:

```
plot_dual(sens_anthro, sens_fire, 1e-2, 1e+4, 'Sensitivity', lognorm=True)
```

TASK: Describe the two integrated sensitivities and point out their differences.

6.4 Calculating the contribution to the black carbon measurement

The contribution of an emission inventory to the measured black carbon concentration can be calculated by simply multiplying the emission flux with the integrated sensitivity. However, one subtlety has to be taken into account: FLEXPART does not calculate the sensitivity to emission *fluxes* (in $\frac{kg}{m^2 \cdot s}$), but instead to *concentration changes* (in units of $\frac{kg}{m^3 \cdot s}$). So you also need to convert the fluxes into concentration changes. This can simply be done by dividing the fluxes by the height (im m) of the layer into which you insert them.

The following function takes care of all this (this is why in the function `integrate_sensitivity` above, we added the layer thickness `dz`):

```
def calc_contribution(emis, sens):
    arr =  ((emis * sens) / sens.attrs['dz'])
    arr.attrs['units'] = 'kg m-3'
    arr.attrs['long_name'] = 'Contribution to BC measurement'
    return arr
```

Now, it is time to calculate the contribution maps for anthropogenic and biomass burning emissions (remember to use the correct slices of the `allsens` array):

```
contrib_anthr = calc_contribution(emis.emis_bc_anthro, sens_anthro)
contrib_fire = calc_contribution(emis.emis_bc_fire, sens_fire)
```

First, you should plot the contribution maps:

```
plot_dual(contrib_anthr, contrib_fire, 1e-15, 1e-10, 'Contribution', lognorm=True)
```

Now, the interesting question is, to what extent does this explain the measured black carbon concentration? In order to do that, you should sum up all contributions and compare to the measured concentration (from the DiscoverAQ data file):

```
total_anthr_contrib = contrib_anthr.sum().values
print('Total anthropogenic contribution:', total_anthr_contrib)

total_bburn_contrib = contrib_fire.sum().values
print('Total biomass burning contribution:', total_bburn_contrib)
```

TASK: Discuss these results. What are possible explanations for the discrepancy between the measured black carbon concentration and the total simulated contributions?