Postprocess

Polyphemus Training Session

About

Purpose: introduction to one-way nesting, advanced visualization and comparison to measurement data; application to photochemistry and aerosol.

Author: Yelva Roustan, `roustan@cerea.enpc.fr`

Polyphemus version: 1.5

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

Contents

Introduction

This document gather several items concerning postprocessing of simulation results. They can be treated separately but many of them are strongly linked to the use of the AtmoPy Polyphemus module which is based on Python language and libraries. For this reason some "basic" knowledge can be found in the Polyphemus User's Guide (part **A Very Short Introduction to Python and Matplotlib** and **Visualization with AtmoPy**). This document is divided in three parts:

- The first one is devoted to the model/measurement comparison.
- The second one deals with one-way nesting.
- The last one show some possibilities to visualize simulation results through the AtmoPy module.

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.

1 Comparison to measurements

1.1 Simulation results

Two choices are available to get simulated data:

- Use results of the first session (photochemistry and aerosol simulation).
- Quickly generate similar results by using the configuration given in

  `aerosol_postprocess/config-nesting/` (Specificity of this simulation will be explained in the second part of this session). Note that in this case you will have to copy the `raw_data` directory from the photochemistry and aerosol session.

All the configuration files proposed in this session are written to be used from the directory `~/polyphemus-session/aerosol_postprocess/` with Polyphemus source code in `~/polyphemus-session/Polyphemus-1.5`.

1.2 Aerosol specific post-treatment

Until now aggregated indicators as PM_{10} or the total sulfate aerosol mass are not computed during the simulation but in the postprocessing phase. This first step consist in using the simulation results to get the desired indicators. It can be done by the Python script `init_aerosol.py` available in the directory `../Polyphemus-1.5/postprocessing/` which reads a configuration file as the one you can find in the configuration directory (`aerosol_postprocess/config-postprocess/init_aerosol.cfg`).

Browsing the file allow you to see three sections:

- `[input]`; parameters related to the exploited data.
- `[output]`; parameters related to the aggregated indicators to compute.
- `[directory_list]`; directory for inputs **and** outputs.

```
[input]

Nt =    integer, number of saved time interval in the result files.
Nz =    integer, number of saved vertical levels.
Ny =    integer, number of saved cells along the latitude.
Nx =    integer, number of saved cells along the longitude.

### Bins definition

# Number of bins.
Nbins =    integer, number of bins used for the simulation.
# Are diameters computed?
computed =    yes or no.
```

```
# If diameters are computed, provide minimum and maximum diameters.
Dmin =   float, minimum diameter given in µm.
Dmax =   float, maximum diameter given in µm.
# If diameters are not computed, put the name of the file
# containing the bounds diameters. file_bounds:   string, path to the file where diameters are given in µm.
# Shift in bin indexing
# if first bin is indexed by 0, put 0
# if first bin is indexed by 1, put 1
bin_index_shift =  0 or 1.

[output]

### Aerosols initialization

# Aerosol species

# 1 - Species names in the chemical mechanism.
primary:   string list, giving the names of the files (without the extension ".bin") to be read.
inorganics:   string list, the same.
organics:   string list, the same.

# Output to be computed.
# Options: PM10, PM2.5, total_species, total_bins, granulometry
output_species:   string list, giving which aggregated indicators are computed among the proposed ones.

# Take organics into account?
with_organics:   yes or no.

[directory_list]

# List of directories.
string, path to the directory where files are read and written.
```

The comand line is:

```
$ python ../Polyphemus-1.5/postprocessing/init_aerosol.py \
config-postprocess/init_aerosol.cfg
```

1.3 Comparison statistics

Measurement data used here are provided on an hourly basis by the EMEP database[1] for ozone and the AirBase database[2] for PM_{10} and $PM_{2.5}$. Comparison statistics can be computed by `../Polyphemus-1.5/postprocessing/evaluation.py` used with an appropriate configuration file like `config-postprocess/evaluation.cfg`.

