SEDOO-AERIS

ACTRIS Footprints

Technical Document
& User Manual

	Name	Organisation	Date	Visa
Written by:	Daria Malik	Magellium		
Checked by:	Vanessa Pedinotti	Magellium		
Approved by:	Béatrice Berthelot	Magellium		

Document reference:	
Edition.Revision:	1.0
Date Issued:	30/05/2024
Customer:	
Ref. Market, consultation:	

Distribution List

	Name	Organisation	No. copies
Sent to :	D.Boulanger P. Henry	OMP OMP	1 1
Internal copy :	V. Pedinotti B.Berthelot	Magellium Magellium	1 1

Document evolution sheet

Ed.	Rev.	Date	Purpose evolution	Comments
1	0	30/05/2024	Creation of document	

Dissemination level

U	Public	x
P	Restricted to other programme participants	
E	Restricted to a group specified by the consortium	
O	Confidential, only for members of the consortium	

Contents

1. ACTRIS Overview

ACTRIS-FR [1] is the French component of ACTRIS for the observation and exploration of aerosols, clouds and reactive gasses and their interactions. ACTRIS is a distributed European research infrastructure, supporting research on climate and air quality. It improves understanding of the evolution of atmospheric processes and composition. ACTRIS provides users with information on the 4-D variability of short-lived species with a high level of quality, from observation and exploration platforms.

ACTRIS operates central platforms (data and calibration centers) and provides services intended for a large community of users working on chemistry/climate models, on the validation of satellite data or on the analysis of weather forecasting or of air quality. Finally, ACTRIS offers physical or remote access to its platforms serving scientific communities and the private sector, thus promoting research, training and technological innovation.

For this project, the objective was to set up a chain of systematic application of FLEXPART and the SOFT-IO tools to the in-situ measurement stations of ACTRIS-France with the development of a subsequent visualization tool. FLEXPART [2] is a Lagrangian particle dispersion model which makes it possible to estimate the backward trajectories of plumes of aerosol or gas particles. SOFT-IO (“Source attribution using FLEXPART and carbon monoxide emission inventories”) [3] is a tool developed within SEDOO department of the Observatoire Midi-Pyrénées which allows the estimation of the geographical origin of pollutants and emission sources which are at origin of CO increases in the troposphere and lower stratosphere. Combined with the in-situ measurements of the stations, the products provided by these two tools will serve as a complement for the data analyzes and atmospheric monitoring that SEDOO-AERIS would like to set up and offer to its users.

1.1 ACTRIS stations

Six ACTRIS stations were selected for the processing chain (in parentheses are indicated the altitudes of the stations):

- SIRTA Atmospheric Research Obs (162m)
- Puy de Dôme (1465m)
- Pic du Midi (2877m)
- Observatoire Perenne de l'Environnement (392m)
- the Maïdo observatory (Reunion, 2160m)
- Lamto (Côte d' Ivory, 105m)

.

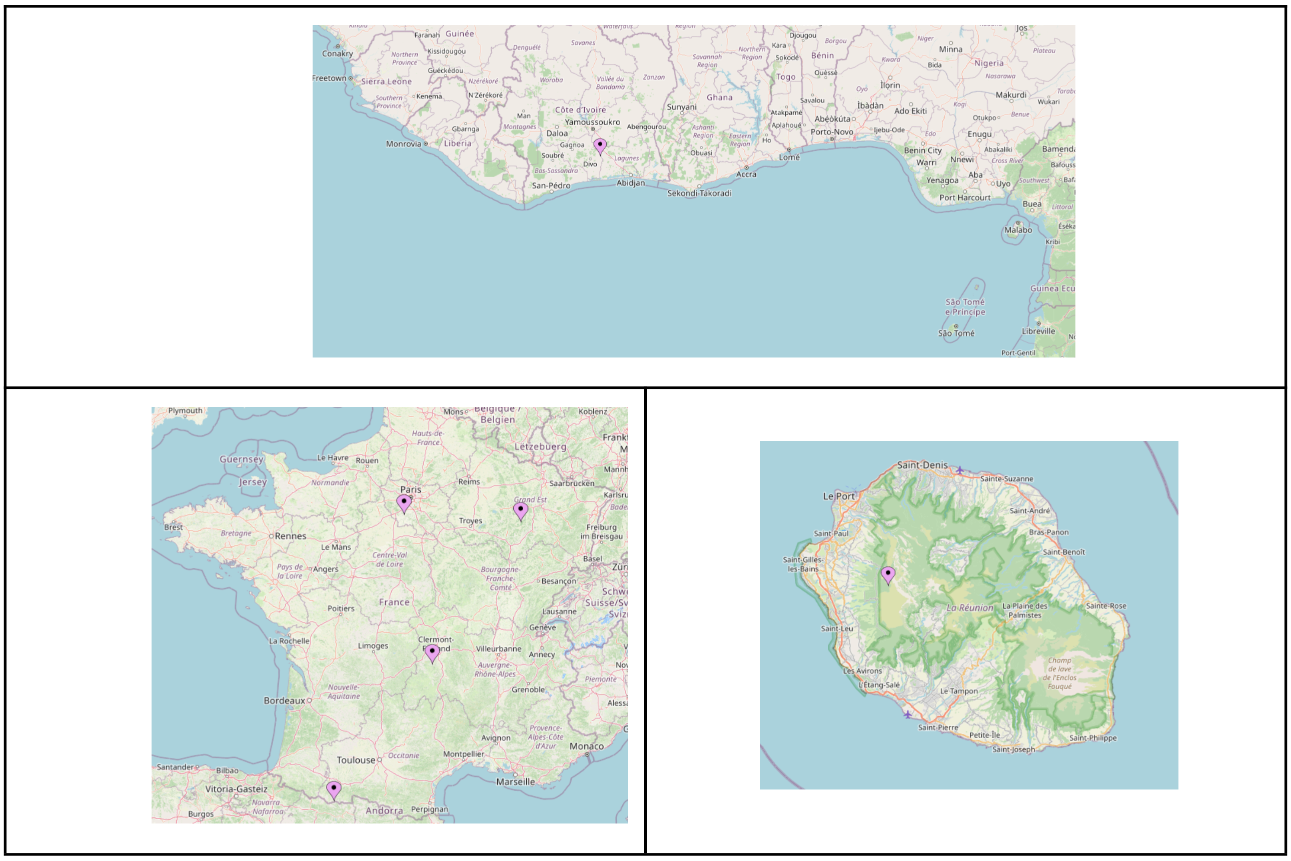


Figure 1 : Locations of the selected ACTRIS stations

These stations present very diverse profiles due to their different altitudes and locations (urban, mountainous, coastal regions, etc.). This allows to study different scenarios and observe different phenomena.

1.2 Expected output

The expected outputs of the processing chain are :

1. output of FLEXPART model, which represents the backward trajectories estimated from meteorological data (Fig. 2)
2. estimated contribution of different geographical areas and emission sources to these calculated trajectories (Fig. 3).

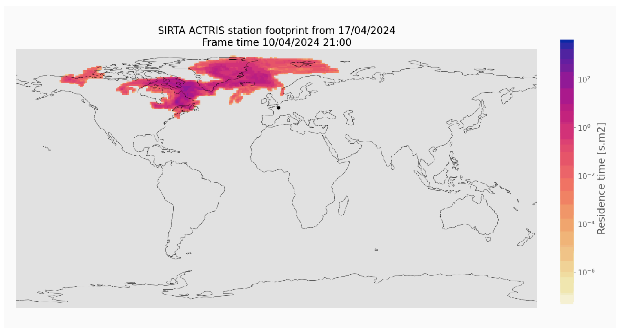

SIRTA ACTRIS station footprint from 17/04/2024
Frame time 10/04/2024 21:00
Residence time [s.m2]

Figure 2 : The output of FLEXPART is the estimated trace of particles arriving at the SIRTA station; the image is the particle position at a certain timestamp along the calculated trajectory

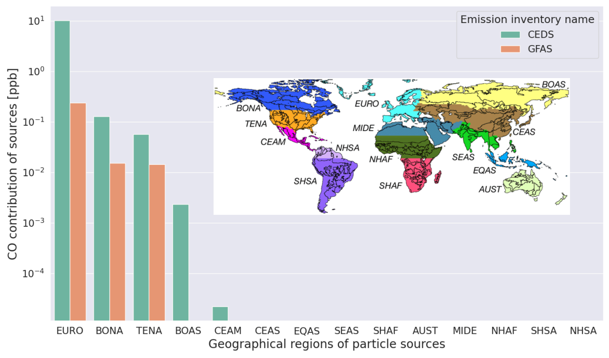


Figure 3 : The estimated contribution of CO from the different geographical regions to the measurement of the SIRTA station on 04/17/2024, calculated by SOFT-IO from the results of FLEXPART. CEDS is an inventory of anthropogenic emissions, GFAS - biomass burning emissions.

The backward trajectories of FLEXPART output allows to analyze what were the upstream influences of the measurements obtained at the stations. After that, as it was already mentioned before, the SOFT-IO tool makes it possible to estimate the contribution of different geographical areas and emission sources in the trajectories estimated by FLEXPART.

Once the processing chain is ready, a visualization tool is to be implemented in order to allow users and scientists to study and analyze chain production results.

2. Processing chain tools

Processing chain for this project contains three main steps :

1. FLEXPART backward simulation
2. CO contribution estimation by SOFT-IO
3. creation of the visualization algorithm

Two of these steps require external tools such as FLEXPART and SOFT-IO which will be detailed in this section.

2.1 FLEXPART tool

ACTRIS chain production is based on the FLEXPART simulation tool which is a Lagrangian transport and dispersion model suitable for the simulation of a large range

.

of atmospheric transport processes [2]. Hence, the main executable that launches simulations is the FLEXPART executable.

The executable needs : 1) FLEXPART input configuration files which follow particular syntax and also 2) the access to the meteorological data in the GRIB format (Fig. 4). The syntax of the input files is detailed in the FLEXPART documentation [2].

