

Local scale

Polyphemus Training Session

About

Purpose: Local scale simulations: introduction to the use of Gaussian models with Polyphemus.

Author: Irène Korsakissok, Irene.korsakissok@cerea.enpc.fr

Polyphemus version: 1.3

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

Contents

Introduction	1
1 Test cases description	2
2 Deposition preprocessing	3
2.1 Description	3
2.2 Using the program	4
3 Discretization	5
3.1 Description	5
3.2 Using the program	6
4 Running simulations	6
4.1 Plume simulation with gaseous species	6
4.1.1 Simulation	6
4.1.2 Visualizing results	8
4.2 Puff simulation with aerosol species	10
4.2.1 Simulation	10
4.2.2 Visualizing results	11
4.3 Puff simulation with line sources	11
4.3.1 Simulation	11
4.3.2 Visualizing results	12

Installing the Session

You are advised to extract the archive **gaussian.tar.bz2** in your home directory.

```
$ tar -xjvf gaussian.tar.bz2
```

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

Place the source `Polyphemus-1.3` in `polyphemus-sessions/` also so that the paths given in configuration files are correct.

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

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.

Introduction

Gaussian diffusion models are widely used in regulatory applications because they are analytical and conceptually appealing, and computationally cheaper to use. Basically, it consists in assuming a Gaussian distribution of mean concentration in the horizontal and vertical directions at any downwind location from the source. The Gaussian plume model gives the analytical expression of the concentration in the case of a continuous emission from the source at a constant rate, with a stationarity hypothesis. The Gaussian puff model gives the concentration for an instantaneous point source at different times.

The aim of this session is to learn the basic use of Polyphemus Gaussian models, thanks to three test cases. Polyphemus contains two types of Gaussian models: `GaussianPlume` which is a stationary Gaussian model, that is, for continuous sources, and `GaussianPuff` which is a puff model. In addition, `GaussianPlume_aer` and `GaussianPuff_aer` are the corresponding models for aerosol species.

Basically, you only need few information to run a Gaussian model. If there are no scavenging, deposition, or radioactive decay, you simply have to enter the name of the species. A meteorological data file provides some meteorological information (wind, temperature, stability) that are supposed to be constant and homogeneous. You also need to provide the source data. Sources are supposed to be points emitting either continuously or instantaneously one species. Once you have created all your configuration files, you just have to compile and launch the corresponding program: `plume.cpp`, for plume model with gaseous species, `puff.cpp` for puff model with gaseous species, or the corresponding programs for aerosol species `plume_aer.cpp` and `puff_aer.cpp`.

Two preprocessing programs are available:

- In case there are scavenging or deposition, `gaussian-deposition` must be used prior to the simulation to compute scavenging coefficients and deposition velocities.
- In case there is a line source instead of a point source, `discretization` must be used to discretize it into a series of point sources.

1 Test cases description

The first test case is a simple case using `plume` with gaseous species emitted by point sources, the second one uses `plume_aer` with aerosol species, and the third one uses `puff` with gaseous species emitted by a line source. The test cases are described in Polyphemus guide (part **Polyphemus Gaussian Test-Case**). However, this training session goes further than the basic test case

description.

The directory `polyphemus-sessions/gaussian/` contains:

- A subdirectory `config/` which holds all the configuration files used in the preprocessing and the simulation.
- A subdirectory `results/` which is meant to hold the results of all simulations. It is also divided in three subdirectories (one for each kind of simulation):
 - `puff_line/` for the Gaussian puff model and a gaseous line source.
 - `puff_aer/` for the puff model with point sources of gaseous and aerosol species.
 - `plume/` for the Gaussian plume model with gaseous species only.

This session is divided into three parts:

1. Using preprocessing programs for Gaussian models.
2. Launching the three simulations.
3. Visualizing results.

2 Deposition preprocessing

2.1 Description

In case loss processes are taken into account, you need to compute scavenging coefficients and deposition velocities for the various species. This is achieved by using `gaussian-deposition_aer`. This program uses one configuration file, `gaussian-deposition_aer.cfg`, which contains all necessary information.

- `[data]` contains paths to other data files containing species data and meteorological data
- `[scavenging]` is used to choose the parameterization to compute scavenging coefficients for gaseous and aerosol species
- `[deposition]` is used to choose the parameterization to compute deposition velocities for gaseous and aerosol species
- `[output]` contains the path to the file where output data will be written.

For a more detailed description, see Polyphemus Guide, part **Preprocessing/Preprocessing for Gaussian models/Programs `gaussian-deposition` and `gaussian-deposition_aer`**.

Diameter file. The file `diameter.dat` contains aerosol diameters. This applies to all aerosol species. For example, if **PM** is an aerosol species, and there are two diameters in the diameter file:

```
[diameter]

# Particle diameter (micrometer)
0.1
5.0
```

the simulation will be run with **PM** particles of the two diameters. Scavenging coefficients, deposition velocities, (as well as concentrations in the main program) will be computed as if there were two different species: **PM_0**, which refers to **PM** with the first diameter in the diameter file, that is, 0.1 μm , and **PM_1**, which refers to **PM** with the second diameter in the diameter file, that is, 5 μm .

