Passive Tracers

Polyphemus Training Session

About

Purpose: presents how to launch a complete simulation for a passive tracer, gives the example of Chernobyl accident.

Author: Meryem Ahmed de Biasi; Updated by Pierre Tran, `tranp@cerea.enpc.fr`

Polyphemus version: 1.6

Location: `http://www.enpc.fr/cerea/polyphemus/sessions.html`

Contents

Introduction

This practical session is aimed at introducing how to perform a simulation for a passive tracer. A simplified simulation will be launched for the case of the Chernobyl release.

You are advised to extract the archive **radionuclides-x.y.tar.bz2** in your home directory where **x.y** is the version of Polyphemus to which the package is adapted.

```
$ tar -xjvf radionuclides-x.y.tar.bz2
```

This creates a directory `radionuclides/` in `polyphemus-sessions/`, which has also been created if necessary.

Place the source `Polyphemus-x.y` in `polyphemus-sessions/` also so that the paths given in configuration files are correct.

The directory **radionuclides/** is subdivided into:

- `raw_data/`: all data necessary to generate input data for the simulation,
- `data/`: input data for the simulation, output of the preprocessing,
- `config/`: all configuration files,
- `results/`: where the results are stored.

Here is the description of the domain (the same domain will be used for the preprocessing and the simulation):

- Along x:
  - minimal value: -10 °
  - step: 1.5 °
  - Number of steps: 49
- Along y:
  - minimal value: 35 °
  - step: 1.5 °
  - Number of steps: 26
- Along z:
  - 9 vertical levels
  - Interfaces of the levels (in meters): 0, 64, 236, 484, 796, 1184, 1616, 1984, 2616, 3184.

A representation of the horizontal domain and how it is meshed is given in Figure 1.

For more details on the programs used and their configuration files, please refer to Polyphemus' User Guide.

During the practical session, we will work in `polyphemus-sessions/radionuclides/`.

In what follows some command lines might be divided by \, they must be typed as a single line and are only divided for clarity's sake.

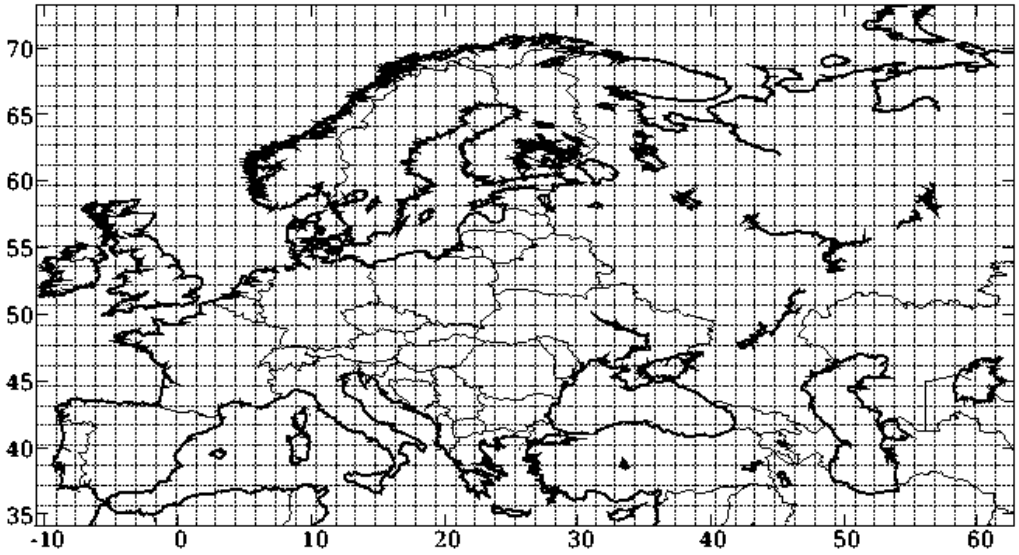

Figure 1: Domain and mesh considered for the simulation.

1 Preprocessing

The preprocessing programs needed depend on which physical processes are involved in the simulation.