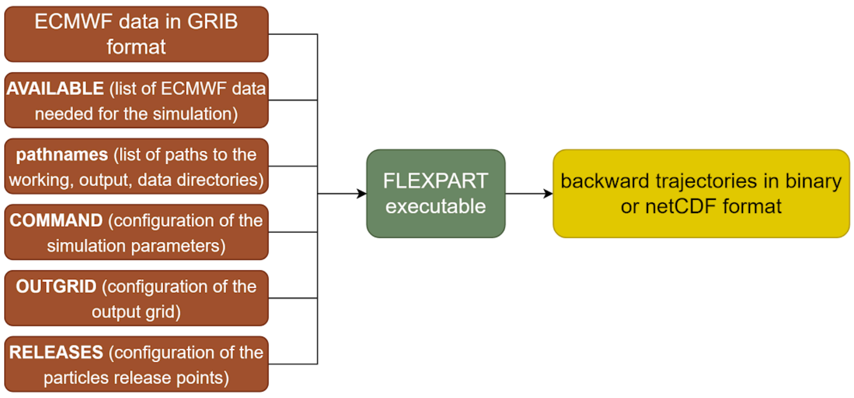

ECMWF data in GRIB format
AVAILABLE (list of ECMWF data needed for the simulation)
pathnames (list of paths to the working, output, data directories)
COMMAND (configuration of the simulation parameters)
OUTGRID (configuration of the output grid)
RELEASES (configuration of the particles release points)
FLEXPART executable
backward trajectories in binary or netCDF format

Figure 4: scheme of the FLEXPART simulation input/output which is the main element of the GIRAFE tool

FLEXPART can be run either forward or backward in time. For forward simulations, particles are released from one or more sources, concentrations (or mixing ratios) are then determined on a regular latitude–longitude–altitude grid. In backward mode, the location where particles are released represents a receptor (e.g., a measurement site). And the output represents the mean estimated residence time of the particles in latitude-longitude-altitude grid cells. In this project, the backward trajectories were estimated via backward run mode of the FLEXPART tool.

2.1.1 FLEXPART input files

FLEXPART needs the following three types of input files [2] :

1. The text file `pathnames` which must be located in the directory from which FLEXPART is executed. It must contain exactly 4 lines:
    - the path to the working directory where the simulation will be executed and where run-defining input files are located (the so-called `options` directory, explained later in this section)
    - the path where output files will be created
    - the path to the directory with meteorological input GRIB files
    - the path to the so-called `AVAILABLE` file (see point 3)
2. Text files with the run-defining settings located in a subdirectory called `options` (given in line 1 of `pathnames`). The brief description of these files is given in Table 1.
3. The meteorological input data, one file for each input time, stored in GRIB format in a common directory (specified in line 3 of `pathnames`). To enable FLEXPART to find these files, a file named `AVAILABLE` (given in line 4 of `pathnames`)

contains a list of all available meteorological input files and their corresponding time stamps (Fig 5).

Table 1 : Alphabetical list of the run-defining input files (upper part) and static input files (lower parts).

File name	Content
`AGECLASSES`	Age class definitions
`COMMAND`	Main control parameters
`OUTGRID`	Output grid definition
`RECEPTORS`	Receptor locations for receptor kernel output
`RELEASES`	Specification of the sources (forward run) or receptors (backward run)
`SPECIES/`	Directory containing files with definitions of physical and chemical parameters of species referenced in RELEASES
`IGBP_int1.dat`	Land cover input data
`surfdata.t`	Roughness length, leaf area index for different land cover types
`surfdepo.t`	Seasonal surface resistances for different land cover types

```
XXXXXX EMPTY LINES XXXXXXXXX
XXXXXX EMPTY LINES XXXXXXXXX
YYYYMMDD HHMMSS   name of the file(up to 80 characters)
20240421 000000      EN24042100      ON DISK
20240421 030000      EN24042103      ON DISK
20240421 060000      EN24042106      ON DISK
20240421 090000      EN24042109      ON DISK
20240421 120000      EN24042112      ON DISK
...
```

Figure 5 : Example of the content of an AVAILABLE file

2.1.2 Run-defining settings

This subsection, describes in more detail the run-defining setting files, their contents and their role. These settings control FLEXPART's physics and program flow.

2.1.2.1 File COMMAND

The `COMMAND` file contains the user settings controlling the simulation and the behavior of the run. Parameters such as simulation begin and end datetime, output type and format, interval of model output and others are defined in this file. A complete listing of all settings with their meaning and preset default values can be found in the FLEXPART documentation [2].

2.1.2.2 File RELEASES

The `RELEASES` file contains details regarding the introduction of particles in the simulation, including information on the timing, location, and characteristics of release points. The header of this file includes the total number of different species intended for release, accompanied by a corresponding list of FLEXPART species numbers (`nnn`). The `SPECIES_nnn` files further define the physical properties of these species. Following the header, an arbitrary number of namelists `&RELEASE` is entered, each defining a distinct release. For each release, the provided information includes the start and end times of the release, spatial coordinates and dimensions, released masses (with one value per species), the quantity of particles slated for release, and an accompanying comment string. For the ACTRIS project, the species that is used for backward FLEXPART runs is the air-tracer, which is configured precisely for the backward simulations.

2.1.2.3 File OUTGRID

The `OUTGRID` file delineates the domain and grid spacing for the three-dimensional output grid. It is important to note that, in a Lagrangian model, the specifications for the domain and resolution of the gridded output are entirely distinct from those pertaining to the meteorological input. The only requirement is that the output domain must be contained within the computational domain.

2.1.2.4 File RECEPTORS

In addition to gridded model output, it is also possible to define receptor points. This option allows to produce the output for certain points at the surface. The `RECEPTORS` file contains a list with the definitions of the receptor name, longitude and latitude.

2.1.3 Meteorological data

FLEXPART is capable of operating with meteorological input data tailored for global domains or more localized, limited-area domains. The computational domain in FLEXPART aligns with the domain established by the input data, while the output domain can be configured to be smaller.

The compilation of FLEXPART version 10.4 results in a single executable that automatically discerns whether the meteorological input data originate from ECMWF IFS or NCEP GFS, and whether they are formatted in GRIB-1 or GRIB-2. However, adjustments to certain parameters in the source `par_mod.f90` file may be necessary to accommodate the size of meteorological input files, particularly in terms of array dimensions. Additionally, the input grid might require shifting relative to the output grid (`nxshift` parameter).

The meteorological data supported by FLEXPART can be extracted via `flex_extract` tool. `Flex_extract` is an open-source software which allows to retrieve meteorological fields from the MARS archive of the European Centre for Medium-Range Weather Forecasts (ECMWF) to serve as input for the FLEXTRA/FLEXPART atmospheric transport modeling system. The installation and the usage of the `flex_extract` tool are presented below.

2.1.3.1 Installation of flex_extract

The following installation steps are destined for the member-state users of the ECMWF MARS server, and the installation is meant to be done on the MARS server. Other installation configurations are presented in the `flex_extract` official user guide [4]. Once you logged in onto the MARS server (ecs or hpc), you can follow the below steps.

First, download the `dev` branch of the `flex_extract` tool via `git clone`. This will create a `flex_extract` local git repository on your machine:

```
$ git clone --single-branch --branch dev https://www.flexpart.eu/gitmob/flex_extract
$ ls flex_extract
Documentation/          For_developers/            Run/
Source/                 Templates/                 Testing/
CODE_OF_CONDUCT.md      LICENSE.md                 README.md
setup.sh                setup_bologna.sh           setup_local.sh
setup_local_bologna.sh  setup_local_reading.sh     setup_reading.sh
```

The script `setup_bologna.sh` is intended to prepare installation files from the source of the git repository. The beginning of this script contains parameters that must be configured by the user.

```
$ cat setup_bologna.sh

#!/bin/bash
#...
TARGET='ecs'
MAKEFILE='makefile_atosecs'
ECUID='<ecuid>'
ECGID='<ecgid>'
GATEWAY='<gatewayname>'
DESTINATION='<username>@genericSftp'
INSTALLDIR='<install_dir>'
```

```
JOB_TEMPLATE=''
CONTROLFILE='CONTROL_EA5'
#...

$
```

The parameters ECUID and ECGID have to be replaced by your username of the MARS server account and your group. To find your username and group name, the `ls -l` command can be used:

```
$ ls -l
-rwxr-xr--  1 as2 fr 34902 20 déc.   2022 file1.txt
drwxr-x---  4 as2 fr  4096  5 oct.  14:02 file2.txt
drwxr-x--- 14 as2 fr  4096 10 oct.  07:21 file3.txt
               ||  ||
             ECUID ECGID
```

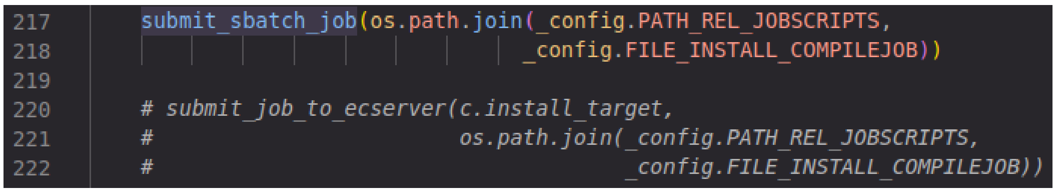
The parameter INSTALLDIR has to be replaced by the directory path where you want to install the flex_extract; the tool will then be installed in ${INSTALLDIR}/flex_extract_[version_number]. The GATEWAY and DESTINATION variables can be set up by the user for a transfer to a local gateway server (read more about this configuration in the documentation [4]). For a default, they must be set as it is with '<gatewayname>' and '<username>@genericSftp'.