It is divided in two sections:

[1] `http://www.nilu.no/projects/ccc/emepdata.html`

[2] `http://air-climate.eionet.europa.eu/databases/airbase/`

- `[input]`; parameters related to the exploited data.
- `[output]`; parameters related to the computed statistical indicators.

```
[input]

file:   string, path to the treated result file.

t_min = date  Delta_t = float  Nt = integer, initial date (YYYYMMDDHH), time step (in
hour) and number of saved time interval in the result files.
x_min = float  Delta_x = float  Nx = integer, longitude of the lower left cell center of the
saved domain (in °), space step (in °), number of saved cell along the longitude.
y_min = float  Delta_y = float  Ny = integer, the same for latitude.
Nz =  integer, number of saved vertical levels.
station_file:   string, path to the station file list.
# The way the station file is written.
station_file_type:  string, emep, airbase, etc...
obs_dir:   string, path to the directory where observation data are read.

[output]

### Observations

station:   string, station code if only one station is studied (see "Statistical measures").

t_range:   date date, beginning and end of the comparison period.

# Which concentrations: hourly / peak / daily?
concentrations:   string, giving which concentration is considered for the comparison.

# In case the peaks are chosen, should they be paired?
paired:   yes or no.

# In case daily concentrations are chosen, are observations
# provided on a daily basis?
daily_basis:   yes or no.

### Statistical measures

# Which stations (statistical measures)?
# single: the station defined with field 'station'
# all: all stations from the station file
# {field} {value}: stations whose attribute {field} is {value}
select_station:   string

# List of statistical measures.
# Available measures (see atmopy.stat):
# meas_mean: measurements (or reference simulation) mean
# sim_mean: simulation mean
```

```
# MBE: mean bias error
# MNBE: mean normalized bias error
# MNB: normalized mean bias
# BF: bias factor
# NME: normalized mean error
# MFBE: mean fractionalized bias error
# MAGE: mean absolute gross error
# MNGE: mean absolute normalized gross error
# FGE: fractional gross error
# RMSE: root mean square error
# correlation: correlation coefficient
# determination: coefficient of determination
# all: all measures
measure:   string list, giving which statistical indicators are computed among the proposed ones.

cutoff:   float, observed concentration value below which the comparison is not taken into account
for the computation of some statistical indicators.

# Minimum ratio of the number of available observations and
# the number of simulated concentrations?
ratio:   float, between 0 and 1, data to a given station are used if the ratio between the number
of observation actually available and the maximum potentially available over the period are above the
chosen ratio.

# When computing statistical measures, the output should be
# (multiple output allowed):
# summary: the means of measures over all stations
# station_names: the name of the stations (or simulations)
# all_stats: the statistical measures at all stations are displayed
#            (then 'station_names' is automatically set)
# file: writes results in a file whose name follows the keyword 'file'
output:   string list.
```

The comand line is:

```
$ python ../Polyphemus-1.5/postprocessing/evaluation.py \
config-postprocess/evaluation.cfg
```

2 One-way Nesting

Nested simulations consist in two simulations. The fist one is only used to create boundary conditions for the second one. As a convention, the first simulation will be denoted with "nesting" and the second one with "nested".

2.1 First Simulation

The configuration of the nesting simulation can be found in `config-nesting/`. The only difference (except for the writing and reading paths) between this simulation configuration and the one used for photochemistry and aerosol simulation lies in the use of additional saver:

- `SaverUnitNesting`
- `SaverUnitNesting_aer`

More informations about these savers can be found in the User's Guide, paragraph **Output Savers**. The aim of the nesting savers is to generate boundary conditions for the nested simulations.

Compare savers configuration file. The use of additional savers need additional sections in the configuration file:

```
[save]

# Put "all" to output all species.
Species:   string list, species name.

Date_beg:   date or -1   # Put -1 to start from the simulation initial date.
Date_end:   date or -1   # Put -1 to end at the simulation final date.
Interval_length:  integer   # 1 for all steps.

# Choices: domain, domain_aer, domain_assimilation, nesting, nesting_aer.
Type:   string, the chosen type of saver.

x_min = float   Delta_x = float   Nx = integer, longitude of the lower left cell center of the
nested domain (in °), space step (in °), number of saved cell along the longitude.
y_min = float   Delta_y = float   Ny = integer, the same for latitude.
Nz =   integer, number of vertical levels in the nested domain.
Vertical_levels:   string, path to the file where vertical levels are given.

Output_file:   string, path to the file where results are stored.
```

The commands to launch, and their order (at least for the first steps), can be found in file `config-nesting/commands.txt`.

2.2 Nested simulation

The configuration of the nested simulation can be found in `config-nested/`. It uses the boundary conditions previously generated by the nesting simulation and a set of suitable data generated by the classical preprocessing tools.

