SEDOO-AERIS

BAMED - Boundary Layer Pressurized Balloons trajectory simulation

Technical Document
& User Manual

	Name	Organisation	Date	Visa
Written by:	Daria Malik	Magellium	13/10/2023	
Checked by:	Vanessa Pedinotti	Magellium		
Approved by:	Béatrice Berthelot	Magellium		

Document reference:	
Edition.Revision:	3.0
Date Issued:	03/05/2024
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	23/03/2023	Creation of document	
2	0	13/10/2023	Document update	
3	0	03/05/2024	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. BAMED Overview

BAMED is a Fortran based numeric tool that allows to simulate Boundary Layer Pressurized Balloons (BLPB) trajectories based on the given meteorological conditions. The model allows simulating balloon path from a given launch site (geographical coordinates) on the given date and time of the launch (Fig. 1).

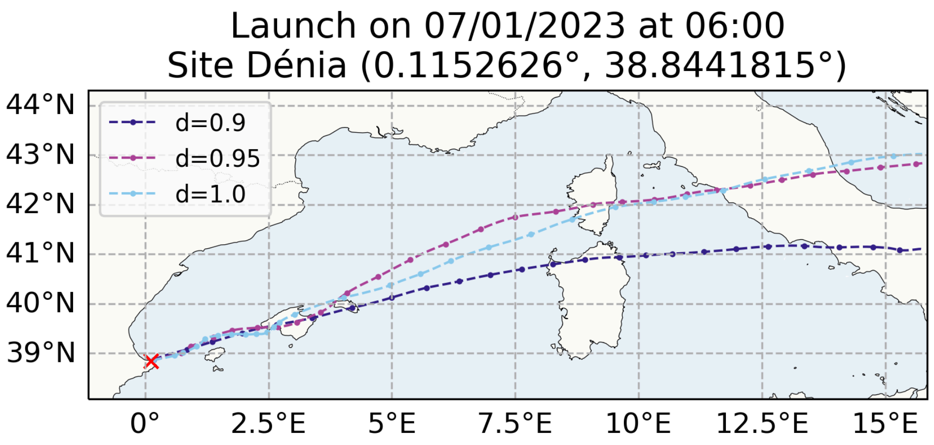


Figure 1 : example of balloons trajectories estimated by the BAMED simulation tool

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

1.1 BAMED architecture

The BAMED main executable that launches simulations is called `traj_blpb` and is located in the source directory of the model in the Singularity container : `/usr/local/bamed/src/traj_blpb`.

The executable needs as input a configuration file `blbp.dat` and a `list.dat` file with a list of the ECMWF GRIB data to use (Fig. 2). The names of these files are not to change. The `traj_blpb` can be located anywhere but has to be called from a directory where the two input files are located. The output file `blpb_[date]_[hour]_[density]_[number].dat` will be written in the directory from which the executable has been called. One simulation launch can simulate multiple balloons' trajectories with different densities, but same date, time and launch site coordinates (which are defined in the `blbp.dat` file).