Then, in order for the script to be able to take into account the INSTALLDIR parameter, the following lines (highlighted in yellow) must be added at the end of the setup_bologna.sh (around lines 90-100):

```
#!/bin/bash
#...
#...
TARGET='ecs
...
...
if [ -n "$CONTROLFILE" ]; then
  parameterlist+=" --controlfile=$CONTROLFILE"
fi
if [ -n "$INSTALLDIR" ]; then
  parameterlist+=" --installdir=$INSTALLDIR"
fi


# -----------------------------------------------------------------
# CALL INSTALLATION SCRIPT WITH DETERMINED COMMANDLINE ARGUMENTS
```

Lastly, some modifications have to be made to the `install.py` script of the `flex_extract` git repository. The file is located in `flex_extract/Source/Python/install.py`. In the function `install_via_gateway` the call of the `submit_job_to_ecserver` has to be commented, and the function `submit_sbatch_job` should be used instead (Fig. 6).



```
    submit_sbatch_job(os.path.join(_config.PATH_REL_JOBSCRIPTS,
                                   _config.FILE_INSTALL_COMPILEJOB))

    # submit_job_to_ecserver(c.install_target,
    #                       os.path.join(_config.PATH_REL_JOBSCRIPTS,
    #                                    _config.FILE_INSTALL_COMPILEJOB))
```

Figure 6 : modification to be made in the install.py script

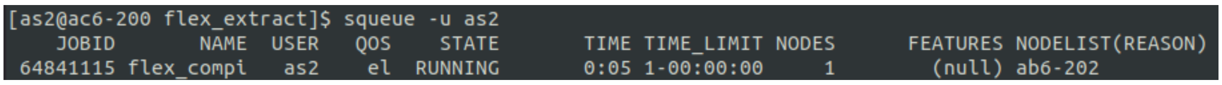
After the `setup_bologna.sh` script has been updated, last step is to execute it :

```
$
$ ./setup_bologna.sh

WARNING: Parameters GATEWAY and DESTINATION were not properly set for working on ECMWF server.
There will be no transfer of output files to the local gateway server possible!
Create tarball ...
SUBMITTED SBATCH JOB  Run/Jobscripts/compilejob.sh
Job compilation script has been submitted to ecgate for installation in /home/as2/FLEX_EXTRACT/flex_extract_v7.1.3
You should get an email with subject "flexcompile" within the next few minutes!
SUCCESS: INSTALLATION FINISHED!

$
```

If everything worked fine and modifications were made correctly, it should be possible to consult the installation job vie the `squeue -u username` command:



```
[as2@ac6-200 flex_extract]$ squeue -u as2
    JOBID       NAME  USER    QOS    STATE       TIME TIME_LIMIT NODES      FEATURES NODELIST(REASON)
 64841115 flex_compi   as2     el  RUNNING       0:05 1-00:00:00     1        (null) ab6-202
```

Once the job is complete, the `flex_extract` should be installed in the `INSTALLDIR` requested by you in the `setup_bologna.sh` script, and the job output should be found in `/scratch/[your_username]/flex_compile.xxxxxxxx.out`. If the job output seems correct, it is possible to test the installation with following instructions:

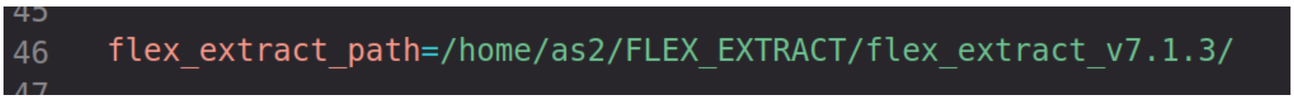
```
$ cd directory_where_flex_extract_was_installed
$ cd Testing/Installation/Calc_etadot
$ ../../../Source/Fortran/calc_etadot
STATISTICS:          98842.4598  98709.7359   5120.5385
STOP SUCCESSFULLY FINISHED calc_etadot: CONGRATULATIONS
```

If everything worked fine, the `flex_extract` was successfully installed!

Last step, is to copy the `script run_bologna.sh` from the git repository, to your `flex_extract` installation folder :

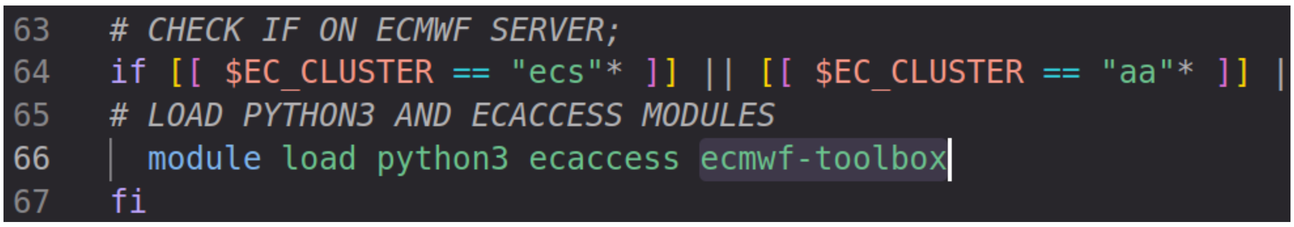
```
$ cp flex_extract/Run/run_bologna.sh [flex_install_dir]/flex_extract_[version_number]/Run/
```

Verify if in the copy of the script the root of the `flex_extract` is correct; then, correct it if it is not (Fig. 7). Also, always in the same script, it is necessary to add the `ecmwf-toolbox` in the `module load` command (Fig. 8); after that the `flex_etxract` is ready for use.



```
flex_extract_path=/home/as2/FLEX_EXTRACT/flex_extract_v7.1.3/
```

Figure 7 : flex_extract_path variable to check



```
# CHECK IF ON ECMWF SERVER;
if [[ $EC_CLUSTER == "ecs"* ]] || [[ $EC_CLUSTER == "aa"* ]] |
# LOAD PYTHON3 AND ECACCESS MODULES
  module load python3 ecaccess ecmwf-toolbox
fi
```

Figure 8 : ecmwf-toolbox to add in module load command

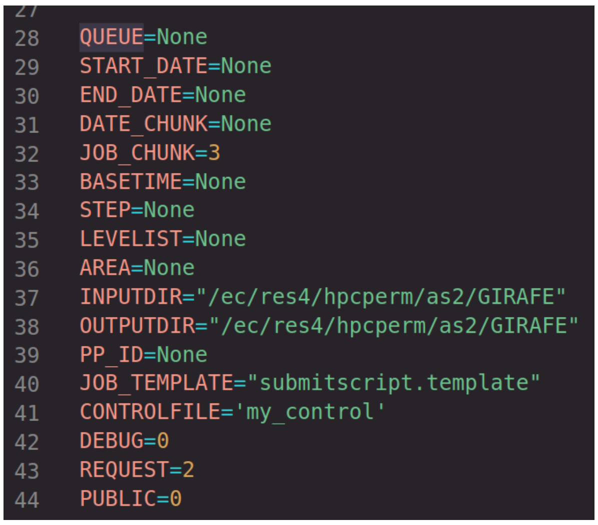
2.1.3.2 flex_extract data extraction

To extract the meteorological data, the script `run_bologna.sh` should be used. The configuration of the data to be extracted is done either by modifying the parameters directly in the script (Fig. 9), or by using the `CONTROL` file (Fig. 10) which allows a more precise and complex configuration of the parameters.

For a simple analysis and/or forecast data, the script parameters can be set as shown on Fig. 9 and 10. The variables `INPUTDIR`, `OUTPUTDIR` and `CONTROLFILE` should still be adapted by the user. `INPUTDIR` and `OUTPUTDIR` should be set with paths where the data and eventual intermediate files will be stored The variable `CONTROLFILE` should follow this rule :

- if `CONTROLFILE` variable contains only the filename of the `CONTROL` file, then the file must be placed in the `[flex_extract_root]/Run/Control` directory
- if `CONTROLFILE` variable contains the full absolute path to the `CONTROL` file, then the file can be placed anywhere

The content of the CONTROL file can be as shown on Fig. 10.



```
QUEUE=None
START_DATE=None
END_DATE=None
DATE_CHUNK=None
JOB_CHUNK=3
BASETIME=None
STEP=None
LEVELIST=None
AREA=None
INPUTDIR="/ec/res4/hpcperm/as2/GIRAFE"
OUTPUTDIR="/ec/res4/hpcperm/as2/GIRAFE"
PP_ID=None
JOB_TEMPLATE="submitscript.template"
CONTROLFILE='my_control'
DEBUG=0
REQUEST=2
PUBLIC=0
```

Figure 9 : run_bologna.sh parameters set up in the script itself for a GIRAFE data extraction

```
START_DATE 20240226
DTIME 3
TYPE AN FC AN FC AN FC AN FC
TIME 00 00 06 00 12 12 18 12
STEP 00 03 00 09 00 03 00 09
ACCTYPE FC
ACCTIME 00/12
ACCMAXSTEP 12
CLASS OD
STREAM OPER
GRID 1.
UPPER 90.
LOWER -90.
LEFT  -180.
RIGHT 179.
LEVELIST 1/to/137
RESOL 255
ETA 1
FORMAT GRIB2
DEBUG 0
REQUEST 0
PREFIX EN
```

Figure 10 : CONTROL file that can be used for the GIRAFE data extraction

In the `CONTROL` file, the parameters are as follows:

- `START_DATE` (and `END_DATE`) are set for a selection of time period for extraction
- `DTIME`, `TYPE`, `TIME` and `STEP` are set for the temporal resolution and data type (`AN` for analysis, or `FC` for forecast) of the extraction; here we extract `AN` data on the respective hour values indicated in `TIME` below `AN`, and `FC` data on hour values indicated in `STEP` below, with forecast basetime being indicated in `TIME`; `DTIME`

should correspond to the hour timestep of the extraction, here we request the data every 3 hour so `DTIME` is 3

- `ACCTYPE`, `ACCTIME` and `ACCMAXSTEP` correspond to the type, forecast times and maximum forecast steps of the flux forecast fields
- `CLASS` and `STREAM` tell us that we want to extract operational deterministic data
- `GRID` is the spatial resolution of the extraction bounding box
- `UPPER`, `LEFT`, `LOWER`, `RIGHT` are coordinates of the limits of the extraction area
- `LEVELIST` is the list of the models levels to extract
- `RESOL` is the horizontal resolution of spectral grid
- `ETA 1` means that horizontal wind fields etadot are retrieved on a regular lat-lon grid from ECMWF for a time-saving and computational ease
- `FORMAT` is the format of the data to extract
- `DEBUG 0` means that temporary files are not saved
- `REQUEST 0` means that a file with MARS requests is not created
- `PREFIX` is the prefix which precedes the YYMMDDHH datetime in the filename

Then the script can be launched via `srun` or `sbatch` command, in order to fasten up the process in comparison with the launch on the frontal nodes. Current version of the ACTRIS processing chain uses the ECMWF data already present on the NUWA server, but `flex_extract` tool can always be used to extract missing/new data or to update the processing chain with its own data extraction process.

