SEDOO-AERIS

GIRAFE - REGIONAL REAL TIME FIRE PLUMES SIMULATION TOOL

TECHNICAL DOCUMENT & USER MANUAL

	Name	Organisation	Date	Visa
Written by:	Daria Malik	Magellium	22/12/2023	
Checked by:	Vanessa Pedinotti	Magellium		
Approved by:	Béatrice Berthelot	Magellium		

Document reference:	
Edition.Revision:	2.0
Date Issued:	06/05/2024
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	22/12/2023	Creation of document	
2	0	06/05/2024	Document update	

Dissemination level

PU	Public	x
PP	Restricted to other programme participants	
RE	Restricted to a group specified by the consortium	
CO	Confidential, only for members of the consortium	

Contents

1. GIRAFE overview

GIRAFE is a simulation tool that provides estimated trajectories of the fire plumes or plumes of other CO emissions, like anthropogenic emission, for example. These trajectories are computed by the Lagrangian particle dispersion model FLEXPART (v10.4) which is suitable for the simulation of a large range of atmospheric transport processes and atmospheric components. The model uses the European Centre for Medium-Range Weather Forecasts (ECMWF) meteorological data for the trajectories estimation and the CO emission inventories or fire detection data for the source detection. The trajectories are represented by the concentration of atmospheric components on the latitude-longitude-altitude grid for date and times of simulation defined by the user. The visualization of results allows to study and see where the particles will be potentially drifted and carried on (Fig.1).

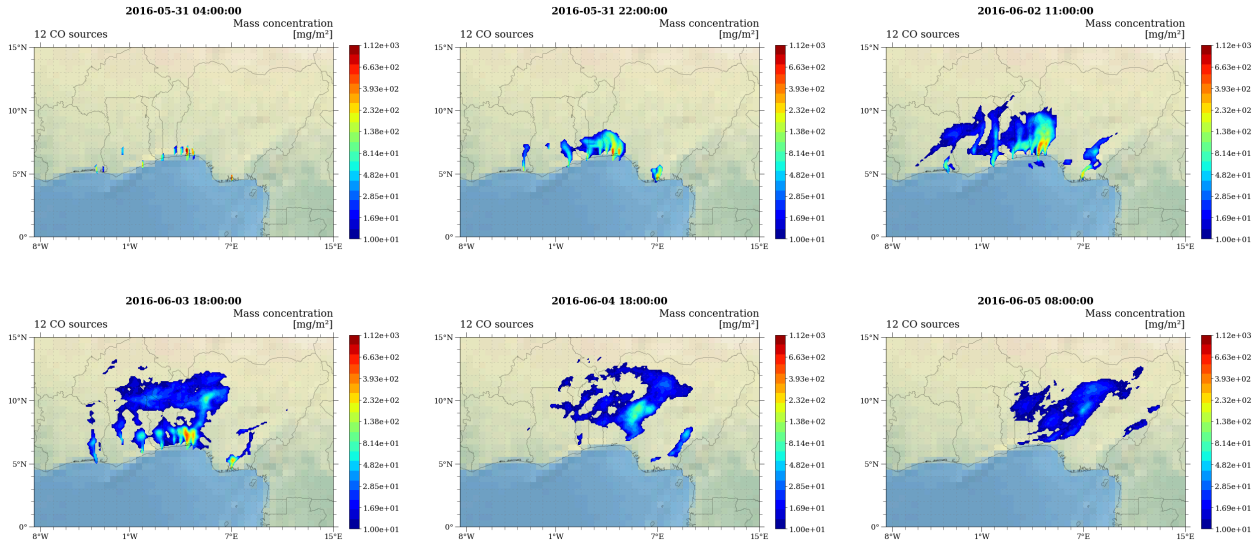

Figure 1: Example of the simulation output for fire plumes trajectories in West Africa

For the detection of CO emissions or fires GIRAFE uses external files: "emission inventories" or MODIS Fire products respectively. The emission inventories contain concentrations of different emissions on the latitude-longitude grid, for various emissions sources (aviation emissions, volcanic SO2 emissions, biomass burning etc). The MODIS fire products contain thermal anomalies / active fires latitude-longitude locations with their respective confidence levels. Current version of GIRAFE supports the majority of the emission inventories. The requirements are : the file needs to be in the netCDF format and respect the standard names of the time, latitude and longitude variables. The supported MODIS fire product is the MCD14DL collection, which can be used either in the txt or csv format.