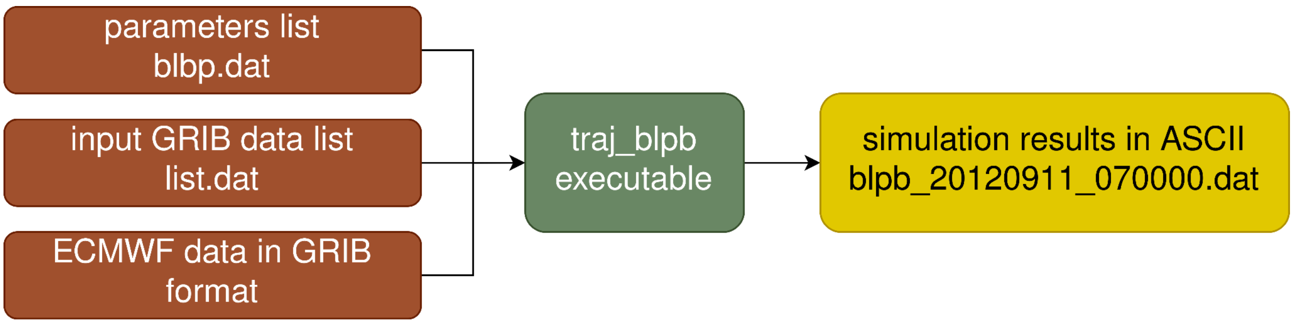


Figure 2 : scheme of the BAMED simulation input/output

1.2 BAMED input files

1.2.1 blbp.dat

The `blbp.dat` configuration file contains all of the needed information about the launch and the input meteorological data. The file syntax is as follows :

```
********************************************
* Input file for the BLBP trajectory model *
********************************************

* COMPUTATIONNAL DOMAIN
-10.0                             WEST LONGITUDE (° EAST)
10.0                              EAST LONGITUDE (° EAST)
38.0                              SOUTH LATITUDE (° NORTH)
48.0                              NORTH LATITUDE (° NORTH)
0.1                               GRID SIZE IN DEGREES
48                                NUMBER OF VERTICAL LEVELS
137                               MAX ID NUMBER OF LEVELS
*
* TIME MANAGEMENT
20120911                          START DATE (YYYMMDD)
100000                            START TIME (HHMMSS)
2                                 SIMULATION DURATION IN NUMBER OF DAYS
1                                 TIMESTEP OF DATA
0.1                               TIMESTEP OF SIMULATION IN HOURS
*
* GRIB DIRECTORY
/home/grib_data/                  GRIB DIRECTORY
*
```

```
* BALOON INFO
1                                 NUMBER OF LAUNCH SITES
*
* LAUNCH SITE
4.273968                          SITE LONGITUDE
39.82833                          SITE LATITUDE
1                                 NUMBER OF BALOONS
0.95                              FIRST BALOON DENSITY
0.0                               DENSITY RANGE BETWEEN BALOONS
*
```

"Computational domain" block contains the information about the ECMWF GRIB files that will be used for the simulation : latitude and longitude boundaries of the data, its spatial resolution and information about vertical levels. The last two parameters are the amount of vertical levels contained in files and the maximum ID number of present levels (e.g. if the GRIB data is on levels from 10 to 16, the amount of levels is 7 and the maximum ID number is 16).

The "*Time management"* block contains the information about the simulation start date and time, the simulation duration (in number of days), the timestep of your GRIB data (in hours), and the timestep of the simulation (in fraction of hour). For example, in the configuration above trajectories will be simulated from 11/09/2012 10:00 until 13/09/2012 10:00 (2 days); the meteorological data provided by the user is an hourly data (one file per hour); balloons position will be calculated for every 6 minutes (0.1 hour). The simulation start date and time could be different from the balloon launch date and time, but nothing will be simulated prior to the balloon launch. Here, the balloon is launched at 10:00 on 11/09/2012. N.B: the data timestep is a very important parameter that should be set carefully depending on your available GRIB data. Based on this value and the duration of simulation the program will search for an exact amount of GRIB files through the `list.dat` file. If this operation fails, the simulation is not launched and exits on error. Furthermore, the timestep between GRIB data files must be the same.

The "*Grib directory"* must contain the path to the directory with the necessary meteorological data. The path string must end with the slash "/".

"Baloon info" should always contain "1" for Fortran file reading purposes.

"Launch site" block contains information about the launch site (latitude and longitude of the launch position) as well as the balloon configuration (number of balloons, the balloon density and eventual density range). The parameter "number of

balloons" has priority over the "density range"; it means that even if the density range is different from zero but the number of balloons is equal to 1, then only one balloon with the "first balloon density" will be simulated. In case of more than one balloon, their respective densities will decrease from the initial density by the density range step (examples in the Table 1 below). Regardless of the number of balloons, the launch coordinates will be the same for all of them for a one given `blbp.dat` configuration.

Table 1 : Examples of different density configurations

Number of balloons	*Initial density*	*Density range*	*Densities of simulated balloons*
1	0.95	0.0	0.95
1	0.95	0.1	0.95
2	1.00	0.05	1.00, 0.95
4	1.10	0.1	1.10, 1.00, 0.90, 0.80

1.2.2 list.dat

The `list.dat` file should contain the list of the meteorological GRIB files needed for the simulation. The user has to make sure that the data listed in this file is consistent with the simulation configuration (temporal and spatial cover).