Here is a list of all preprocessing programs available and an explanation on whether they will be used or not.

1.1 Ground Data

Even though ground data are not used directly in a simulation, they are usually necessary to generate meteorological data, deposition velocities and biological emissions. They should therefore be generated for most simulations. For this, two programs exist: `luc-usgs.cpp` and `luc-glcf.cpp`, depending on the origin of the raw data.

Here we will use data from the Global Land Cover Facility. We will therefore use the program `luc-glcf`.

First you need to **download the ground raw data** (from `http://glcfapp.umiacs.umd.edu:8080/esdi/index.jsp`, search for "AVHRR, Global Land Cover Product" for global region, in Lat/Long projection and at 1 km resolution, file `gl-latlong-1km-landcover.bsq.gz`), extract it and put it in `raw_data/ground`.

Compile and launch `luc-glcf` with:

```
$ cd  ../Polyphemus-x.y/preprocessing/ground/
$ ../../utils/scons.py luc-glcf
$ cd ../../../radionuclides/
$ ../Polyphemus-x.y/preprocessing/ground/luc-glcf config/general.cfg config/luc-glcf.cfg
```

Roughness file, generated by `roughness`, is only necessary when the option `Fluxes_diagnosed` in `MM5-meteo.cfg` is set to yes, which is not our case. This is why we won't generate it.

1.2 Meteorological Data

Meteorological files are provided to generate data for the simulation. You can find these two MM5 files in raw_data/meteo/.

Two programs may be used to get information on a MM5 file:

- ../Polyphemus-x.y/utils/MM5_var_list which gives general information on the file and a list of all variables stored in the file with their dimensions. Note that the date given is not the date for which data are stored in the file but the starting date of the MM5 simulation. For these meteorological files, the beginning date of the simulation is the 25th of April 1986, at 00:00 AM, yet the files only contain data from 25th of April 1986 09:00 PM.

- ../Polyphemus-x.y/utils/get_info_MM5 which gives statistical information on a variable stored in the MM5 file, for example:

```
$ cd raw_data/meteo
$ ../../../Polyphemus-x.y/utils/get_info_MM5 MM5-1986-04-25 U

Min: -37.3155
Max: 63.03
Mean: 6.34659
Std. dev.: 10.9088

$ cd ../..
```

Actually, for the simulation of Chernobyl, data are needed from the 25th of April 1986, at 09:00 PM, to the 06th of May 1986, at midnight. However, generating data for such a period is quite long and requires a lot of raw data, that is why data have already been generated and are available in data/meteo.

In order to manipulate the program MM5-meteo, we will generate data with the two files provided, that is to say the meteorological data from the 25th of April 1986, at 09:00 PM, to the 27th of April 1986, at midnight.

Please note that the end date given in command line is **excluded** and that the program can only process one MM5 file at a time.

Warning: MM5-meteo appends its results at the end of the binary files if they exist already. To make sure that the files provided in data/meteo won't be modified, make sure to save the output of MM5-meteo in directory data/meteo-test/. To do so, open config/MM5-meteo.cfg and uncomment the line "Directory_meteo: <Directory_computed_fields>/meteo-test/", that is to say, remove the # sign at the beginning of the line.

```
# Uncomment one of the following lines
Directory_meteo: <Directory_computed_fields>/meteo-test/  #For MM5-meteo
# Directory_meteo: <Directory_computed_fields>/meteo/  #For Kz_TM
```

Then launch the program:

```
$ ../Polyphemus-x.y/preprocessing/meteo/MM5-meteo config/general.cfg config/MM5-meteo.cfg \
1986-04-25_21 1986-04-26_01
$ ../Polyphemus-x.y/preprocessing/meteo/MM5-meteo config/general.cfg config/MM5-meteo.cfg \
1986-04-26_01 1986-04-27_01
```