To ensure the coherence of the simulation results, it should be checked that the data provided in the savers of the nesting simulation correspond to those used for the preprocessing of the nested simulation. *Compare configuration of both simulations.*

The commands to launch, and their order, can be found in file `config-nested/commands.txt`.

3 Visualization

A first experience in using the Python library AtmoPy is proposed at the paragraph **Post-processing** of the Polyphemus User's Guide. If this has not been done before, it is strongly recommended to read this paragraph now and proceed the proposed examples. A documentation is available online[3].

[3] `http://www.enpc.fr/cerea/polyphemus/atmopy.html`

3.1 Using online command

The use of online commands should be limited to rather "simple" examples as those presented in the Polyphemus User's Guide. *Use files* ***disp.cfg*** *that you can find in the directories* ***∼/aerosol_postprocess/config-nesting/*** *and* ***∼/aerosol_postprocess/config-nested/*** *to visualize a concentration field.*

Apply the commands given in the guide to your results.

3.2 Using scripts

Two Python script examples are used in the paragraph 1, `evaluation.py` and `init_aerosol.py`. Two other examples, `disp.py` and `graph_aerosol.py`, dedicated to the visualization can be found in the same directory (∼`/Polyphemus-1.5/postprocessing`). Both of them use the configuration file `disp_aerosol.cfg` in ∼`/config-postprocess`. As some needed data have already been presented in previous paragraph for the use of `evaluation.py`, we will refer to it in the following.

```
[input]

file:  string, path to the treated result file.
# Or
multiple_files:  yes or no, if yes paths to the treated result files are given at the section
[file_list].
t_min =    Delta_t =    Nt =  see evaluation.cfg
x_min =    Delta_x =    Nx =  see evaluation.cfg
y_min =    Delta_y =    Ny =  see evaluation.cfg
Nz =     see evaluation.cfg

station_file:   see evaluation.cfg
# The way the station file is written.
station_file_type:   see evaluation.cfg
obs_dir:  see evaluation.cfg

[output]

### Observations

station:   see evaluation.cfg

t_range:   date date, beginning and end of the display period.

# Which concentrations: hourly / peak / daily?
concentrations:   see evaluation.cfg

# In case the peaks are chosen, should they be paired?
paired:   yes or no.

# In case daily concentrations are chosen, are observations
# provided on a daily basis?
```

```
daily_basis:  yes or no.

### Graphics

# Adjust the limits y_min and y_max along y (and along x for scatter plots).
# If only one number is provided, the limits are automatically set.
y_range:   float float, beginning and end of the displayed y axis.

# Scatter plot? Options:
# 0: No
# 1: Yes
# 2: Yes, with equal axes
# 3: Yes, with equal axes and lines
# Note: the drawing options are set with 'meas_style'.
scatter:   0, 1, 2 or 3.

# You may provide a style (--, -, d, etc.) and a linewidth for measurements.
# Note: (1) The linewidth may be omitted (default: 0.5).
#       (2) Put '..' instead of ':'.
meas_style:   Following the above explanations.

# Styles (and linewidths) for simulated concentrations, separated by '\&'.
# If there is less styles (and linewidths) than simulations,
# only the first style is applied.
# Note: A single style leads to a single entry in the legend.
sim_style:   Following the above explanations.

# For aerosols graphs (observation not supported)

# Expected graphs?
# pie_chart: pie chart of aerosol chemical composition
# bar_chart: bar chart of aerosol chemical composition
# temporal: temporal variation of chemical composition
# mass_distribution: PM mass distribution along size spectrum
# number_distribution: PM Number distribution along size spectrum
#                      (granulometry)
graph_type:   string list, keywords among the proposed ones.

# Aerosols graphs computed at station (yes) or over a domain (no)?
graphs_at_station:   yes or no, if yes graph is displayed for the previously chosen station.

# If graphs are computed over a domain, specify the horizontal domain
# (indices) over which data is averaged.
i_range:  float float   # range for longitudes indices
j_range:  float float   # range for latitudes indices

# Are mass- and number-distribution displayed with log scale for diameters?
log_plot:   yes or no.
```

```
[file_list]

# List of files used if 'multiple_files' is set.
```
string, *path to the treated result file, one line by file.*

```
[legend] # Associated with the listed files.
```
string, *graph legend, one line by file.*