2.1.4 FLEXPART output

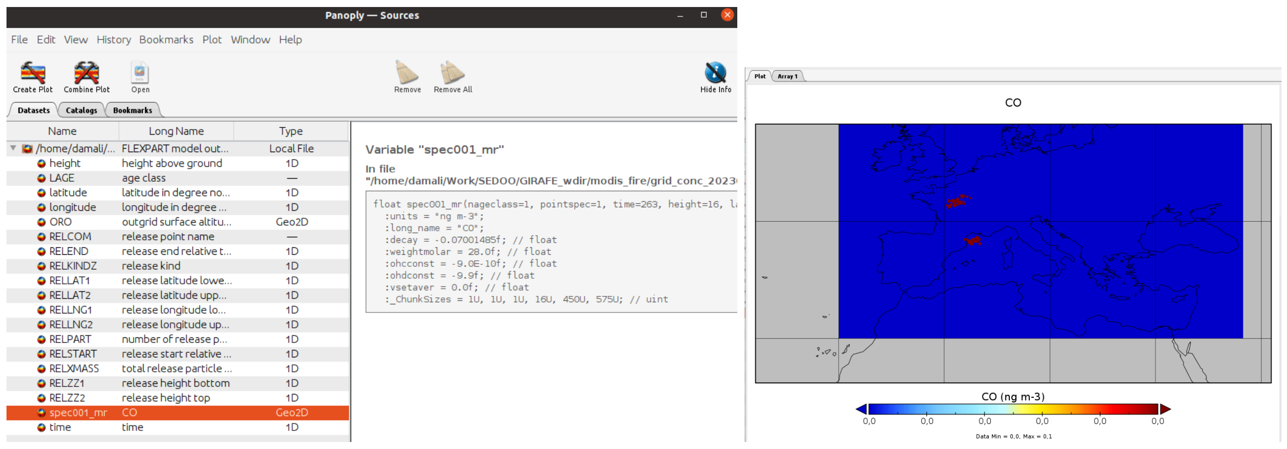
At each output time, FLEXPART generates files containing gridded output, with separate files produced for each species. The file-naming convention follows the pattern `grid_[type]_[date]_[nnn]`. For backward runs, type can be `time` for the sensitivity of receptor concentrations to emission fluxes, `drydep` for the sensitivity of receptor dry deposition to emissions or `wetdep` for the sensitivity of receptor wet deposition to emissions. For backward runs, there can also be an output file `grid_initial_nnn`, which gives the receptor sensitivity to initial conditions; `date` denotes the date and time for which the output is valid, and `nnn` is the species number as specified in RELEASES. The list of the output times is progressively written to the text file dates.

The physical unit used for the output data in the files `grid_conc_date_nnn` and `grid_time_date_nnn` depends on the settings of the switches `ind_source` and `ind_receptor` (Table 2). It's important to note that the unit of mass mixing ratio can also be utilized in `grid_conc_date_nnn`. Source–receptor relationships (i.e., emission sensitivities) in backward mode for atmospheric receptors are written out in `grid_time_date_nnn` files; those for deposited mass are recorded in files `grid_wetdep_date_nnn` and `grid_drydep_date_nnn`. Additionally, it's noteworthy that all gridded output quantities in FLEXPART represent grid cell averages, not point values.

Table 2 : Physical units of input/output data in forward runs for various settings of ind_source, ind_receptor

File name	`ind_source`	`ind_receptor`	Input unit	Output unit
`grid_time*`	1	1	1	s
`grid_time*`	1	2	1	s.m3.kg-1
`grid_time*`	2	1	1	s.kg.m-3
`grid_time*`	2	2	1	s

By default, FLEXPART output is written in the native binary format. However, FLEXPART v10.4 can also support output in NetCDF format if the NetCDF libraries are installed. To activate NetCDF support, the option `ncf=yes` must be appended to the compilation `make` command. Only one NetCDF file is written; this file contains all species and all time steps. Since the NetCDF output is specified in the climate and forecast (CF) format, any standard software can be used for displaying and processing the output (Fig. 11). NetCDF output data files are compressed. The NetCDF output file contains information on the run settings and the simulation grid from the `COMMAND` and `OUTGRID` files.



Name	Long Name	Type
/home/damali/...	FLEXPART model out...	Local File
height	height above ground	1D
LAGE	age class	—
latitude	latitude in degree no...	1D
longitude	longitude in degree ...	1D
ORO	outgrid surface altitu...	Geo2D
RELCOM	release point name	—
RELEND	release end relative t...	1D
RELKINDZ	release kind	1D
RELLAT1	release latitude lowe...	1D
RELLAT2	release latitude upp...	1D
RELLNG1	release longitude lo...	1D
RELLNG2	release longitude up...	1D
RELPART	number of release p...	1D
RELSTART	release start relative ...	1D
RELXMASS	total release particle ...	1D
RELZZ1	release height bottom	1D
RELZZ2	release height top	1D
spec001_mr	CO	Geo2D
time	time	1D

Variable "spec001_mr"

In file
"/home/damali/Work/SEDOO/GIRAFE_wdir/modis_fire/grid_conc_2023

```
float spec001_mr(nageclass=1, pointspec=1, time=263, height=16, la
  :units = "ng m-3";
  :long_name = "CO";
  :decay = -0.07001485f; // float
  :weightmolar = 28.0f; // float
  :ohcconst = -9.0E-10f; // float
  :ohdconst = -9.9f; // float
  :vsetaver = 0.0f; // float
  :_ChunkSizes = 1U, 1U, 1U, 16U, 450U, 575U; // uint
```

Figure 11 : FLEXPART output in netCDF format, opened with Panoply

2.2 SOFT-io tool

SOFT-IO tool is a Python library and an implementation of the methodology of source attribution using FLEXPART and carbon monoxide emission inventories. This tool allows the estimation of the geographical origin of pollutants and emission sources which are at origin of CO increases in the troposphere and lower stratosphere.

.

The tool takes as input a FLEXPART backward trajectories output to analyze it and compare it to the emission sources. The current version of the SOFT-IO tool works with CO emission inventories CEDS (A Community Emissions Data System for Historical Emissions, anthropogenic emissions) [5] and GFAS (CAMS Global Fire Assimilation System) [6].

The output of the SOFT-IO tool is a dataset with estimated contributions for different geographical regions and for two of the mentioned inventories. As shown on Fig. 12, for a FLEXPART simulation for SIRTA station on a certain day, the anthropogenic contributions are between 0.001 and 10 ppb CO for CEDS inventory and between 0.01 and 0.1 ppb CO for GFAS inventory. As for the geographical regions, it seems to come mostly from regions EURO (Europe), BONA (Boreal North America), TENA (Temperate North America) and BOAS (Boreal Asia) (Fig. 13).

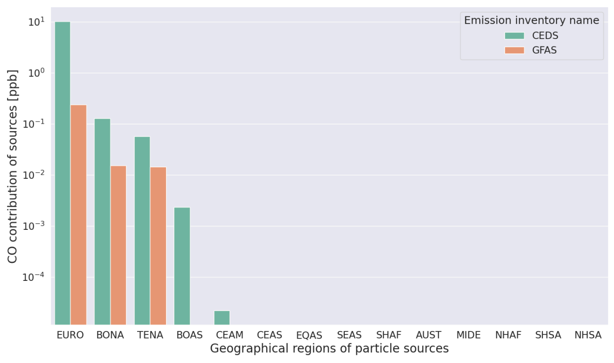


Figure 12 : Estimated contribution of CO from the different geographical regions to the measurement of the SIRTA station on 04/17/2024, calculated by SOFT-IO from the results of FLEXPART.

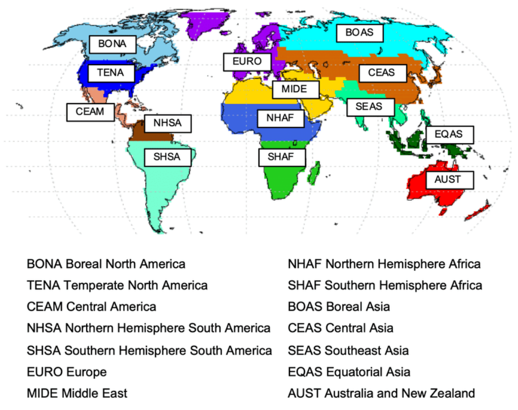

BONA
TENA
CEAM
NHSA
SHSA
EURO
MIDE
NHAF
SHAF
BOAS
CEAS
SEAS
EQAS
AUST
BONA Boreal North America
TENA Temperate North America
CEAM Central America
NHSA Northern Hemisphere South America
SHSA Southern Hemisphere South America
EURO Europe
MIDE Middle East
NHAF Northern Hemisphere Africa
SHAF Southern Hemisphere Africa
BOAS Boreal Asia
CEAS Central Asia
SEAS Southeast Asia
EQAS Equatorial Asia
AUST Australia and New Zealand

Figure 13 : GFAD map of the geographic regions used in the SOFT-IO [7]

It is possible to concatenate results from FLEXPART and SOFT-IO in order to create a time series of CO contributions.