For the easier installation, maintenance and portability, GIRAFE executables and dependencies have been wrapped up into a Singularity container.

1.1 GIRAFE architecture

GIRAFE tool 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. Hence, the main executable that launches simulations is the FLEXPART executable installed in the Singularity container.

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

On top of the FLEXPART executable and configuration, an overlay Python script was created to facilitate user experience with the GIRAFE tool. This script represents the user interface for this tool and allows to configure the simulation through one simple configuration xml file, without changing manually every FLEXPART input file, and thus also allows to automate the process in case of multiple simulations. The output of the simulation (concentrations of CO emissions) can either be in the binary or netCDF format (based on your FLEXPART configuration). However, the post-processing of the user interface which allows to create output visualization supports only the netCDF format.

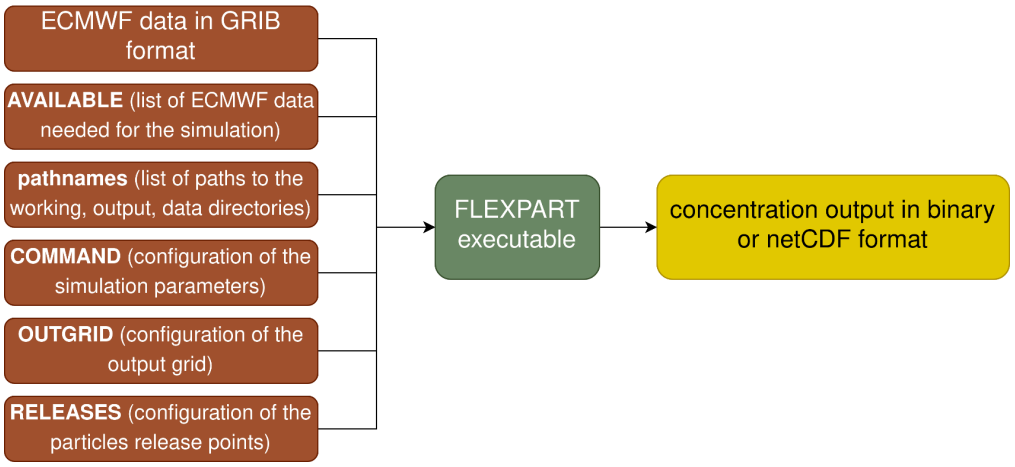

Figure 2 : scheme of the FLEXPART simulation input/output which is the main element of the GIRAFE tool

1.2 FLEXPART input files

FLEXPART needs the following three types of input files [1] :

1. The text file `pathnames` which must be located in the directory where FLEXPART is executed. It must contain four 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 (given in line 1 of `pathnames`) called `options`. The brief description of these files is given in the 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.

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

1.2.1 Run-defining settings

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

1.2.1.1 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 [1].

1.2.1.2 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.

1.2.1.3 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.

1.2.1.4 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.

1.2.2 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 `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.

1.2.2.1 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 [2]. Once you logged in onto the MARS server (ecs or hpc), you can follow the below steps.

First, download 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 some 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 [2]). 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

$script $parameterlist
```

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. 3).

```
217     submit_sbatch_job(os.path.join(_config.PATH_REL_JOBSCRIPTS,
218                                    _config.FILE_INSTALL_COMPILEJOB))
219
220     # submit_job_to_ecserver(c.install_target,
221     #                        os.path.join(_config.PATH_REL_JOBSCRIPTS,
222     #                                     _config.FILE_INSTALL_COMPILEJOB))
```

Figure 3 : 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 if 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. 4). Also, always in the same script, it is necessary to add the `ecmwf-toolbox` in the `module load` command (Fig. 5); after that the `flex_etxract` is ready for use.