The format of the `list.dat` is as follows (example below) :

- date of the data in `YYYYMMDD` format
- two spaces
- time of the data in `HHMMSS` format
- two spaces
- name of the corresponding file (name string must have a maximum length of 80 characters)

```
20230206  000000  23020600.grib
20230206  030000  23020603.grib
20230206  060000  23020606.grib
20230206  090000  23020609.grib
20230206  120000  23020612.grib
20230206  150000  23020615.grib
20230206  180000  23020618.grib
20230206  210000  23020621.grib
```

```
20230207  000000  23020700.grib
20230207  030000  23020703.grib
20230207  060000  23020706.grib
20230207  090000  23020709.grib
20230207  120000  23020712.grib
20230207  150000  23020715.grib
20230207  180000  23020718.grib
20230207  210000  23020721.grib
20230208  000000  23020800.grib
```

1.2.3 Meteorological data

The ECMWF data for the BAMED simulation must be in GRIB format and data variables must be organized in separate files based on their date and time, so there is one file per one given day and time of the day.

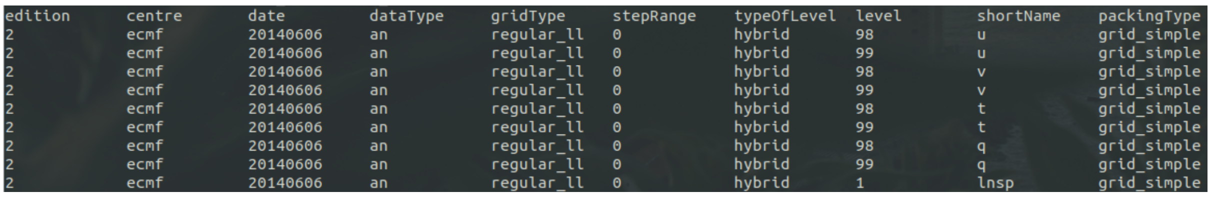
The mandatory meteorological fields are:

- U component of wind (U)[1]
- V component of wind (V)
- specific humidity (q)
- temperature (t)
- logarithm of surface pressure (lnsp)

The optional meteorological fields are:

- vertical velocity (w)
- total precipitation (tp)

All of the variables, except the logarithm of surface pressure, are level variables. An example of the `grib_ls` command on one of the files is presented on Fig. 8 below.



```
edition      centre       date         dataType     gridType     stepRange    typeOfLevel  level        shortName    packingType
2            ecmf         20140606     an           regular_ll   0            hybrid       98           u            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       99           u            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       98           v            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       99           v            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       98           t            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       99           t            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       98           q            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       99           q            grid_simple
2            ecmf         20140606     an           regular_ll   0            hybrid       1            lnsp         grid_simple
```

Figure 8 : example of grib_ls command output for the BAMED ready meteorological GRIB file

The data should cover the desired time of your simulation, *and* be extended 1 day before and after the simulation. For example, if your simulation period is from 5th to 7th of March 2021, the data should cover the time period from 4th to 8th of March 2021. The data extraction script that will be presented later in this document takes

[1] the short name of the field in the parentheses corresponds to the short name of this field in the ECMWF MARS database

care of these extensions, so in the input configuration you should define your true simulation time period, without taking into account this +- 1 day surplus.

1.3 BAMED Output Files

BAMED outputs balloons' trajectories in the ASCII format with a semicolon separator. There is one `.dat` file per balloon. The output name is always in the `blpb_[date]_[hour]_[density]_[number].dat` format. The file itself contains following information :

```
launch_lat;44.4680000000
launch_lon;11.3240000000
launch_date;20230701
launch_time;030000
density;0.900
Variables:;time;longitude;latitude;altitude;vertical_velocity;total_precipitation
Units:;seconds;degrees_east;degrees_north;meters;Pa/s;meters
;360.00;11.30015857;44.46197394;2751.19;-.48956685;.00065226
;720.00;11.27665363;44.45572407;2746.49;-.46246899;.00068714
;1080.00;11.25351685;44.44912894;2736.90;-.42619016;.00074184
;1440.00;11.23072450;44.44204434;2721.75;-.38302931;.00078967
;1800.00;11.20819935;44.43430726;2700.33;-.33591633;.00081080
;2160.00;11.18580403;44.42573992;2671.75;-.28843430;.00079018
;2520.00;11.16333724;44.41616158;2634.78;-.24597784;.00072275
;2880.00;11.14052695;44.40542292;2588.68;-.21542174;.00064191
;3240.00;11.11702922;44.39343465;2532.60;-.20389508;.00059015
;3600.00;11.09245787;44.38019171;2465.33;-.21609697;.00061181
end_status;0
```

The header of the file allows to keep track of the launch information : launch position (latitude, longitude), launch date and time as well as the balloon density. Some of this information is also present in the name of the output file.