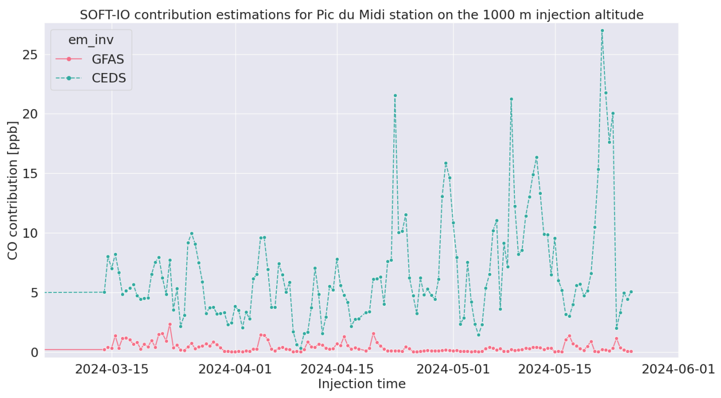


Figure 14 : Time series of the SOFT-IO estimation for Pic du Midi station on 1000m injection altitude

3. ACTRIS simulation for user

The two main steps of the ACTRIS processing chain are FLEXPART simulation for the backward trajectories calculation and the application of the SOFT-IO algorithm on the FLEXPART result. User interfaces for these steps and the tools installation are detailed below in this section.

On top of the FLEXPART executable and configuration, an overlay of Python and Bash scripts were created to facilitate user experience with the tool and chain production integration. The Python script represents the user interface for the FLEXPART tool and allows to configure the simulation through one simple configuration xml file, without changing manually every FLEXPART input file. The Bash script can then be used over the Python script in order to automate the process and handle multiple simulations and steps (FLEXPART and SOFT-IO) one by one and for multiple stations. These scripts will be detailed later in this document.

3.1 ACTRIS Singularity container

FLEXPART and SOFT-IO tools, as well as their dependencies, are containerized into a Singularity container, which allows any user to run it without previous library installation (except the installation of the Singularity application itself).

The file `actris-container.def` is a Singularity Definition File. A Singularity Definition File is a set of blueprints explaining how to build a custom container. It includes specifics about the base OS to build or the base container to start from, software to install, environment variables to set at runtime, files to add from the host system, and container metadata. The command which allows the build of a `.sif` Singularity image is:

```
$ sudo singularity build actris-container.sif actris-container.def
```

⚠️ One must have sudo rights on the system where the container is built. If it is not possible, the option `--fakeroot` can be used in order to build a container without sudo rights. In case of a multi-user server or machine, if the `--fakeroot` option throws an error, the one should contact the administrator of the machine to add one's user to white list. Once the container is built, the image can be run anywhere and without sudo rights.

3.2 FLEXPART Python interface

As it was presented in the previous section, FLEXPART tool requires a certain number of input configuration files to be set by the user. It can be tedious to edit all these files manually and go through them one by one, especially in the case of a production chain where things must be performed smoothly and in an automatic way. Thus, firstly, to overcome this disadvantage and start preparing the processing chain, a Python interface for FLEXPART was created in order to facilitate the use of the model.

The interface is a Python script `actris.py` which requires one configuration file from the user as input, `actris-config.xml`. The script then writes FLEXPART input files based on the parameters set up by the user in the XML file and launches the simulation itself. This Python script must be executed inside of the ACTRIS Singularity container as it relies on the Python libraries and FLEXPART tool itself.

Below is the `--help` message of the Python script which shows the usage of the script :

```
Singularity> python ./actris.py -h
usage: actris.py [-h] [--config CONFIG]

Python code that prepares all FLEXPART inputs and launches a backward FLEXPART
simulation for an ACTRIS station based on the user’s configuration file.

options:
  -h, --help       show this help message and exit
  --config CONFIG  Filepath to your configuration xml file.
Singularity>
```

The configuration file is structured as follows :

```
<config>
  <actris>

    <simulation_start>
      <date>20230501</date>
      <time>000000</time>
    </simulation_start>
    <simulation_end>
      <date>20230510</date>
      <time>233000</time>
```

simulation start and end date and time, knowing that the simulation will be ran backward from the end_date to the start_date

```
</simulation_end>

<ecmwf_time>
   <dtime>3</dtime>
</ecmwf_time>
```

time between two ECMWF fields in hours (ECMWF fields provided by the user)

```
<flexpart>
   <root>/usr/local/flexpart_v10.4_3d7eebf/</root>
```

root node is not to change, it is the FLEXPART installation path inside the Singularity container

```
   <par_mod_parameters>
      <nxmax> 361 </nxmax>
      <nymax> 181 </nymax>
```

number of point of meteorological fields in x and y direction

```
      <nuvzmax> 138 </nuvzmax>
      <nwzmax> 138 </nwzmax>
```

maximum dimension of (u,v) and (w) wind fields in z direction (for fields on eta levels)

```
      <nzmax> 138 </nzmax>
   </par_mod_parameters>
```

maximum dimension of wind fields in z direction for the transformed Cartesian coordinates

```
   <out_grid>
      <longitude>
         <min> -15 </min>
         <max> 42.5 </max>
      </longitude>
      <latitude>
         <min> 75 </min>
         <max> 30 </max>
      </latitude>
      <resolution> 1 </resolution>
```

the latitude and longitude boundary of the region for which the simulation will be executed (f.ex. -180/180 and -90/90 is for global simulation) and the spatial resolution of the output

```
      <height>
         <level> 100 </level>
         <level> 200 </level>
         <level> 500 </level>
      </height>
   </out_grid>
```

altitude levels on which simulation will be performed (upper boundaries of each desired level)

```
        <command>
           <forward>-1</forward>
           <time>
              <output> 3600 </output>
           </time>
           <iOut> 9 </iOut>
        </command>
```

forward flag set to -1 indicates that the simulation is to be performed backward; node time/output is the output timestep in seconds (f.ex. 3600 means that the particle concentrations will be estimated every hour between simulation start and end date/time) and iOut is output type/format (option 9 is mandatory for the ACTRIS processing chain, it produces netCDF output)

```
        <releases>
           <species> 24 </species>
```

the node release concern the particle injections (emitted by the human activities or f.ex. forest fires); here we use the number 24 to indicate the AIR-TRACER species which was configured specifically for the backward simulations)

```
           <release>
              <start_date>20230501</start_date>
              <start_time>230000</start_time>
              <duration>00000000</duration>
              <altitude_min>10</altitude_min>
              <altitude_max>100</altitude_max>
```

each release is characterized by the start date YYYYMMDD, time HHMMSS and its duration DDHHMMSS; the release date/time(s) indicate the injection date/time(s) between indicated minimum and maximum altitudes (meters above ground); the emitted mass will be unitary with 100 000 particles injected

```
              <zones>
                 <zone name=”Landes”>
                    <latmin>43.799231645</latmin>
                    <latmax>44.411882453</latmax>
                    <lonmin>-1.267829619</lonmin>
                    <lonmax>-0.378441805</lonmax>
                 </zone>
              </zones>
           </release>
        </releases>
     </flexpart>
```

for each date/time release it is possible to define multiple release ROI(s) with corresponding names and latitude/longitude boundaries

```
        <paths>
            <working_dir>/home/actris/work</working_dir>
            <ecmwf_dir>/home/actris/data</ecmwf_dir>
        </paths>
    </girafe>
</config>
```

paths node is where must be indicated paths to the :

- *working directory (simulation run-time files, executables etc)*
- *directory with ECMWF files*

The `actris.py` script does not perform any post-processing after the simulation is done. The only output is the FLEXPART native netCDF output file. However, the file is used later by other processing steps of the ACTRIS chain.

3.3 SOFT-IO user interface

The Python FLEXPART user interface allows managing only one of the processing steps, FLEXPART. To continue preparing the processing chain for the production and integration, the second step, SOFT-IO tool, was studied.

As the SOFT-IO tool is a Python library itself, it was decided to create a Python script `apply-softio.py` in order to manage this processing step. This script allows to apply SOFT-IO to a FLEXPART output of the ACTRIS use case and to save results as a standalone output file. If requested by the user, the results can also be added to a common database where CO contribution estimations are concatenated into a single variable with multiple dimensions (geographical region, emission inventory type etc). In our case, this common database solution is used for the time-series creation and for the latter visualization.

The `apply-softio.py` Python interface works as follows:

```
Singularity> /usr/local/micromamba/envs/softio_env/bin/python3 ./apply-softio.py -h
usage: apply-softio.py [-h] [-f FILE] [-n NAME] [-d DIR] [-o OUTPUT]

Applying SOFT-IO to the FLEXPART output for the ACTRIS stations

options:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  Path to the FLEXPART output netCDF file
  -n NAME, --name NAME  Short name of the ACTRIS station in question
  -d DIR, --dir DIR     Path to the directory where to store the output SOFT-IO file
  -o OUTPUT, --output OUTPUT
                        Path to the SOFT-IO merged database file where to add new
data
Singularity>
```

The argument `-n/--name` is used to identify results for the same station in the common database and also to keep track of this information in the standalone output. The output netCDF from the SOFT-IO has the following structure :

```
<xarray.Dataset>
Dimensions:          (release_time: 1, geo_region: 15, em_inv: 2, height: 1,
                      station_id: 1)
Coordinates:
  * release_time     (release_time) datetime64[ns] 2021-12-31T23:00:01
    release_lon      float32 -5.033
    release_lat      float32 6.217
    release_pressure float32 1.25e+04
    release_npart    int32 100000
  * geo_region       (geo_region) object 'TOTAL' 'BONA' ... 'EQAS' 'AUST'
  * height           (height) float32 100.0
  * station_id       (station_id) object 'LTO'
  * em_inv           (em_inv) object 'GFAS' 'CEDS'
Data variables:
    CO_contrib       (em_inv, height, station_id, release_time, geo_region) float64
...
```

The `CO_contrib` variable contains values of the CO contribution estimated by the SOFT-IO tool. The dimensions of this variable are (`em_inv`, `height`, `station_id`, `release_time`, `geo_region`). Even if some of these dimensions have only one possible value in the standalone output, it allows easier management of multiple SOFT-IO outputs when merging them in the common database. The data from this file alone allows to obtain Fig. 15 with footprints on a given day for a given station/height combination.