```
45
46   flex_extract_path=/home/as2/FLEX_EXTRACT/flex_extract_v7.1.3/
47
```

Figure 4 : flex_extract_path variable to check

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

Figure 5 : ecmwf-toolbox to add in module load command

1.2.2.2 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. 6), or by using the `CONTROL` file (Fig. 7) where parameters are indicated and which allows a more precise and complex configuration of the parameters.

For a simple analysis and/or forecast data for GIRAFE, the script parameters can be set as shown on Fig. 6 and the `CONTROL` file can be as follows (Fig. 7) :

```
28  QUEUE=None
29  START_DATE=None
30  END_DATE=None
31  DATE_CHUNK=None
32  JOB_CHUNK=3
33  BASETIME=None
34  STEP=None
35  LEVELIST=None
36  AREA=None
37  INPUTDIR="/ec/res4/hpcperm/as2/GIRAFE"
38  OUTPUTDIR="/ec/res4/hpcperm/as2/GIRAFE"
39  PP_ID=None
40  JOB_TEMPLATE="submitscript.template"
41  CONTROLFILE='my_control'
42  DEBUG=0
43  REQUEST=2
44  PUBLIC=0
```

Figure 6 : 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 7 : CONTROL file that can be used for the GIRAFE data extraction

In the script parameters, the variables `INPUTDIR` and `OUTPUTDIR` should be set with paths where the data and eventual intermediate files will be stored. The `CONTROLFILE` is

the name of the CONTROL file to use, and the file itself must be placed in the `Run/Control` directory of the `flex_extract` root folder.

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.

A Bash script that facilitates the data extraction and the simulation launch was also created specifically for the use cases of the GIRAFE tool. This script is described in section 2.5.

1.3 FLEXPART output files

At each output time, FLEXPART generates files comprising gridded output, with separate files produced for each species. The file-naming convention follows the pattern `grid_[type]_[date]_[nnn]`. For forward runs, the type can be either *conc* or *pptv*, representing concentrations and mixing ratios, or *flux* for 3-D mass fluxes across grid cell faces. In forward simulations, wet and dry deposition fields are computed on the same horizontal output grid and appended to files named `grid_conc_date_nnn` and `grid_pptv_date_nnn`. The accumulation of deposited matter occurs throughout the model run, generally increasing with model time. However, for species with radioactive decay, losses are possible.

For surface points, concentrations or mixing ratios in forward simulations can be independently calculated from the grid using a kernel method. The results are recorded in files named `receptor_conc` and/or `receptor_pptv`.

If the particle dump option is activated, in addition to the gridded output, particle coordinates, along with additional variables such as pressure, humidity, density, tropopause height, atmospheric boundary layer (ABL) height, and orography height, are recorded in binary files named `partposit_date`. FLEXPART version 10.4 also offers the option to write out time-averaged particle positions and meteorological data, recorded in files named `partposit_average_date`. In plume trajectory mode, for every release, the positions of trajectory clusters representing the centers of mass of all released particles are recorded in the file `trajectories.txt`.