The time, latitude, longitude and altitude column contain the balloon estimated position information. Columns vertical velocity and the total precipitation are estimated only if these variables are also present in the ECMWF input meteorological files. These variables are not mandatory at input, and it does not impact the results of the simulation. If they are not present, the values in the output file will simply be "0.00".

The last line of the file contains the exit status of the simulation :

- 0 = balloon's launch site is outside of the geographical domain prior to the simulation start or balloon position (during simulation) exits the geographical domain before the requested duration of simulation was reached
- 1 = simulation successful
- 4 = balloon exits through domain bottom-level

- 5 = balloon exits through domain top-level
- -9= density warning, two last estimated balloon positions are identical

In all of the above cases an output file is created even if no balloon position were estimated.

2. BAMED simulation for User

2.1 BAMED Singularity container

BAMED code is containerized into a Singularity container. Singularity is a container platform which allows you to create and run containers that package up pieces of software in a way that is portable and reproducible. You can build a container using Singularity on your laptop, and then run it on many of the largest HPC clusters in the world, local university, company clusters or other. A container is a single file, and the user does not have to worry about how to install all the software you need on each different operating system and system.

The file `bamed-container.def` is a Singularity Definition File, which is a set of blueprints explaining how to build a custom container, install needed softwares and libraries and set environment variables. The command which allows to build a Singularity container (image) is:

```
$ sudo singularity build bamed_image.sif bamed-container.def
```

⚠️One must have sudo rights on the system where the container is built. Otherwise, the option `--fakeroot` can be used with `singularity build` command. If the second solution does not work, you may require to contact the administrator of your machine/server. Once the Singularity image is built, the image can be run anywhere on the host system. The container usage does not require sudo rights.

2.2 BAMED User interface

An overlay script was created to facilitate user experience with the BAMED tool. This overlay is a Python script which requires a special XML configuration file from the user; the script then takes care of writing BAMED input `.dat` files, looping through multiple simulations if needed, launching the simulation itself and creating additional output files for users. This Python code must be executed *inside of the BAMED Singularity container*.

Python script requires an XML configuration file where main simulation parameters are defined by the user. The XML parameters mainly repeat the parameters from the initial `blbp.dat` file, but are more flexible and allow defining multiple different simulations at once.

After the simulation is done, the code takes the ASCII output from Fortran and recreates it in different additional formats for future user's analysis. For each launch site and launch datetime, additional outputs from Python script are :

- PNG plot of the estimated balloon trajectories overlaid on a map
- KML file with balloon positions for a quicker visualization in GIS applications
- netCDF file with ballon positions and launch information

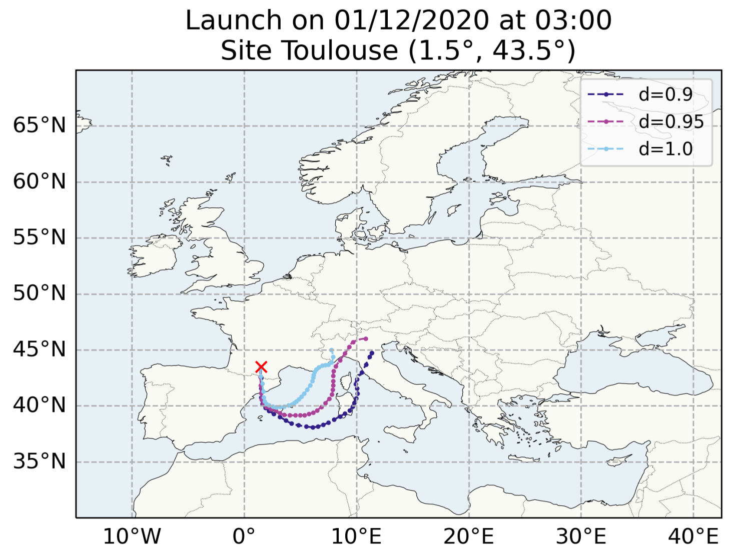


Figure 4 : plot of estimated balloon positions for a launch at (1.5°E, 43.5°N) coordinates, on 1st of December 2020 at 03:00 UTC; balloons have density of 0.9, 0.95 and 1.0 respectively

