Passive Tracers – Advanced Options

Polyphemus Training Session

About

Purpose: Use the results of the first session on radionuclides to present advanced options and postprocessing abilities of Polyphemus.

Author: Meryem Ahmed de Biasi, `Meryem.Ahmed_de_Biasi@inria.fr`; Denis Quélo, `Denis.Quelo@irsn.fr`

Polyphemus version: 1.3

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

Contents

Introduction **1**

1 Models, Modules and Drivers **2**

2 Python **3**

3 Comparison to Measurements **4**

4 Plume In Grid **6**

5 Going Further **7**

Introduction

For the advanced options, we will mostly use the results and configuration file of the first session on passive tracers. It should be in `polyphemus-sessions/radionuclides/`. Additionally, Polyphemus-1.3 should be put in `polyphemus-sessions/Polyphemus-1.3/`.

Here is the description of the domain:

- 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: 0, 64, 236, 484, 796, 1184, 1616, 1984, 2616, 3184.

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

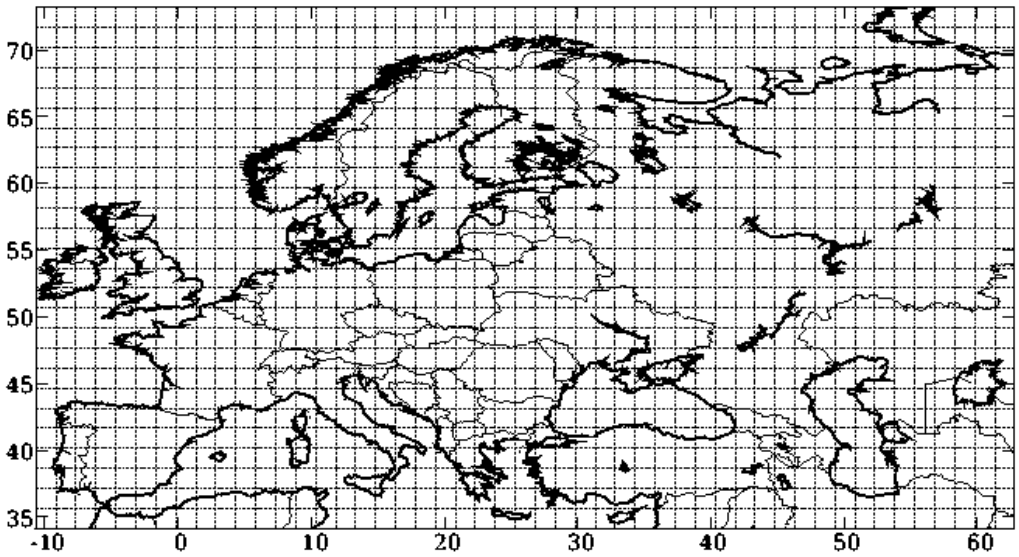

Figure 1: Domain and mesh considered for the simulation.

1 Models, Modules and Drivers

Models are C++ classes representing the various CTMs. They have several attributes, in particular a list of all gaseous species and all aerosol species (even for models without particulate species) and their concentrations. Concentrations are represented as multi-dimensional `Data` (see SeldonData documentation):

- for gaseous species, there are 4 dimensions: the species name, the space coordinates (z, y, x) (time is not involved as the data is updated at each time step and concentrations are overwritten).
- for aerosol species, there are 5 dimensions: the species name, its "bin" and the space coordinates. The *bin* represents the diameter class of the particulate matter.

Models based on Polair3D will most likely be used: `Polair3DTransport` for passive tracers, `Polair3DChemistry` which inherits Polair3DTransport and adds the use of a chemical module and `Polair3DAerosol` which inherits Polair3DChemistry and adds aerosol species.

Question : In the simulation for Chernobyl, we did not model chemical reactions, yet we used `Polair3DChemistry` with an option `With_chemistry` set to no. Modify `polair3d.cpp` to use model `Polair3DTransport`, save it as `polair3d-transport.cpp`, compile it and modify the configuration files to redo the simulation using this program.

Modules are used to implement chemical reactions or transport mechanisms.

Depending on the model used, you have a choice of compatible modules for chemistry (if necessary) and transport. See Polyphemus user's guide on the list of possible model-module(s) pairs.

Drivers are used to perform the simulation, the simplest of them just perform a forward iteration. For more advanced applications (such as data assimilation), more complicated drivers can be used.

2 Python

Python is an interpreted and object-oriented language. It is used for visualization and postprocessing in Polyphemus because of the availability of Matplotlib and its many functions used to display array.

Information on Matplotlib can be found on its web site at `http://matplotlib.sourceforge.net/`.

To learn how to program in Python, we suggest "Dive into Python", available at `http://www.diveintopython.org/`.