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`. For forward runs, additional files `grid_pptv_date_nnn` may be generated, containing data such as volume mixing ratios (requiring molar weight in `SPECIES_nnn` file). 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_conc*	1	1	kg	ng.m-3
grid_conc*	1	2	kg	ppt by mass
grid_conc*	2	1	1	ng.m-3
grid_conc*	2	2	1	ppt by mass
grid_conc*	1	1 or 2 (deposition)	kg	ng.m-2
grid_conc*	2	1 or 2 (deposition)	1	ng.m-2
grid_pptv*	1	1	1	ppt by volume

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 (Figure 8). 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.

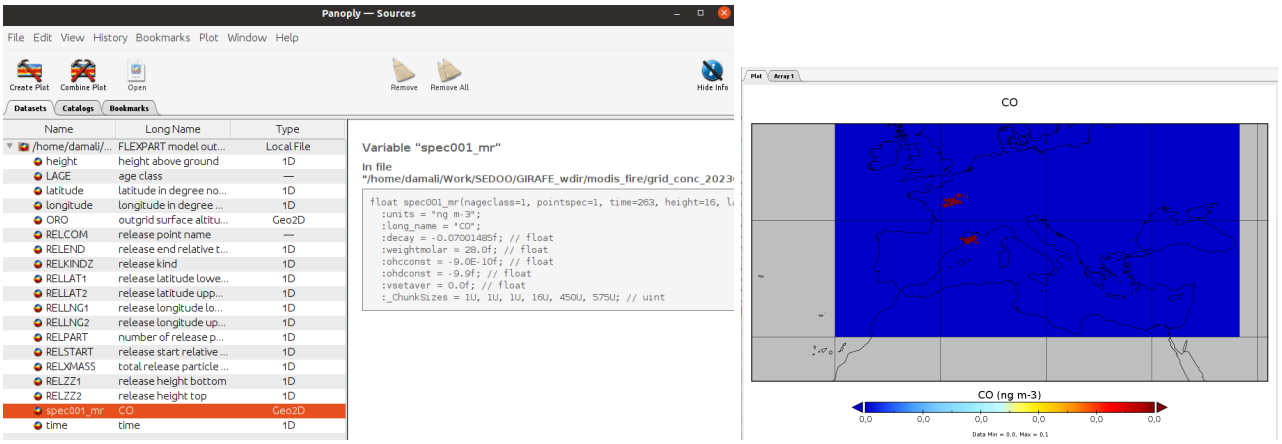

Figure 8 : FLEXPART output in netCDF format, opened with Panoply

2. GIRAFE simulation for user

FLEXPART tool is containerized into a Singularity container along with its dependencies and aux libraries needed for the post-processing or pre-processing of the GIRAFE simulation. Such containerization allows any user to run it without previous libraries installation and source code configurations (except the installation of the Singularity application itself).

The file `girafe-container.def` is a Singularity Definition File, which 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 girafe.sif girafe-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.

2.1 GIRAFE User interface

GIRAFE User interface encapsulates the FLEXPART architecture seen earlier (Figure 1) into a user-friendly "black box" designed specifically for the GIRAFE scenarios and its purpose of use. GIRAFE functioning can be summarized in following steps :

1. simulation parameters are defined by user through a configuration file `user-config.xml` (which can be renamed, but has to follow a predefined syntaxe)
2. Python script `girafe.py` reads the configuration file, writes input files for FLEXPART, compiles sources codes to obtain FLEXPART executable suited to the current simulation and launches the simulation
3. if the simulation ran successfully, the simulation output is written and the post-processing is performed on the netCDF output (if available) to generate images with results visualization

By default, log and error messages are displayed in the terminal, where the simulation was launched. This message output can also be redirected into a file via bash `>` / `>>` operators.

⚠ The simulation must be launched inside the Singularity container.

2.2 Tool scripts and necessary files

2.2.1 Configuration file

Configuration file `user-config-basic.xml` is an XML file which allows users to define simulation parameters, such as dates of the simulation, latitude-longitude grid of the simulation etc. This file has mandatory nodes and option nodes; last ones are more suitable for more experienced FLEXPART users. In the case of a beginner user, optional nodes can be omitted in the configuration file and will be handled by the Python script and configured to their default values[1]. These settings summarize all of the run-defining parameters needed for the FLEXPART input files (`RELEASES`, `OUTGRID` etc). Below (Table 3) is the description of mandatory configuration nodes; optional nodes for more proficient users can be found in the `user-config-pro.xml` file.

Table 3 : Description of configuration mandatory parameters