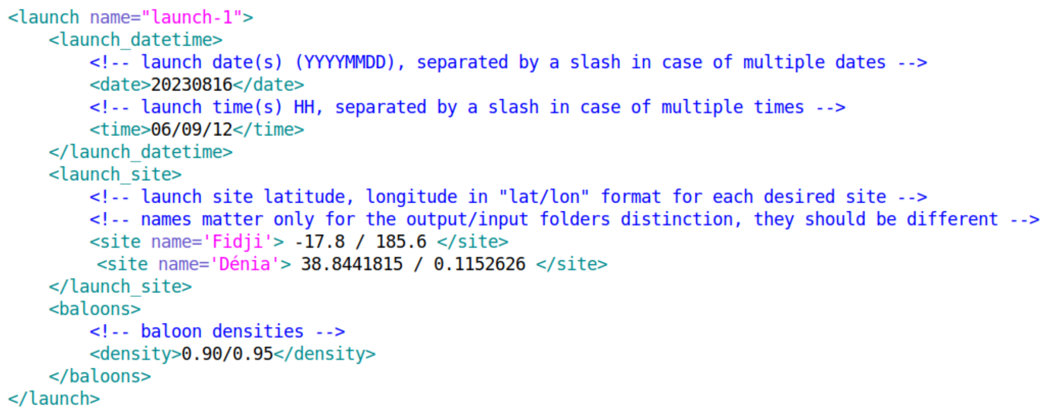
2.2.1 User configuration file

User configuration file is an XML file where it is possible to set up multiple different variables and parameters of the simulation.

The node `<launch>` (Fig. 5) allows configuring multiple date and times of launches as well as multiple launch sites. The node `<date>` allows to indicate one or multiple dates (separated by a slash) for the beginning of simulation(s), and the node `<time>` allows to configure one or multiple times (separated by a slash) for previously requested date(s). The simulation will be performed for each possible date/time combination. *The datetimes of launches have to match exactly the datetimes of available GRIB files.*

The `<launch_site>` node allows you to configure one or multiple launch sites. There has to be only one `<launch_site>` node, but inside this node each child `<site>` node will correspond to one distinct launch place. The syntaxe is `<site name=”my_site”> latitude_value / longitude_value </site>`. The attribute name in the node must be different for each site because these names will be also used to create output directories for each launch site. Simulations of each of the sites will run for every launch date/time combination configured earlier in the file. The longitude values can be either in the [-180°;+180°] or [0°;+360°] convention, but this convention must be the same as the one used for the simulation geographical extent (explained later in this section).

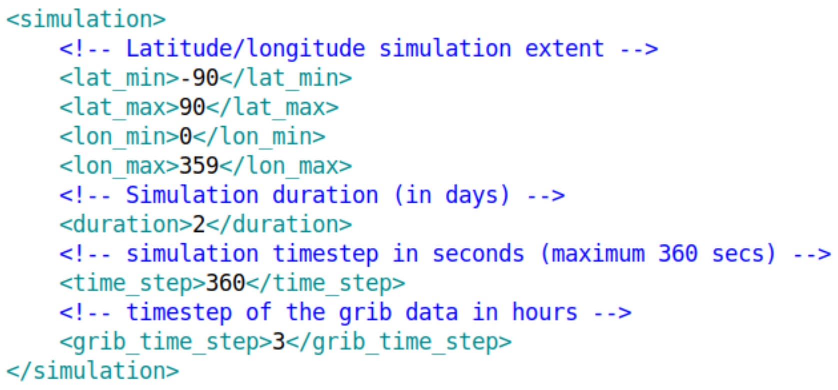
The node `<balloons>` allows to set up the values of the balloons densities. The syntaxe is `<density> density_value_1 / density_value_2 / density_value_3 / etc </density>`.



```
<launch name="launch-1">
    <launch_datetime>
        <!-- launch date(s) (YYYYMMDD), separated by a slash in case of multiple dates -->
        <date>20230816</date>
        <!-- launch time(s) HH, separated by a slash in case of multiple times -->
        <time>06/09/12</time>
    </launch_datetime>
    <launch_site>
        <!-- launch site latitude, longitude in "lat/lon" format for each desired site -->
        <!-- names matter only for the output/input folders distinction, they should be different -->
        <site name='Fidji'> -17.8 / 185.6 </site>
         <site name='Dénia'> 38.8441815 / 0.1152626 </site>
    </launch_site>
    <baloons>
        <!-- baloon densities -->
        <density>0.90/0.95</density>
    </baloons>
</launch>
```