Then we will apply program Kz_TM to the meteorological binary files in data/meteo. Make sure to modify Directory_meteo in MM5-meteo.cfg again so the value of Directory_meteo is <Directory_computed_fields>/meteo/.

```
# Uncomment one of the following lines
# Directory_meteo: <Directory_computed_fields>/meteo-test/  #For MM5-meteo
Directory_meteo: <Directory_computed_fields>/meteo/  #For Kz_TM
```

Here you do not need the MM5 files anymore, so you can launch the preprocessing for the whole period:

```
$ ../Polyphemus-x.y/preprocessing/meteo/Kz_TM config/general.cfg config/MM5-meteo.cfg \
1986-04-25_21 1986-05-06_01
```

Both programs (MM5-meteo and Kz_TM) need MM5-meteo.cfg in addition to the general configuration file general.cfg. File general.cfg mostly describes the domain considered, that is why it can be shared by all preprocessing programs, whereas MM5-meteo.cfg gives option and parameters specific to the meteorological preprocessing.

You are advised to check that Kz_TM has been used for the whole period. To do so, the simplest way is to check that data/meteo/Kz_Louis.bin and data/meteo/Kz_TM/Kz_TM.bin have the same size.

1.3 Deposition Velocities

Deposition velocities can be generated using the program dep.cpp but in this simulation we will use a constant and uniform deposition velocity for Cesium 137.

In that case, we do not need to generate a binary file. In the data file chernobyl-data.cfg, we will enter directly the constant value of the deposition velocity instead of the path to the file.

1.4 Emissions

In this simulation, we consider the Cesium 137 which has no "non-accidental" sources of emission, be it anthropogenic or biogenic. The accidental release is not generated by the preprocessing but will be modelled as a point source, given in the file config/source.dat.

1.5 Initial and Boundary Conditions

It is considered that there is no background level of Cesium137. This means that the simulation will not use initial or boundary conditions.

2 Information on a File

The preprocessing is done, we are now in possession of every file needed for the simulation.

Before we launch the simulation, it is advised to check the data to be sure an error did not occur during the preprocessing.

2.1 First Test

You can check the size of the files, for instance the meteorological files you generated .

Question: The size of a "float" is 4 bytes, then what is the size of a meteorological file generated for the whole domain and for every level, between the 25th of April 1986, 09:00 PM, and the 27th of April, 00:00 PM? Check that it is the size of the file **data/meteo-test/Temperature.bin**.

Note: All meteorological files do not have this size because they may be generated for domain which is not the one described. For example, they can be created for only the first level or for the interfaces of the vertical levels.

2.2 Second Test

The program `../Polyphemus-x.y/utils/get_info_float` allows you to have statistical information on a file, such as it maximum, minimum and mean values.

```
$ ../Polyphemus-x.y/utils/get_info_float data/meteo/Kz_TM/Kz_TM.bin

Minimum: 0.2
Maximum: 1500
Mean: 10.7975
```

2.3 Third Test

You can display the values of a binary file on a map using library `AtmoPy` (see Section 5).

3 Setting Up the Simulation

Several files are used to manage the simulation:

- `config/chernobyl.cfg`, which gives all options for the simulation and the description of the domain;
- `config/chernobyl-data.cfg`, which gives the input data;
- `config/chernobyl-saver.cfg`, which gives the options to save the results;
- `config/source.dat`, which describes the point source of Cs137;
- `config/species.dat`, which gives information on the species involved.

Open those files, check that you understand all the values given.

4 Launching the Simulation

We will first launch the simulation with the default options and data given in the configuration files.

```
$ ../Polyphemus-x.y/processing/racm/polair3d config/chernobyl.cfg
```

The simulation should take a few minutes.

Modifying Data

Now we can modify some of the data in the configuration file in order to assess the impact of those changes.

Attention: Make sure to save the results of the various simulations in different directories to be able to compare them easily. You can also change the name in the file `config/chernobyl-saver.cfg`, for instance from `results/&f.bin` to `results/&f-modified.bin`, where `&f` will be replaced by the species name.

Question 1: Modify the deposition velocity, for example divide it by 5, then multiply it by 5.