```
<config>
   <girafe>

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

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

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

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

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

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

simulation start and end date and time, meaning from when to when the trajectories of particles are estimated

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

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

number of point of meteorological fields in x and y direction

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

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

[1] Default values were taken from FLEXPART official documentation

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

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

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

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

```
<command>
   <time>
      <output> 3600 </output>
   </time>
   <iOut> 9 </iOut>
</command>
```

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; possible options for last one being :

Output types and formats	*Binary*	*netCDF (binary option number +8)**
mass	*1*	*9*
pptv	*2*	*10*
mass + pptv	*3*	*11*
plumes	*4*	*12*
mass + plumes	*5*	*13*

** Options 9,11 and 13 are supported by the Python post-processing for the quicklooks creation of mass concentration*

```
<releases>
   <species> 22 </species>
```

the node release concern the particle injections (emitted by the human activities or f.ex. forest fires); species node contains the number of desired species from available ((2) O3, (3) NO, …(22) CO etc; see FLEXPART documentation for more species)

```
   <fire_confidence> 80 </fire_confidence>
```

when using MODIS fire products for fire CO emissions, the user must indicate the minimum fire confidence

between 0 and 100%; hot points with confidence below this threshold won't be taken into account in emissions calculations

```
            <release>
               <start_date>20230501</start_date>
               <start_time>000000</start_time>
               <duration>00120000</duration>
               <altitude_min>10</altitude_min>
               <altitude_max>100</altitude_max>

               <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>



      <paths>
         <working_dir>/user/girafe/wdir</working_dir>
         <ecmwf_dir>/data/grib</ecmwf_dir>
         <emissions>/data/modis_product.csv</emissions>
         <emissions_variable> sum </emissions_variable>
      </paths>
   </girafe>
</config>
```

each release is characterized by the start date YYYYMMDD, time HHMMSS and its duration DDHHMMSS; the simulation will emit particles between indicated minimum and maximum altitudes (meters above ground)

for each date/time release it is possible to define multiple ROI with corresponding names and latitude/longitude boundaries; the emissions will then be searched and calculated inside these ROIs

paths node is where must be indicated paths to the :

- *working directory (simulation run-time files, executables etc)*
- *directory with ECMWF files*
- *path to the emissions file (MCD14DL csv fire products or emission inventory in netCDF format); in the case of an emission inventory, the node "emission variable" is mandatory and indicates the variable in netCDF file which contains desired emissions*

2.2.2 Python script

Python code `girafe.py` is the main script that the user has to launch to perform a simulation. This script has one mandatory input argument:file path of the XML configuration file. This script prepares the working directory for the simulation, writes FLEXPART input files, launches the simulation and performs the post-processing.

```
usage: girafe.py [-h] [-gc CONFIG]

Python code that prepare all FLEXPART inputs and launch FLEXPART simulations based on
your configuration.xml file.
```

```
options:

  -h, --help                          show this help message and exit
  -gc CONFIG, --config CONFIG         file path to configuration xml file
```

In addition to these codes the user must have access to the meteorological data and the emission inventory files. The paths to these files have to be indicated in the configuration file `user-config.xml`.

2.2.3 Particle emissions products

As it was presented earlier, GIRAFE supports two types of the particle releases :

- fire detection via MODIS MCD14DL products
- emission inventories

The emission inventories contain concentrations of different emissions on the latitude-longitude grid, for various emissions sources (aviation emissions, volcanic SO2 emissions, biomass burning etc). Current version of GIRAFE supports the majority of the emission inventories. The requirements are : the file needs to be in the netCDF format, variables of time , latitude and longitude must satisfy netCDF variable name conventions (either via standard name or the name of the variable), and the necessary emission variable must have units of kg.m-2.s-1. The amount of emissions will then be computed based on releases date/time/duration/location settings indicated in the configuration file and the data offered by the emission inventory. A variety of different inventories can, for example, be found on https://eccad.sedoo.fr [3].

The supported MCD14DL MODIS fire products contain latitude-longitude locations of thermal anomalies and active fires, along with their respective confidence levels. The MCD14DL products are distributed in the txt or csv extension. The Python pre-processing function will find the thermal anomalies within the releases lat-lon domain and with the same start date as the one requested by the user. The start *time* of each release will be taken from the MODIS product. However, the duration of these releases, and the minimum confidence level of the thermal anomalies detection must be indicated by the user. More information about this collection can be found on https://www.earthdata.nasa.gov/learn/find-data/near-real-time/firms/mcd14dl-nrt [4]. Pay attention to download the data in csv or txt (comma-separated text files) formats, as these formats are the only ones supported by current version of GIRAFE.

2.3 GIRAFE simulation launch

The python script has to be executed inside the Singularity container where the GIRAFE was installed (`girafe.sif`). For these commands to work, the meteorological

GRIB data must already be present on the machine. Refer to the section 1.2.2.2 or 2.5 for more information on how to extract the GRIB data from ECMWF. After the data is ready, there are multiple methods to launch the GIRAFE simulation.