Figure 5 : bamed-config.xml file example, <launch> node

The `<simulation>` node (Fig. 6) allows configuring simulation geographical and temporal extent. The `<grib_time_step>` node indicates the timestep between your input ECMWF data files. The longitude values can be either in the [-180°;+180°] or [0°;+360°] convention, but this convention must be the same as the one used for the launch sites coordinates.



```
<simulation>
    <!-- Latitude/longitude simulation extent -->
    <lat_min>-90</lat_min>
    <lat_max>90</lat_max>
    <lon_min>0</lon_min>
    <lon_max>359</lon_max>
    <!-- Simulation duration (in days) -->
    <duration>2</duration>
    <!-- simulation timestep in seconds (maximum 360 secs) -->
    <time_step>360</time_step>
    <!-- timestep of the grib data in hours -->
    <grib_time_step>3</grib_time_step>
</simulation>
```

Figure 6 : bamed-config.xml file example, <simulation> node

The `<paths>` node (Fig. 7) indicates the script where to search for the input data, and where to store working and output files.

```
<paths>
    <!-- path to the directory where grib files are stored -->
    <grib_path>/my_ecmwf_data/bamed/2023</grib_path>
    <!-- path to the directory where working files for the simulation will be stored -->
    <working_path>/home/work/bamed/launch_1</working_path>
</paths>
```

Figure 7 : bamed-config.xml file example, <paths> node

Two different simulation scenarios (meaning simulations with two different XML configuration files) can be performed simultaneously but in different working directories. Otherwise, the first simulation must be finished before launching the next one.

2.2.2 Python simulation launch script

Python script constitutes a link between user configuration file and the BAMED executable in the Singularity container. The script reads the XML configuration file, and manages all of the simulation combinations : writing input `.dat` files for BAMED, launching simulation, output log message, creating figures for a quick and easy visualization of results, and transforming output results into the KML and netCDF formats in addition to a default ASCII output of the executable.

The Python interface has one mandatory argument : file path to the configuration XML file. The output log messages are displayed in the shell while the simulation is running.

```
usage: bamed.py [-h] [-bc CONFIG]
This Python script allows to manage multiple BAMED simulations based on the
user configuration XML file

optional arguments:
  -h, --help            show this help message and exit
  -bc CONFIG, --config CONFIG
                        Filepath to the xml configuration file (mandatory)
```

2.3 BAMED simulation launch

The python script has to be executed *inside the Singularity container* with the BAMED code. There are multiple methods to do so.

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 after that, you can follow these steps:

1. launch Singularity container in the shell mode

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

2. launch Python script

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

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/bamed.py
--config /path/to/the/bamed-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 a 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 BAMED working directory structure

Directories `launch_site_name_1` and `launch_site_name_2` were created by the Python script based on the user configuration where two launch sites were requested, respectively with names "*launch_site_name_1*" and "*launch_site_name_2*". For each launch site, there will be output `.dat`, `.kml` and `.nc` files, as well as PNG plots for each launch date/time requested in the configuration file.

```
/root
 └wdir/                                -> working directory set by the user
      ├launch_site_name_1              -> one directory per each launch site
      |   ├blpb_YYYYMMDD_HHMMSS_XX_YY.dat   -> balloon positions in different format
      |   ├blpb_YYYYMMDD_HHMMSS_XX_YY.kml
      |   ├blpb_YYYYMMDD_HHMMSS_XX_YY.nc
      |   ├blbp.dat                    -> files needed for the simulation run
      |   ├list.dat
      |   └YYYYMMDD_HHMM.png           -> balloons' trajectories visualization
      |                                   for each launch date/time
      └launch_site_name_2
          └...
```

.

3. ECMWF data extraction for BAMED with automatic simulation

It is possible to extract the ECMWF data for the BAMED simulation and, if configured, to launch the simulation right afterwards on a distant server via a bash script `bamed_extract_ecmwf.sh`. This script extracts the meteorological data, then launches the simulation remotely on a user's requested server. This script has to be launched on one of the ECMWF MARS servers, as it uses the MARS API for meteorological data extraction. The remote server must have BAMED Singularity container along with the BAMED Python user interface.

The script `bamed_extract_ecmwf.sh` contains three main parts :

1. prepare MARS requests for the data extraction, extract the data
2. send the data to the distant server indicated by user
3. launch simulation remotely on this distant server, if configured by the user

Some simulation parameters must be provided by the user in order to configure the data extraction and the simulation. Following parts of this chapter describes in detail how to extract the data with the given script and how to make it launch simulations automatically as soon as the data extraction has finished.