3.3 Creation of footprints images

The last step of the processing is the creation of the summary footprint images for each of the FLEXPART outputs. These images are then intended to be used in the visualization tool for an easier analysis and easier representation of rich information given by the initial FLEXPART output.

The Python code that creates these images were based on the image creation used for the IAGOS Footprints portal [8]. The calculations allow to integrate FLEXPART result on the vertical levels and also reproject it onto a special latitude/longitude grid that can be overlayed with a global map for a visual representation (Fig. 15).

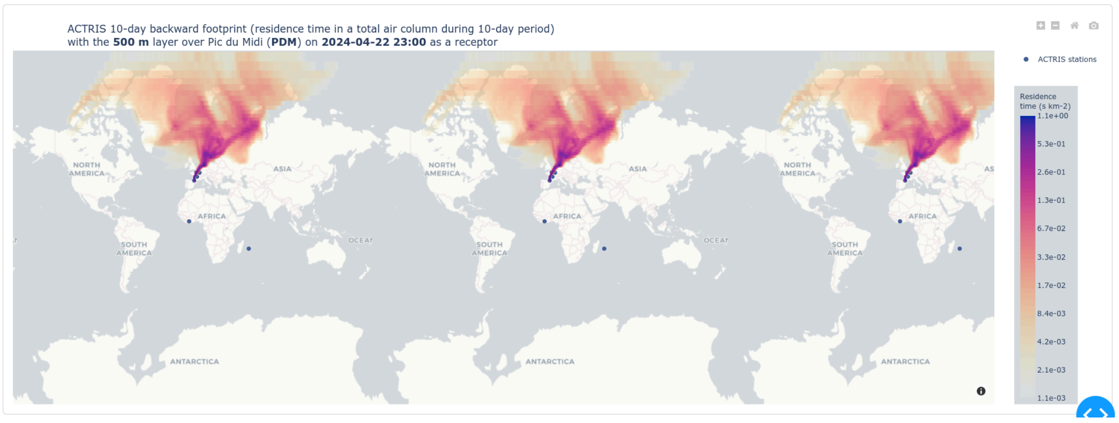

ACTRIS 10-day backward footprint (residence time in a total air column during 10-day period)
with the **500 m** layer over Pic du Midi (**PDM**) on **2024-04-22 23:00** as a receptor
ACTRIS stations
Residence time (s km-2)
1.1e+00
5.3e-01
2.6e-01
1.3e-01
6.7e-02
3.3e-02
1.7e-02
8.4e-03
4.2e-03
2.1e-03
1.1e-03
NORTH AMERICA
SOUTH AMERICA
AFRICA
ASIA
ANTARCTICA

Figure 15 : FLEXPART footprint image overlays on a global map; the colorbar is the scale of the residence time of the particles estimated by FLEXPART

The output image data is saved into `.zarr` format via `xarray` Python library and can be opened as a Dataset in Python to access the data as in a simple netCDF file with variables, dimensions and attributes. Below is the example of such footprints database saved in a `.zarr` format and re-opened with xarray:

```
<bound method Dataset.load of <xarray.Dataset>
Dimensions:           (height: 5, latitude: 180, longitude: 360, time: 450,
                       station_id: 6)
Coordinates:
  * height            (height) float32 100.0 500.0 1e+03 1.5e+03 2e+03
  * latitude          (latitude) float32 -89.5 -88.5 -87.5 ... 87.5 88.5 89.5
  * longitude         (longitude) float32 -179.5 -178.5 -177.5 ... 178.5 179.5
  * station_id        (station_id) <U3 'LTO' 'OPE' 'PDM' 'PUY' 'RUN' 'SAC'
  * time              (time) datetime64[ns] 2021-12-31T23:00:00 ... 2024-05-2...
Data variables:
    res_time_per_km2  (time, station_id, height, latitude, longitude) float16 ...
```

The variable `res_time_per_km2` still represents the residence time calculated by FLEXPART, but here it is reprojected and integrated over the vertical column in order to get only one 2D matrix for a given simulation. ‘`height`’ is the release height of particles in the simulation, and ‘`time`’ is the release time of the particles in the simulation.

3.4 Bash User Interface

In order to incorporate all of the processing steps together and to facilitate the integration of the processing chain and a batch processing for the future, an additional Bash script was created. With a few configuration setups and input arguments, this script allows to easily manage all of the steps of the processing chain for a given ACTRIS station on a given simulation date.

The usage of the Bash script is as follows :

```
user@pc:~/git/actris-footprints> ./launch_simulation_cron.sh -h

###                 |    *
###                 |  *
###                 | *
###             ,,gg|dY""""Ybbgg,,
###        ,agd""'  |           `""bg,
###     ,gdP"     A C T R I S       "Ybg,
###                     FRANCE
###
### This script is intended for FLEXPART simulation, SOFT-io CO contribution computation and creation of the CO receptor-
### -source footprints for an ACTRIS station. Simulation will be performed in the backward mode to the J-10 days time limit. The
### user should just select the station, the date/time of the simulation, from which the J-10 period will be counted, and
### provide the configuration file where paths to working and output directories are indicated. Options --flexpart, --softio and
### --footprint allow to activate one or multiple processings.
###
### Usage : launch_simulation_cron.sh [options] arguments
###
### Options:
###   -h|--help        Show this help message and exit
###   --flexpart       Activate FLEXPART simulation
###   --softio         Activate SOFT-io computations
###   --footprint      Activate creation of the footprint image
###
### Arguments:
###   -n        station_name_code   ID code (short_name) of one of the stations from your configuration file
###   -d        YYYYMMDDHH          date and time of the simulation
###   -c,--conf configuration.file  configuration file with paths
```

The user’s inputs are :

- [`-n`] ACTRIS station code name/identifier from the configuration file (detailed later in this section)
- [`-d`] simulation start date and hour in `YYYYMMDDHH` format, the simulation is ran backward from this date
- [`-c`] configuration file itself (detailed later in this section)
- [`--flexpart, --softio, --footprint`] processing flags that can activate different processing steps (at least one if the flags must be set up)

The summary of the steps performed by this Bash script is presented on Fig. 16 below.

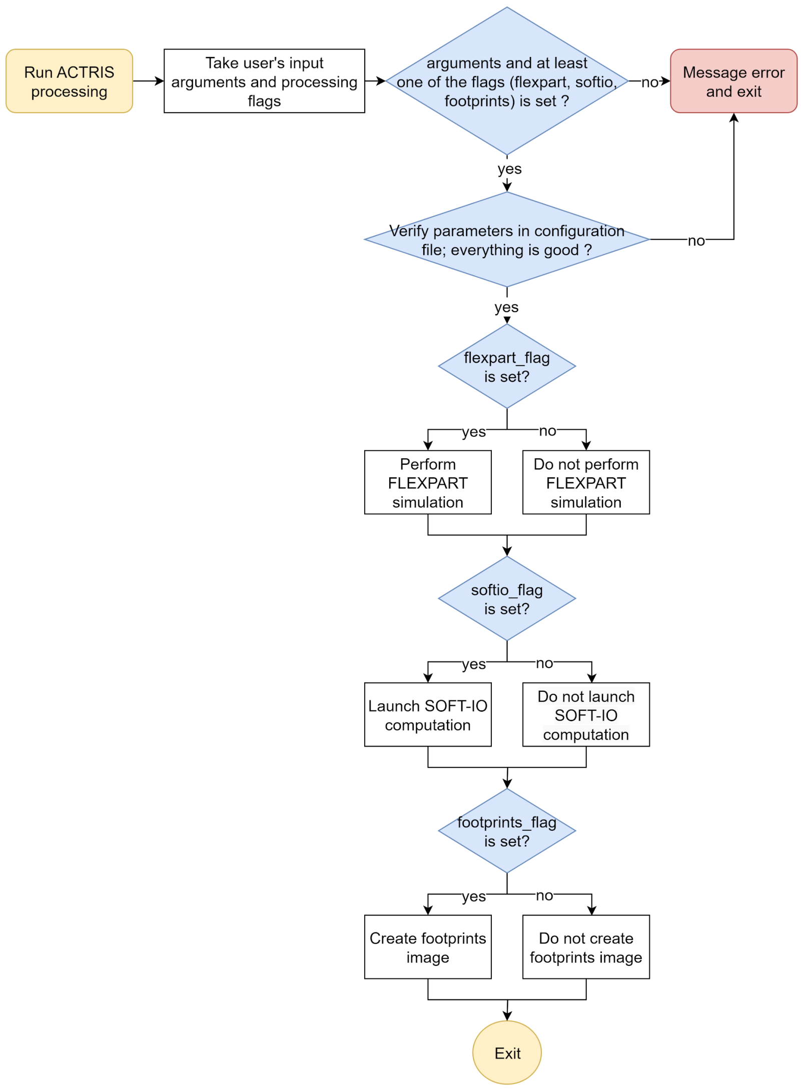

Run ACTRIS processing
Take user's input arguments and processing flags
arguments and at least one of the flags (flexpart, softio, footprints) is set ?
no
Message error and exit
yes
Verify parameters in configuration file; everything is good ?
no
yes
flexpart_flag is set?
yes
no
Perform FLEXPART simulation
Do not perform FLEXPART simulation
softio_flag is set?
yes
no
Launch SOFT-IO computation
Do not launch SOFT-IO computation
footprints_flag is set?
yes
no
Create footprints image
Do not create footprints image
Exit

Figure 16 : flowchart of the Bash script interface for the ACTRIS processing chain

The FLEXPART processing includes : getting information about chosen station, based on this information and user's data writing the configuration XML file for Python script, launch Python script with this configuration file through a Singularity container; if simulation was successful, renaming the output file and move it to the user request output directory.

The SOFT-IO processing consists of finding requested FLEXPART simulation and running SOFT-IO on this output.

The footprint creation step consists of creating a 2D footprint image from FLEXPART output and adding it to the .zarr database of the footprints.