2.3.1 Interactive launch

If you want to launch the simulation in the interactive mode and be able to continue to work in the Singularity container, you can follow these steps:

1. launch Singularity container in the shell mode

```
$ singularity shell path/to/the/container.sif
```

2. launch Bash script with the simulation configuration

```
Singularity>
Singularity> python3 /my/path/girafe.py --config /path/to/the/user-config.xml
```

The log messages will be displayed in the terminal; it is possible to redirect the output into a file:

```
Singularity> python3 /my/path/girafe.py --config
/path/to/the/user-config_1.xml > /user/girafe/simulation_1.log
```

2.3.2 Non-interactive launch

It is also possible to launch the script in one go with the `singularity exec` command :

```
$ singularity exec path/to/the/container.sif python3 /my/path/girafe.py
--config /path/to/the/user-config.xml
```

All of the methods described above can be wrapped into cron, bash scripts or slurm jobs depending on one's needs and number of simulations.

2.3.3 Bind option

If during the simulation the input ECMWF data, or other files and/or directories, were not found, while the configuration is correct and the data is present, this could

mean that the Singularity container does not recognize the directory where the GRIB data is stored. To fix this problem, the `bind` option can be used.

The `--bind` option allows to map directories on the host system to directories within the container. When Singularity 'swaps' the host operating system for the one inside your container, the host file system becomes partially inaccessible. The system administrator has the ability to define what bind paths will be included automatically inside each container. Some bind paths are automatically derived (e.g. a user's home directory) and some are statically defined (e.g. bind paths in the Singularity configuration file). In the default configuration, the directories `$HOME`, `/tmp`, `/proc`, `/sys`, `/dev`, and `$PWD` are among the system-defined bind paths. Thus, in order to read and/or write files on the host system from within the container, one must bind the necessary directories if they are not automatically included. Here's an example of using the `--bind` option and binding `/data` on the host to `/mnt` in the container (`/mnt` does not need to already exist in the container):

```
$ ls /data
bar  foo
$ singularity exec --bind /data:/mnt my_container.sif ls /mnt
bar  foo
```

You can bind multiple directories in a single command with this syntax:

```
$ singularity shell --bind /opt,/data:/mnt my_container.sif
```

This will bind `/opt` on the host to `/opt` in the container and `/data` on the host to `/mnt` in the container.

2.4 GIRAFE working directory structure

The working directory of the simulation is the directory where input files for the simulation, the executable for simulation launch and the output of the simulation will be stored. This directory follows the structure describe on the Fig.3:

```
/root
 └──my_simulation_wdir/          -> working directory
      ├──options/                -> FLEXPART configuration files prepared by the
      │    ├──COMMAND            girafe.py and FLEXPART unchangeable input files
      │    ├──OUTGRID            copied from the source
```

```
│   ├──RELEASES
│   ├──. . .
│   └──SPECIES/
│      ├──SPECIES.README
│      ├──SPECIES_001
│      └──...
├──output/                -> output directory
├──quicklooks/            -> quicklooks created by the girafe.py
├──flexpart_src/          -> FLEXPART source code
├──AVAILABLE              -> AVAILABLE input file for FLEXPART
├──pathnames              -> pathnames input file for FLEXPART
└──FLEXPART               -> FLEXPART executable
```

Figure 9 : Working directory structure after a GIRAFE simulation

3. GIRAFE meteorological data extraction

A script that unites the data extraction and the simulation launch was created for a more transparent user experience. This script takes as input 1) the xml configuration file, the same that is used with the `girafe.py` script, and 2) additional configuration file where the paths to the utility directories and files are indicated.

The additional configuration file has mandatory parameters to set and has to follow the syntaxe described below:

```
GIRAFE_CONFIG_FILE="./girafe-config-an.xml"
WDIR="/working/dir/on/MARS/server"
FLEX_EXTRACT_ROOT="/home/user/FLEX_EXTRACT/flex_extract_v7.1.3"
DATA_OUTPUT_DIR="/data/dir/on/MARS/server"
REMOTE_ADDRESS="remote.where.simulation.will.be.launched"
REMOTE_USER="remote_user"
REMOTE_CONTAINER_PATH="/remote/path/girafe.sif"
REMOTE_PYTHON_PATH="/remote/path/girafe.py"
LAUNCH_SIMULATION=true[false]
```

Paths indicated in the GIRAFE XML configuration file for a data extraction/simulation should correspond to the paths on the machine where the simulation will be run.