3.1 Input parameters

The script takes as input a configuration txt file. This configuration file contains all the parameters that the user can change and configure:

```
# Data configuration
ID_NAME="palavas_campaign"
START_DATE=""
N_DAYS=2
GRID_RESOLUTION="0.1"
LAT_MIN="26.56"
LAT_MAX="28.92"
LON_MIN="6.02"
LON_MAX="10.00"
WORKING_DIR="/home/user/bamed"
DATA_DIR="/root/data/bamed_data"
SERVER_USER="remote_user"
SERVER_ADDRESS="remote.server.com"
```

```
SERVER_DATA_DIR="/remote_root/remote_bamed_data"
# Simulation configuration
LAUNCH_SIMULATION=true
LAUNCH_DATE="20230701"
LAUNCH_TIME="0300/0900/1500"
LAUNCH_SITE_NAME="Italy/Germany"
LAUNCH_LAT="44.468/48.503"
LAUNCH_LON="11.324/9.246"
BALOONS_DENSITY="0.90/0.95/1.00"
SIMULATION_DURATION="3"
SIMULATION_TIMESTEP="360"
SERVER_WORKING_DIR="/remote_root/remote_bamed_working_dir"
SINGULARITY_CONTAINER_REMOTE_PATH="/remote_root/remote_bamed_src/bam
ed.sif"
PYTHON_REMOTE_PATH="/remote_root/remote_bamed_src/bamed.py"
```

The `ID_NAME` parameter can be a name or an ID character sequence that will allow you to distinguish between your different data requests. Along with the log file of your request that will contain this ID name in the filename, the directory with your corresponding data will also be entitled with this ID and start/end dates of the requested data.

The `START_DATE` corresponds to the start date of the time period for which you would like to extract the data and perform the simulation. This date can be any day up until the J+10 days date and you can also leave it empty. Leaving it empty is equivalent to putting a "today" date.

The `N_DAYS` value is a number of days in your time period or which you would like to extract the data. Note, that the maximum forecast period is 10 days (J+240h). Based on the starting date and the duration of the extraction, if the requested dates are in the past or past+future, the data timestep will be of 3h. The "past" part will have the analysis fields every 6h and the filling of 3h is done with the forecast fields of the corresponding date. The "future" part will be the forecast fields of every 3h. However, if the requested data is only in the future (forecast only), the step is adapted based on the ending date of the request: 1h step if up to +90h, 3h step up to +144h and 6h step for up to +240h. This variable timestep in the forecast data is governed by the ECMWF. The script handles all of these different requests and parameters based just on your starting date and the number of days.

The parameters `GRID_RESOLUTION` and `LAT[LON]_MIN[MAX]` allow the user to configure the geographical extent and spatial resolution of the data. The values of the lat/lon are the boundaries of your ROI and not the coordinates of boundary pixels' centers.

The `WORKING_DIR` parameter corresponds to the directory on the MARS server where the ID directory for the extraction will be created in order to store the working files.

DATA_DIR is the directory on the MARS server where the final data will be stored. Depending on the volume of data, you should consult the documentation [1] to find a suitable directory based on its available free disk space.

SERVER_USER[ADDRESS][DATA_DIR] allows you to send the data onto your local server where you would like to store the data and run simulations. The script anticipates the ssh-copy-id command to establish the password-free connection between the MARS and your final server for the data transfer. If this command has never been executed before, first launch could ask you for your password in order to initiate the automatic authentication. These parameters can also be left empty. In this case, no files will be transferred, and the extracted data will remain in the data directory of the MARS server.

LAUNCH_SIMULATION parameter is set either to "true" or "false" and allows to define whether the simulation should be performed or not after the data extraction and eventual data transfer.

LAUNCH_DATE[TIME][SITE_NAME][LAT][LON], BALOONS_DENSITY, SIMULATION_DURATION[TIMESTEP] and SERVER_WORKING_DIR are parameters that will be used in the XML configuration file for the BAMED Python script described in the previous chapter. Multiple launch dates (and times) can be configured via writing required dates (or times) separated by the slash. The code will then run simulations for every combination of launch date/time. Same method can be applied to multiple density values configuration and the launch sites. For the launch sites, names, latitudes and longitudes are each separated by the slash in the respective variable setting in the parameter file. SERVER_WORKING_DIR is a remote directory where the simulation will be run.

SINGULARITY_CONTAINER_REMOTE_PATH and PYTHON_REMOTE_PATH are file paths to the Singularity container and the Python overlay script respectively, existing on the remote server where the simulation will be performed.

If all the parameters are set properly, then the script will perform all of the steps as indicated in the introduction of this chapter: extract the ECMWF data, send it to your "working" server, prepare the Python configuration file and launch Python inside the singularity container.