Here is a small script (save.py) to save images for all saved time-steps:

```
import matplotlib
matplotlib.interactive(False)


import atmopy.display
import pylab

m = atmopy.display.getm("disp.cfg")
d = atmopy.display.getd("disp.cfg")

# rang(240) creates a list of 240 integers, from 0 to 239.
for i in range(240):
    atmopy.display.dispcf(m, d[i, 0])
    if i < 10:
        i_str = "00" + str(i)
    elif i<100:
        i_str = "0" + str(i)
    else:
        i_str = str(i)
    name = "chernobyl-" + i_str
    pylab.savefig(name)
```

In Python, a loop start with a colon (":") and ends when the indentation ends. Code lines with the same indentation are supposed to be part of the same block. If a line is more indented than the previous one, the previous one must ends with a colon, otherwise there will be an error.

As in the previous session, d is a 4D array giving the concentration for each time step and each space points. You can check its dimensions with the function `shape()`.

Launch the script with:

```
$ python save.py
```

Actually, given the differences of concentration, it is better to use a logarithmic scale, which is what we will do later.

3 Comparison to Measurements

Files provided in `polyphemus-sessions/radionuclides-advanced/comparison-observation` allow you to compare the results of the simulation to actual measurements.

The measurements are extracted from REM data bank at CEC Joint Research Centre Ispra. Download the file `CHEMAIR.txt` at `http://rem.jrc.cec.eu.int/dataset.html` and put it in `comparison-observation/raw_data/measurements/`.

In `CHEMAIR.txt` the following information is given:

- country code
- locality name
- latitude, longitude [degrees]
- date, hour of end of sampling
- duration
- concentration of I-131, Cs-134 and Cs-137 (aerosol particles) [$\mathrm{Bq\,m^{-3}}$]

Apply to `CHEMAIR.txt` the script `format.py` in order to have the measurements in the right format:

```
$ cd comparison-observation/raw_data/measurements
$ python format.py
```

The map of the 88 measurement stations is given in Figure 3.

Comparison with observations is performed with AtmoPy. Details are available in AtmoPy documentation (`http://www.enpc.fr/cerea/polyphemus/doc/atmopy/index.html`).

Question 1 : Display a map of Cs-137. In this first example, the objective is to use some basic functions of AtmoPy: `getm` and `getd` with the configuration file `config/chernobyl.cfg` which allows to draw quickly a map of concentrations.

Script postreat.py

In the following, the script `postreat.py` will be used. It provides some functionality to perform model-to-data comparisons. 5 options are available :

1. bars at stations.

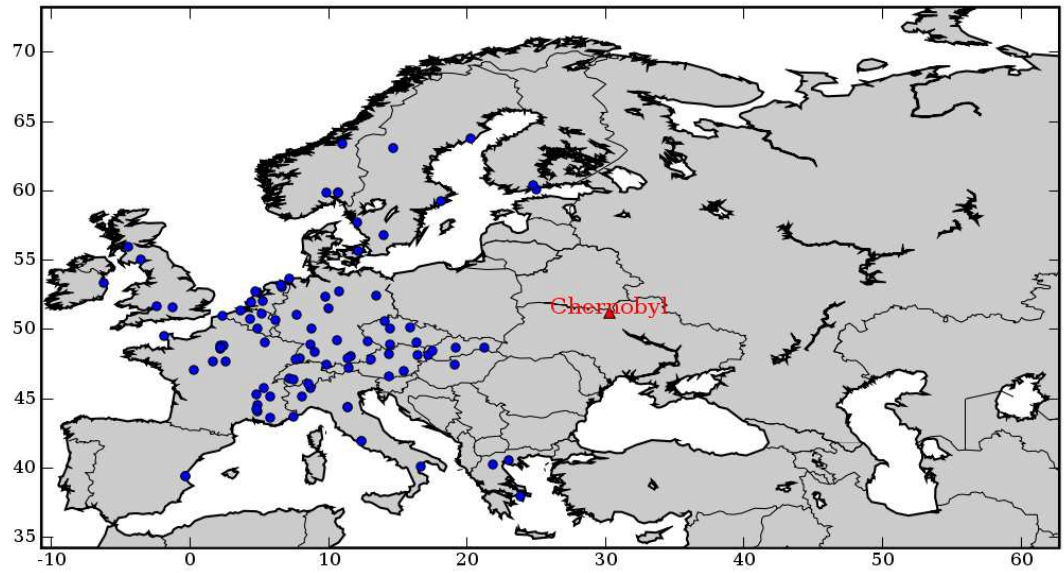

Figure 2: Chernobyl accident: maps of stations.

2. plume evolution maps.
3. station map.
4. statistics.
5. scatter plot of the concentrations (log scale).