Question 2: Modify the value of the rain, for example replace it by a constant value equal to 2/3 of its maximum value:

- How can you use a constant value instead of a binary file? Remember that we used a constant value for deposition velocity.
- How can you access the maximum value of the rain in the reference simulation?

Question 3: Modify the source: quantity emitted or duration of the release. You can try and keep the total quantity emitted approximatively constant (about 8.5e16 Bq in total).

Question 4 : Change the vertical diffusion used: use Louis parameterization instead. Do you need to perform preprocessing again?

Comparing Results

Function `../Polyphemus-x.y/utils/get_diff_float` allows you to compare two binary files of the same size. You can use it to compare the results with the various configurations, for example:

```
$ ../Polyphemus-x.y/utils/get_diff_float results/Cs137.bin results-source/Cs137.bin
```

5 Displaying Results

Results will be displayed using `AtmoPy`. Refer to the user's guide on how to use and test this visualization library.

AtmoPy Library `AtmoPy` is based on Matplotlib, a Python-based plotting library with commands similar to those of Matlab.

We advise you to use IPython, an interactive python interpreter with autocompletion and an history of previous commands. It is launched with `ipython`. You can leave IPython by typing Ctrl-D.

Then you have to load modules which contain the functions you will need to display data. To load a module, use the function `import`.

```
$cd results
$ ipython
Python 2.3.5 (#2, Feb  9 2005, 00:38:15)
```

```
Type "copyright", "credits" or "license" for more information.

IPython 0.6.5 -- An enhanced Interactive Python.
?       -> Introduction to IPython's features.
%magic  -> Information about IPython's 'magic' % functions.
help    -> Python's own help system.
object? -> Details about 'object'. ?object also works, ?? prints more.

In [1]: import atmopy.display
```

`atmopy.display` is a module used to display data on a map, in particular four functions will be useful:

- `getm` which creates a map based on a domain description;
- `getd` which extracts a binary file in an array;
- `disp` which displays a 2D-array on a map;
- `dispcf` which is similar to `disp` but with filled contours.

To use the function you have to type:

```
atmopy.display.function(arguments)
```

To display a binary file, you are advised to use a configuration file, which gives, in a section "[input]":

- `Nt`: Number of time steps.
- `x_min`: Minimum abscissa of the domain.
- `Delta_x`: Space step along the x dimension.
- `Nx`: Number of space steps along the x dimension.
- `y_min`: Minimum ordinate of the domain.
- `Delta_y`: Space step along the y dimension.
- `Ny`: Number of space steps along the y dimension.
- `Nz`: Number of vertical levels.
- `file`: Path to the binary file.

For example, here such a file will be:

```
[input]
Nt = 240
x_min = -10    Delta_x = 1.5    Nx = 49
```

```
y_min = 35      Delta_y = 1.5     Ny = 26
Nz = 9
file: Cs137.bin
```

Save this configuration file as `disp.cfg` in `results/`.

Remember that "amopy.display.dispcf" is used to display a 2D-array, that is to say that you have to specify the indices (which start at 0) of the time and z coordinates, like that:

```
In [1]: import atmopy.display

In [2]: m = atmopy.display.getm('disp.cfg')

In [3]: d = atmopy.display.getd('disp.cfg')

In [4]: atmopy.display.dispcf(m, d[11, 3])
```

The output is displayed in Figure 2.