3.4.1 User paths configuration

A path configuration is required by the Bash script in order to define where to find input data and where to store output data. This configuration file is a simple ASCII file that will be sourced in the Bash script. Its syntaxe is as follows:

```
SRC_DIR="/path/to/folder/with/python/source/codes"
DATA_DIR="/path/to/folder/with/ecmwf/grib/data"
IMAGE_FILEPATH="/path/to/the/singularity/image.sif"
ROOT_WDIR="/path/to/root/working/folder/for/subfolders"
FLEXPART_OUT_DIR="/path/to/folder/for/flexpart/output"
SOFTIO_OUT_DIR="/path/to/folder/for/softio/output"
SOFTIO_DATABASE="/path/to/output/softio/database.nc"
FOOTPRINTS_DATABASE="/path/to/output/footprints/images/database.zarr"
STATIONS_CONF="/path/to/file/with/stations/configuration.json"
```

`ROOT_WDIR` is the root working directory where the sub folders for each simulation will be created. `FLEXPART_OUT_DIR` and `SOFTIO_OUT_DIR` will host output files directly without subfolders. The working directory tree will then be as follows :

```
root_wdir/
 ├─PDM/
 │   ├─wdir-2022011012-500/
 │   ├─wdir-2022011012-1000/
 │   └─...
 └─LTO/
     ├─wdir-2022050512-100/
     ├─wdir-2022050512-500/
     ├─wdir-2022050512-1500/
     └─...
```

Structure of the root working directory

```
flexpart_out_dir/
 ├─PDM-2022011012-20211231-500.nc
 ├─PDM-2022011012-20211231-1000.nc
 ├─...
 ├─LTO-2022050512-20220425-100.nc
 ├─LTO-2022050512-20220425-500.nc
 └─LTO-2022050512-20220425-1500.nc

softio_out_dir/
 ├─softio-PDM-2022011012-20211231-500.nc
 ├─softio-PDM-2022011012-20211231-1000.nc
 ├─...
 ├─softio-LTO-2022050512-20220425-100.nc
 ├─softio-LTO-2022050512-20220425-500.nc
 └─softio-LTO-2022050512-20220425-1500.nc
```

Structures of the output directories for FLEXPART and SOFT-IO outputs respectively

`PDM` and `LTO` are short names of Pic Du Midi and Lamto station respectively that were passed to the script as `-n` argument; their working directories and FLEXPART/SOFT-IO output files are also named with these short names in order to identify them. The first datetime values in the output filenames and in working directories names indicate the simulation start date passed to the script with `-d` argument. The second datetime, which is only in the output filenames, indicates the “end date” of the simulation which corresponds to the J-10 days. This simulation duration to J-10 days is configured directly in the script. This duration was defined as the best compromise between uncertainty of the results and the diffusion scale. The last value in the names corresponds to the height where the particles were injected for the simulation. These values are configured in the stations configuration file which will be discussed in the section *3.4.2 Station JSON configuration*.

3.4.2 Stations JSON configuration

A JSON configuration file must be set up with general information about ACTRIS stations in question. The syntaxe of the file is as follows:

```
[
  {
    "short_name":"PUY",
    "long_name":"Puy de Dôme",
    "longitude":2.964886,
    "latitude":45.772223,
    "altitude":1465,
    "release_heights":"500 1500"
  },
  {
    "short_name":"SAC",
    "long_name":"SIRTA Atmospheric Research Obs",
    "longitude":2.1588888889,
    "latitude":48.7086111,
    "altitude":162,
    "release_heights":"500"
  },
…
]
```

“`short_name`” allows users to identify a station by its short and unique ID name that is easy to use when calling scripts for a particular station or to identify its data in a common database. This ID will also allow to identify the station in question throughout the entire processing chain between different processing steps, and this short name must be used when calling the Bash script for one of the stations. “`altitude`” is the station altitude in meters, and “`release_heights`” is the list of release heights for the FLEXPART simulations. Each value in this list will correspond to its own independent simulation. These values have to be defined depending on what phenomenon one seeks to observe and study (global diffusion, regional diffusion etc.). Annex 1 of this document presents the configuration of chosen ACTRIS stations that have been defined in the scope of this project by the scientist in charge of the study. This file can be changed if a new station is added to that list or if an already existing station has to be updated.

3.4.3 Processing flags

Processing flags `--flexpart`, `--softio` and `--footprint` allows to activate respective processing steps. At least one of the flags must be set up when calling the script. When calling `--softio` or `--footprint` steps, the script will search for already existing FLEXPART output files. If FLEXPART simulation fails or if FLEXPART output were not found for SOFT-IO computations and footprints visualization the script will throw an error/warning message and exit with exit status = 1.

3.5 ACTRIS batch production

For a continued production of the whole processing chain for all of the defined ACTRIS stations a production Bash script was created. This script can be launched in two different modes which will be detailed later in this section after a general presentation of the script. The `actris-production.sh` script's interface is as follows :

```
###                 |    *
###                 |  *
###                 | *
###             ,,gg|dY""""Ybbgg,,
###        ,agd""'  |           `""bg,
###     ,gdP"     A C T R I S       "Ybg,
###                     FRANCE
###
### This script allows to handle multiple ACTRIS simulations in the produciton mode. Processing steps are called via Bash
### user interface with --flexpart, --softio or --footprint flags. This script can be called via cron tool as well.
###
### Usage :  [options] arguments
###
### Options:
###   -h|--help        Show this help message and exit
###
### Arguments:
###   -d        list_of_dates.txt    list of the date/hours to process
###   -c,--conf configuration.file  configuration file
###
### Configuration file is an ASCII file containing paths to source codes and workingd/data directories. The syntaxe is as follows :
###    SERVER_USER="username"
###    LOGS_DIR="/path/to/the/directory/where/to/save/log/files"
###    PATHS_CONF_FILEPATH="/path/to/the/source/configuration.conf"
###    LOG_CATALOGUE_FILEPATH="/path/to/the/common/log/database.txt"
###    FLEXPART_HOUR="00"
###    DELAY_N_DAYS=5
###
### --> PATHS_CONF_FILEPATH is the configuration file used by the actris-processing.sh. In this files parameters such as
### SRC_DIR, SINGULARITY_FILEPATH and others are set up by the user.
### --> FLEXPART_HOUR is the start hour of the simulation. For the ACTRIS project it was defined that the simulation will be launched
### twice a day, at 00h and 12h.
### --> DELAY_N_DAYS is the number of days to delay the simulation start from the 'today' date. This can number can be 0,
### meaning that the simulation date will be the same as the processing date, but dû to the ECMWF data availability it is not
### always possible. Thus this parameter should be adapted based on the delay between 'today' date and the data availability
### of the user.
```

```
###
### List of the dates is a simple ASCII file where each line is a simulation date/hour to process. Dates should be in the
### format YYYYMMDDHH. This list is not mandatory (f.ex. in a case of a cron production this file is not required), but if it
### is provided it will overwrite parameters FLEXPART_HOUR and DELAY_N_DAYS set up in the configuration file. This list of dates
### can be used to process a particular time period or to re-process particular dates that were not processed or threw an error
### during the automatic production.
```

Figure 17 shows the steps of this Bash script which can be used for an automatic continued production via tools such as cron that allows cyclic / scheduled script launches. All of the processing steps (FLEXPART, SOFT-IO and footprints images creation) are done via calling the Bash user interface presented in the section *2.2.4*.

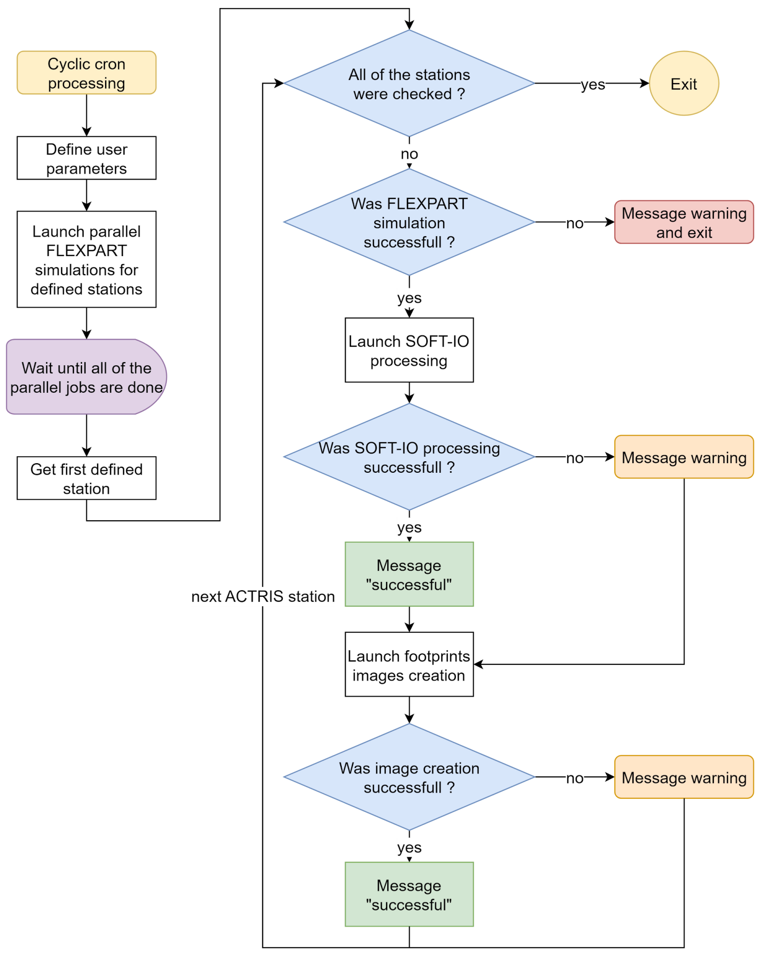

Cyclic cron processing
Define user parameters
Launch parallel FLEXPART simulations for defined stations
Wait until all of the parallel jobs are done
Get first defined station
All of the stations were checked ?
yes
Exit
no
Was FLEXPART simulation successfull ?
no
Message warning and exit
yes
Launch SOFT-IO processing
Was SOFT-IO processing successfull ?
no
Message warning
yes
Message "successful"
next ACTRIS station
Launch footprints images creation
Was image creation successfull ?
no
Message warning
yes
Message "successful"

Figure 17 : flowchart of the Bash script for an automatic ACTRIS production

All of the processing steps are launched via SLURM tool in order to use the computing resources of the server. FLEXPART jobs are launched in parallel in order to gain time. SOFT-IO and footprints visualization are launched sequentially for each of the stations because of the concurrent input/output access to their respective output databases. Since these two processing steps do not take as much time as FLEXPART simulations, it is not binding to perform these steps sequentially.