Go to `polyphemus-sessions/radionuclides_advanced/comparison-observation`.

```
$ cd ../../
```

Launch `postreat.py` with Python shell and the configuration file `config/chernobyl.cfg`. The results will be written in the directory `results/`.

Question 2 : Station map
Select option 3 in `chernobyl.cfg`. It gives a map of all stations for which measurements are available.

Question 3 : Plume evolution maps
Select option 2 in `chernobyl.cfg`. Maps of the computed surface concentration are drawn.

Question 4 : Bars at stations
Select option 1 in `chernobyl.cfg`.

Question 5 : Scatter plot of the concentrations (log scale)
Select option 5 in `chernobyl.cfg`. The scatter plot for activity concentrations in logarithmic scale are drawn.

Question 6 : Statistics
Some statistical indicators may be computed. Select option 4 in `chernobyl.cfg`.

The figures created will not be displayed during computation because the mode of Matplotlib has been set to not-interactive. If you want to see what happens, open postreat.py and change "matplotlib.interactive(False)" in "matplotlib.interactive(True)". Note however that the script will take much longer in interactive mode.

4 Plume In Grid

Go to `polyphemus-sessions/radionuclides_advanced/plume-in-grid`:

```
$ cd ../plume-in-grid/
```

"Plume in Grid" is based on the fact that point emissions in Eulerian models are assumed to be instantaneously diluted into the cell, which is quite a crude assumption, especially when using coarse grids. The plume-in-grid method consists in first treating a point emission with a Gaussian model (a puff model in our case) and, when a puff has reached the Eulerian cell size, in feeding it back to the Eulerian model. Therefore, the puff is much less diluted. In this part, we will see how this applies to the simulation of the Chernobyl case.

Compile `../../Polyphemus-1.3/driver/plume-in-grid.cpp`.

Program `plume-in-grid` requires several configuration and data files, provided in `polyphemus-sessions/radionuclides-advanced/plume-in-grid/config`:

- `chernobyl-ping.cfg`, which gives the option for the Eulerian model. A few options are added to take into account the coupling of the Eulerian and Gaussian models.
- `chernobyl-data-ping.cfg`, which gives data for the Eulerian model. The meteorological data provided is also used for the Gaussian model. Additional data, found in section `[gaussian_meteo]` is needed to compute the Pasquill stability class: `LowCloudiness`, `MediumCloudiness`, `HighCloudiness`, `SolarRadiation` and `FirstLevelWindModule`. Pasquill stability class is needed for Briggs sigma parameterization, but if you have not generated those meteorological files, you can use Doury sigma parameterization which does not need them and put any constant value in the section `[gaussian_meteo]`.
- `chernobyl-saver.cfg`, which is identical to the one used during the first session for Eulerian model. Please refer to Polyphemus user's guide (section on Drivers and subsection on OutputSavers) for more details.
- `puff.cfg`, which gives information on the puff model. Actually, some values given in this file are ignored and are actually inferred from the Eulerian configuration file.
- `puff-saver.cfg`, this file is not used for the plume in grid but the Gaussian puff model requires that it is read. You can put anything, for example "—" in `puff.cfg`.
- `levels.dat`, levels file used for the Eulerian model. The Gaussian model also requires a levels file which is not read, so you can use the same file in both cases.
- `source.dat`, the source used by PlumeInGridDriver, it is used as a source in the Gaussian model and propagated in the Eulerian model. It should not be used as a point source in the Eulerian model, otherwise it would be as if it was emitted twice. Make sure that option `With_point_emission` is set to "no" in `chernobyl-ping.cfg`.

- species.dat, species used. You can use different files for the Eulerian and Gaussian models, for instance if you use fewer species in the Gaussian model, but you are advised against it.

The most important information given in puff.cfg is the time step Delta_t. If it is larger than the time step for the Eulerian model, it is considered equal to it. Otherwise, a number N of iterations for the Gaussian model are performed at each iteration of the Eulerian model, where:

$$N = int \left[\frac{Delta\_t\_Eulerian}{Delta\_t\_Gaussian} \right] \quad (1)$$

You are strongly advised to use a time step for the Gaussian model smaller than the one for the Eulerian model in order to perform several iterations of the Gaussian model.

Note that, currently, scavenging and dry deposition are not implemented for the Gaussian model in Plume in Grid, set the options With_scavenging and With_dry_deposition to no in puff.cfg.

Launch plume-in-grid with:

```
$ ../../Polyphemus-1.3/driver/plume-in-grid config/chernobyl-ping.cfg
```

You can then compare results with or without plume-in-grid. You can also compare the results with measurements, thanks to postreat.py and chernobyl-ping.cfg.

5 Going Further

Add other species for the simulation: Iodine 131 (period of 8 days) and Caesium 134 (period of 2 years).

First use the same rate and length for the sources of all species, then change the sources.

Compare results with or without Decay.