3.2 Data and folder organization

The structure of the working and data folders is summarized on Fig. 9. For a given ID name (here campaign_n1 and campaign_n2) and a start date with N days duration, the folder "<id_name>" with a folder "<start_date>_<end_date>" inside it will be created in your working directory, and the same structure will be created in the data directory as well. The remote working directory will follow the same rule. Thus, all of the directories indicated in the configuration file should be "parent" directories for simulation working or data sub-directories (example on the Fig. 9; the directories highlighted in bold are the paths indicated in the configuration file; the `campaign_nX` and `YYYYMMDD_YYYYMMDD` directories are then created by the bash and Python scripts based on the simulation dates parameters). The directories `YYYYMMDD_YYYYMMDD` act then as the working directories and have the same structure as it was presented in the section 2.4.

```
/root_ecmwf_mars_server
 ├─data/
 │    └─bamed/
 │      ├─campaign_n1/
 │      │   ├─20190501_20190505/
 │      │   │   ├─data1.grib
 │      │   │   ├─data2.grib
 │      │   │   └─...
 │      │   └─20190502_20190506/
 │      │       ├─data1.grib
 │      │       ├─data2.grib
 │      │       └─...
 │      └─campaign_n2/
 │          └─20200610_20200620/
 │              ├─data1.grib
 │              ├─data2.grib
 │              └─...
 └─work/
      └─bamed/
        ├─campaign_n1/
        │   ├─20190501_20190505/
        │   │   ├─bamed-config.xml
        │   │   ├─remote-job-script.sh
        │   │   └─...
        │   └─20190502_20190506/
        │       ├─bamed-config.xml
        │       ├─remote-job-script.sh
        │       └─...
        └─campaign_n2/
            └─20200610_20200620/
              ├─bamed-config.xml
              ├─remote-job-script.sh
              └─...
```

```
/root_user_remote_server
 ├─remote_data/
 │    └─bamed/
 │      ├─campaign_n1/
 │      │   ├─20190501_20190505/
 │      │   │   ├─data1.grib
 │      │   │   ├─data2.grib
 │      │   │   └─...
 │      │   └─20190502_20190506/
 │      │       ├─data1.grib
 │      │       ├─data2.grib
 │      │       └─...
 │      └─campaign_n2/
 │          └─20200610_20200620/
 │              ├─data1.grib
 │              ├─data2.grib
 │              └─...
 └─remote_work/
      └─bamed/
        ├─campaign_n1/
        │   ├─20190501_20190505/
        │   │   ├─bamed-config.xml
        │   │   ├─remote-job-script.sh
        │   │   └─...
        │   └─20190502_20190506/
        │       ├─bamed-config.xml
        │       ├─remote-job-script.sh
        │       └─...
        └─campaign_n2/
            └─20200610_20200620/
              ├─bamed-config.xml
              ├─remote-job-script.sh
              └─...
```

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

3.3 Data extraction launch

To extract the data, you should follow these steps:

1. log in to the ECMWF MARS server
2. set up parameters in your configuration file [.conf]
3. execute

```
$ path/to/the/bamed_extraction.sh --config path/to/the/ecmwf_file.conf
```

The script will launch the extraction based on your set up parameters, and then continue with the data copy and simulation launch. It is advised to launch this script as a SLURM job on the MARS server, as the data extraction can take a long time even on the computing nodes, let alone the frontal user interface nodes. The shell will also be inaccessible if the script is launched directly without submitting it as a job. To launch this script as a job, you can use the following syntax :

```
$ sbatch --wrap=”path/to/the/bamed_extraction.sh --config path/to/the/conf_file.conf
```

This command will launch a SLURM job with your configuration file and create a `.out` log file in the directory from where the command was called. You can consult the SLURM documentation [2] to see how you can customize this call and add features like job name or mail user for notifications.

In the end of the extraction and file formatting the data will be transferred to your server (if indicated), and you will receive an email notification that the job has been completed, if you enabled the mail notifications of the job [2]. The job can exit if an error occurs during the data extraction, data transfer or simulation job submission. If the simulation itself has not succeeded, you should check the simulation log output on your distant server to investigate the error.

References

[1] ECMWF MARS server filesystems
https://confluence.ecmwf.int/display/UDOC/HPC2020%3A+Filesystems

[2] SLURM sbatch command input options
https://slurm.schedmd.com/sbatch.html