Error, warning and "successful" messages are traced in a log file defined by the user, which allows to keep track of production and processing. If FLEXPART simulation was not successful, the warning message is written for all of the three processing steps since SOFT-IO and footprints visualization require FLEXPART output and it was not created dû to the failed FLEXPART simulation.

Each of the processing steps is logged into an output log file defined by the user. For continued production, this log file should be the same, as it will allow to gather all of the important information about failed steps in one place. The log messages have the following syntax :

```
{'processing_date':'29/04/2024 13:04:56 CEST', 'station_id':'PUY',
'simulation_date':'2024042412',
'log_filepath':'/sedoo/resos/actris/LOGS_CRON/PDM/PDM-2024042412.out',
'processing_step':'flexpart', 'status':0}
{'processing_date':'29/04/2024 13:05:44 CEST', 'station_id':'PUY',
'simulation_date':'2024042412',
'log_filepath':'/sedoo/resos/actris/LOGS_CRON/PUY/PUY-2024042412.out',
'processing_step':'softio', 'status':0}
{'processing_date':'29/04/2024 13:06:40 CEST', 'station_id':'PUY',
'simulation_date':'2024042412',
'log_filepath':'/sedoo/resos/actris/LOGS_CRON/PUY/PUY-2024042412.out',
'processing_step':'footprints', 'status':0}
{'processing_date':'29/04/2024 13:10:20 CEST', 'station_id':'LTO',
'simulation_date':'2024042412',
'log_filepath':'/sedoo/resos/actris/LOGS_CRON/LTO/LTO-2024042412.out',
'processing_step':'softio', 'status':0}
{'processing_date':'29/04/2024 13:11:40 CEST', 'station_id':'LTO',
'simulation_date':'2024042412',
'log_filepath':'/sedoo/resos/actris/LOGS_CRON/LTO/LTO-2024042412.out',
'processing_step':'footprints', 'status':0}
```

‘processing_date’ is the date when the processing was launch; ‘station_id’ is the short name of the station in question; ‘simulation_date’ is the simulation start date in the simulation tool; ‘log_filepath’ is the log file with all of the processing messages for this station/date combination (messages from FLEXPART, SOFT-IO etc); ‘processing_step’ correspond to either one of the steps (FLEXPART, SOFT-IO or footprints images creation); ‘status’ is the status of the corresponding processing step (0 means that the processing was successful; 1 means that there were an error). The key ‘status’ can be used to quickly check if there are any errors in the production process (via Bash, Python or other parsing/reading tools). The key ‘log_filepath’ allows users to quickly access respective log files with more details about processing steps.

The configuration file (listed below) passed to this script is a more general configuration than the one used in the Bash interface of section 3.4. In this configuration, `PATHS_CONF_FILEPATH` is the configuration file used by the `actris-processing.sh`. In this file parameters such as `SRC_DIR`, `SINGULARITY_FILEPATH` and others are set up by the user. `FLEXPART_HOUR` is the start hour of the simulation. For the ACTRIS project it was defined that the simulation will be launched twice a day, at 00h and 12h. `LOGS_DIR` is a directory where simulation log files with details about processing steps will be stored. `DELAY_N_DAYS` is the number of days to delay the simulation start from the 'today' date. This number can be 0, meaning that the simulation date will be the same as the processing date, but dû to the ECMWF data availability it is not always possible. Thus this parameter should be adapted based on the delay between 'today' date and the data availability of the user.

```
SERVER_USER="username"
LOGS_DIR="/root/actris/log_files"
PATHS_CONF_FILEPATH="/root/actris/actris.conf"
LOG_CATALOGUE_FILEPATH="/root/actris/production.log"
FLEXPART_HOUR="00"
DELAY_N_DAYS=5
```

Configuration file content for the cron script

The second argument `-d` is an optional argument and it is a list of dates to process. This mode can be useful when there is a need to process particular dates or to reprocess dates that have thrown error in the production mode processing. The behaviour of the script will depend on whether this list is provided or not:

- no list is provided : the simulation date and hour are calculated from the 'today' date, `DELAY_N_DAYS` and `FLEXPART_HOUR` parameters; this date is the only simulation date for this script launch
- a list of dates/hours is provided (example listed below) : the parameters DELAY_N_DAYS and FLEXPART_HOUR are not taken into account, the script loops through the dates provided in the list and launches simulations for these dates one by one. This mode can be used to process a particular time period or to reprocess dates that were not processed in the automatic production mode dû to the circumstances.

```
2022050100
2022050112
2022050312
2022050412
```

Example of dates to process that can be passed as a list with -d option

Bibliography

[1] ACTRIS - Climat et qualité de l'air

https://www.actris.fr/

[2] The Lagrangian particle dispersion model FLEXPART version 10.4

Pisso, I., Sollum, E., Grythe, H., Kristiansen, N. I., Cassiani, M., Eckhardt, S., Arnold, D., Morton, D., Thompson, R. L., Groot Zwaaftink, C. D., Evangeliou, N., Sodemann, H., Haimberger, L., Henne, S., Brunner, D., Burkhart, J. F., Fouilloux, A., Brioude, J., Philipp, A., Seibert, P., and Stohl, A.: The Lagrangian particle dispersion model FLEXPART version 10.4, Geosci. Model Dev., 12, 4955–4997, https://doi.org/10.5194/gmd-12-4955-2019, 2019

[3] Source attribution using FLEXPART and carbon monoxide emission inventories: SOFT-IO version 1.0

Sauvage, B., Fontaine, A., Eckhardt, S., Auby, A., Boulanger, D., Petetin, H., Paugam, R., Athier, G., Cousin, J.-M., Darras, S., Nédélec, P., Stohl, A., Turquety, S., Cammas, J.-P., and Thouret, V.: Source attribution using FLEXPART and carbon monoxide emission inventories: SOFT-IO version 1.0, Atmos. Chem. Phys., 17, 15271–15292, https://doi.org/10.5194/acp-17-15271-2017, 2017

[4] Flex_extract v7.1.2 – a software package to retrieve and prepare ECMWF data for use in FLEXPART

Tipka, A., Haimberger, L., and Seibert, P.: Flex_extract v7.1.2 – a software package to retrieve and prepare ECMWF data for use in FLEXPART, Geosci. Model Dev., 13, 5277–5310, https://doi.org/10.5194/gmd-13-5277-2020, 2020

[5] Historical (1750–2014) anthropogenic emissions of reactive gases and aerosols from the Community Emissions Data System (CEDS)

Hoesly, R. M., Smith, S. J., Feng, L., Klimont, Z., Janssens-Maenhout, G., Pitkanen, T., Seibert, J. J., Vu, L., Andres, R. J., Bolt, R. M., Bond, T. C., Dawidowski, L., Kholod, N., Kurokawa, J.-I., Li, M., Liu, L., Lu, Z., Moura, M. C. P., O'Rourke, P. R., and Zhang, Q.: Historical (1750–2014) anthropogenic emissions of reactive gases and aerosols from the Community Emissions Data System (CEDS), Geosci. Model Dev., 11, 369-408, https://doi.org/10.5194/gmd-11-369-2018, 2018

[6] Biomass burning emissions estimated with a global fire assimilation
system based on observed fire radiative power

Kaiser, J. W., Heil, A., Andreae, M. O., Benedetti, A., Chubarova, N., Jones, L., Morcrette, J.-J., Razinger, M., Schultz, M. G., Suttie, M., and van der Werf, G. R. (2012). Biomass burning emissions estimated with a global fire assimilation system based on observed fire radiative power. BG, 9:527-554

.

[7] The interactive global fire module pyrE (v1.0)

Mezuman, Keren & Tsigaridis, Kostas & Faluvegi, Gregory & Bauer, Susanne. (2020). The interactive global fire module pyrE (v1.0). Geoscientific Model Development. 13. 3091-3118. 10.5194/gmd-13-3091-2020.

[8] AGOS viewer of FLEXPART (Lagrangian model) footprints and modelled SOFT-IO CO contributions

https://services.iagos-data.fr/atmo-access/footprint

Annexe

Annexe 1: ACTRIS stations configuration

```
[
  {
    "short_name":"PUY",
    "long_name":"Puy de Dôme",
    "longitude":2.964886,
    "latitude":45.772223,
    "altitude":1465,
    "release_heights":"500 1500"
  },
  {
    "short_name":"SAC",
    "long_name":"SIRTA Atmospheric Research Obs",
    "longitude":2.1588888889,
    "latitude":48.7086111,
    "altitude":162,
    "release_heights":"500"
  },
  {
    "short_name":"PDM",
    "long_name":"Pic du Midi",
    "longitude":0.141944,
    "latitude":42.936667,
    "altitude":2877,
    "release_heights":"500 1000"
  },
  {
    "short_name":"OPE",
    "long_name":"Observatoire Perenne de l'Environnement",
    "longitude":5.50555556,
    "latitude":48.562222,
    "altitude":392,
    "release_heights":"500"
  },
  {
    "short_name":"RUN",
    "long_name":"La Réunion - Maïdo atmospheric observatory",
    "longitude":55.383006,
    "latitude":-21.076449,
    "altitude":2160,
    "release_heights":"500 1500 2000"
  },
  {
    "short_name":"LTO",
    "long_name":"Lamto",
    "longitude":-5.03333333,
    "latitude":6.216666667,
    "altitude":105,
    "release_heights":"100 500 1500"
  }
]
```