The script will then operates as follows :

1. extract the data with `flex_extract` based on the GIRAFE xml configuration file and the additional configuration file; the working files like requests and `flex_extract` log output will be saved in the `WDIR` path, and the data will be saved in the `DATA_OUTPUT_DIR`
2. if the remote address is provided, the data will then be copied to this remote, in the `ecmwf_dir` which is set in the xml configuration file

3. if the `LAUNCH_SIMULATION` option is true, then the input files for the GIRAFE simulation will be created, and the simulation will be launched remotely on the remote address set above; if the `LAUNCH_SIMULATION` option is false, then the data will just be extracted on the MARS server and there will be no simulation launch.

3.1 Data and folder organization

The structure of the working and data folders is summarized on Fig. 9. On the MARS server the working directory path will be used as it is; the directory will be created if it does not exist and will host working files like CONTROL files for flex_extract and prepared SLURM job script for the remote server. This path should be different for each simulation if multiple simulations/data extraction are running at once. The given data directory path will be used as a parent directory; the directory will be created if it does not exist and will host subdirectories with "`<start_date>_<end_date>`" name which will be set automatically by the data extraction script.

On the remote server where the GIRAFE simulation itself will be run the working directory path set by the user will be used as it is; more details about its content was presented in the section 2.4. The data directory will also directly host the transferred data in contrast to the MARS data path where the data is separated into subfolders.

```
/root_ecmwf_mars_server
 ├─user_wdir/
 │  ├─CONTROL_FILE(s)
 │  └─job_girafe.sh
 └─user_data_dir/
    ├─20220101_20220110
    │  ├─EN22010100
    │  ├─EN22010103
    │  ├─...
    │  └─EN22011100
    └─20230613_20230617
       ├─EN23061300
       ├─...
       └─EN23061800
```

```
/root_user_remote_server
 ├─remote_user_wdir/
 │  ├─AVAILABLE
 │  ├─FLEXPART
 │  ├─flexpart_compile.out
 │  ├─flexpart_src/
 │  ├─[girafe-configuration-name].xml
 │  ├─girafe-simulation.out
 │  ├─job_girafe.sh
 │  ├─options/
 │  ├─output/
 │  ├─pathnames
 │  └─quicklooks/
 └─remote_user_data_dir/
    ├─EN23061300
    ├─...
    └─EN23061800
```

Figure 9 : tree structure of the data and working directories on the MARS server and user remote server

Bibliography

[1] 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

[2] 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.

[3] ECCAD, GEIA Global Emission InitiAtive data portal (AERIS, French data service for Atmosphere)

https://eccad.sedoo.fr

[4] MODIS/Aqua+Terra Thermal Anomalies/Fire locations 1km FIRMS V0061 NRT (Vector data), DOI: 10.5067/FIRMS/MODIS/MCD14DL.NRT.0061

https://www.earthdata.nasa.gov/learn/find-data/near-real-time/firms/mcd14dl-nrt

Annexes

Annexe 1 : get GRIB grid size

Below is the command that can be used inside the GIRAFE singularity container in order to retrieve the x/y size of the meteorological fields. These values then should be used in XML configuration file.

```
$ singularity shell girafe.sif
Singularity>
Singularity> /usr/local/grib2/wgrib2/wgrib2 EN23122600.grib -d 1 -V
1:0:vt=2023122600:1 hybrid level:anl:UGRD U-Component of Wind [m/s]:
                        ndata=64800:undef=62027:mean=44.126:min=-0.114808:max=101.543
                                                              grid_template=0:winds(N/S):
    lat-lon grid:(360 x 180) units 1e-06 input WE:SN output WE:SN res 48
    lat -89.500000 to 89.500000 by 1.000000
    lon 0.000000 to 359.000000 by 1.000000 #points=64800
```