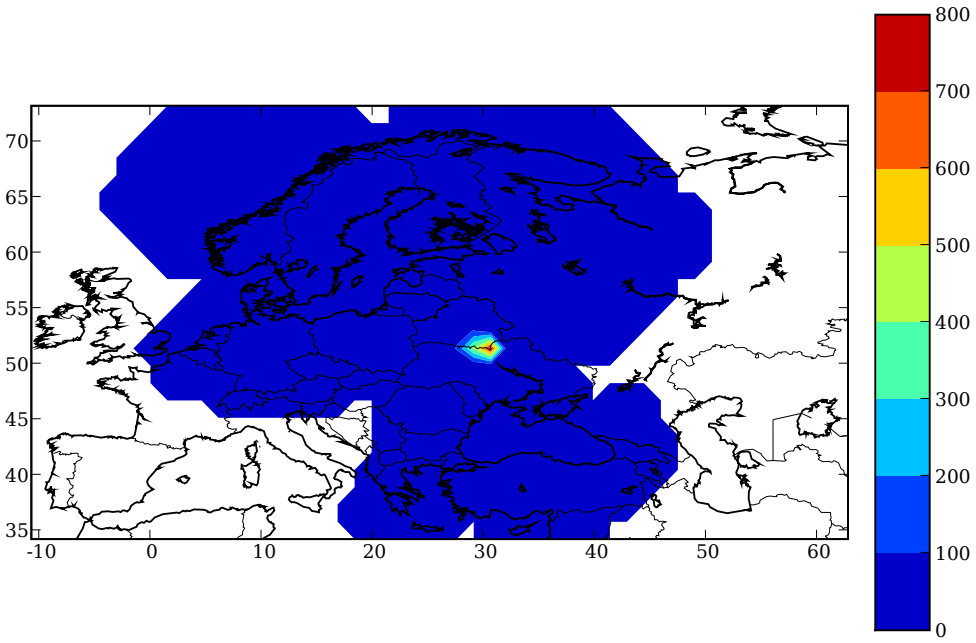

Figure 2: Concentration displayed for the 12th time step and the 4th level.

You can also use Matplotlib functions to plot something without a map, for example:

```
import pylab
pylab.plot(d[:, 0, 20, 10])
```

Here you display the variation of the radioactivity during time. The output is shown in Figure 3.

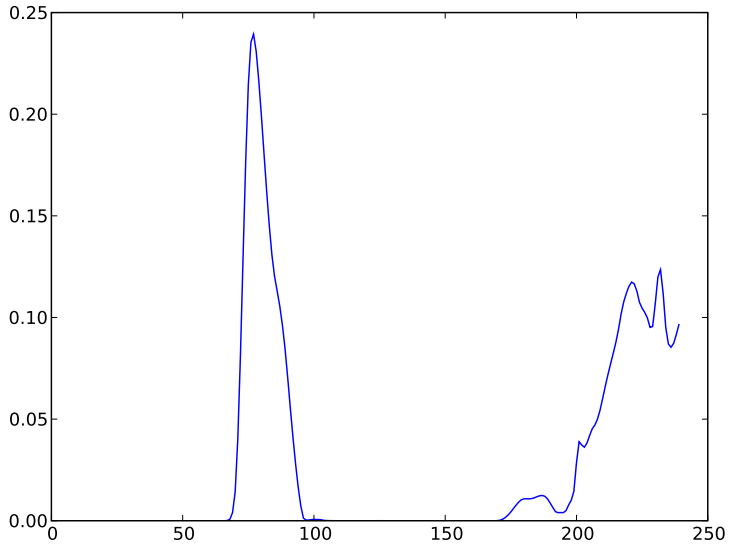

Figure 3: Variation in time of the concentration at point of indices (0, 20, 10).

Question Adapt this file to the meteorological data you generated and save it as `disp-pre.cfg`. Then use it to display the temperature on a map.

6 Going further

Question 1: Use the chemistry module `Decay`, i.e. the program `polair3d-decay` to be compiled in `Polyphemus-x.y/processing/decay/`. Once you have run the simulation, compare the results with or without decay. Note that the period of Cesium 137 is about 30 years and the simulation is launched for 10 days so there should be very little difference between the two simulations.

Question 2: Modify other things such as the position of the source. For example center it in Paris (approximately 48 ° N, 3 ° E). Then change the altitude of the source.

Question 3: Change the species emitted and give it a much shorter period (such as 15 days). Then duplicate the section "[source]" of **source.dat** in order to have two sources, either sources of different species or sources of the same species at different location.

In other words use every option of the simulation until you feel comfortable with them.