Note that there is no need to define diameters in a special file. It can as well be defined inside a section `[diameter]` in the main configuration file. The reason it is defined in a separated file here is that it is common to the preprocessing program and to the main program.

Species file. The file `gaussian-species_aer.dat` contains data about the species.

- `[species]` and `[aerosol_species]` contain the list of gaseous and aerosol species respectively
- `[scavenging]` and `[deposition]` contain the list of gasous species for which scavenging and deposition occur. For others, coefficients are set to 0. For aerosol species, scavenging and deposition always occur.
- `[scavenging_constant]` and `[deposition_constant]` contain values of scavenging coefficients and deposition velocities for gaseous species when constant values are used. One value is needed for each species.
- `[scavenging_constant_aer]` and `[deposition_constant_aer]` contain values of scavenging coefficients and deposition velocities for aerosol species when constant values are used. One value for each aerosol diameter.

Other data can be added if needed. For a more detailed description, see Polyphemus Guide, part **Preprocessing/Preprocessing for Gaussian models/Programs gaussian-deposition and gaussian-deposition_aer** and also part **Models/GaussianPlume**.

Meteorological data file. The file `meteo.dat` contains meteorological data. For each meteorological situation, temperature, wind angle, wind speed, boundary height and stability class are needed. These information are provided inside a section `[situation]`, and the meteorological data file contains as many sections as there are situations. Note that other information might be provided, depending on the parameterization to compute the standard deviations.

For a more detailed description, see Polyphemus Guide, part **Preprocessing/Preprocessing for Gaussian models/Programs gaussian-deposition and gaussian-deposition_aer**.

2.2 Using the program

First compile the preprocessing program:

```
cd ../../Polyphemus-1.3/preprocessing/dep/  
make gaussian-deposition_aer
```

Then run it from the configuration files directory `~/polyphemus-sessions/gaussian/config/`.

```
../../Polyphemus-1.3/preprocessing/dep/gaussian-deposition_aer \  
gaussian-deposition_aer.cfg
```

The output on screen will be :

```
Reading configuration file... done.  
Reading meteorological data... done.  
Reading species... done.  
Reading diameter... done.  
Computation of the scavenging coefficients... done.  
Computation of the deposition velocities..done.  
Writing data... done.
```

The file `gaussian-meteo-aer.dat` has been created and placed in the configuration files directory `polyphemus-sessions/gaussian/config/`. It will be used for all simulations. Open it to see if it has been correctly written: a list of scavenging coefficients and deposition velocities must have been added at the end of each section [situation].

Note: For simulations where only gaseous species are taken into account, the program `gaussian-deposition` could be used instead of `gaussian-deposition_aer`.

Run now the program in order to have constant scavenging coefficients equal to $1. \times 10^{-4} \text{ s}^{-1}$ for aerosol species and for Iodine and no scavenging for Caesium, and constant deposition velocities equal to $0.5 \times 10^{-2} \text{ ms}^{-1}$ for all species.

3 Discretization

3.1 Description

This step is only necessary for the simulation with a line source. Its aim is to discretize this source into a series of point sources. There are two possible types of line sources:

1. An instantaneous line source is discretized into a given number N_p of point sources that all are emitted at the same time.
2. A moving source (typically an aircraft or a car). Given the source trajectory, its velocity, and its starting time, one can represent it by a series of puffs emitted along the trajectory at different emission times. The number of points on the trajectory is determined by the time step between two puffs Δt .

In addition, the non-moving line source, as point sources, can be a continuous source (for example modeling a road traffic) or a puff source. It will then be discretized either into a series of puffs or into a series of plumes.

The program `discretization` uses one configuration file, `discretization.cfg`, which contains all necessary information.

- `[trajectory]` contains the path to a file which gives the trajectory. It also contains N_p (for instantaneous sources) or Δt (for moving sources).
- `[source]` contains basic source data: the type (puff or continuous) and the emitted species.
- `[puff-source]` contains puff data: quantity (divided into the number of points on the trajectory), source velocity, initial release time.
- `[plume-source]` contains plume source rate.
- `[output]` contains the path to the file where output data will be written.

Note that when the source velocity is not equal to 0., it is assumed to be a moving source and N_p is not read. Also note that N_p is the number of points to be computed, so the final number of points on the trajectory is equal to N_p plus the points already given in the trajectory file.

The file defining the trajectory looks like this:

```
#X(m) Y(m) Z(m)
0. 0. 30.
30. 40. 50.
```

Each line corresponds to the coordinates (abscissa, ordinate, height) of a point on the trajectory. Hence, this contains at least two points, but there can be more, defining several segments.

For a more detailed description of the files, see Polyphemus Guide, part **Preprocessing/Preprocessing for Gaussian models/Program discretization**.

3.2 Using the program

The first line to be discretized is an instantaneous line source. The trajectory is defined in file `line-emission.dat`. Change the main configuration file to discretize it into 28 points, including those defined in the file `line-emission.dat`.

Compile the preprocessing program `discretization`.

```
cd ../../Polyphemus-1.3/preprocessing/emissions/
make discretization
```

Then run it from the test case directory `polyphemus-sessions/gaussian/config/`.

```
../../Polyphemus-1.3/preprocessing/emissions/discretization \
discretization.cfg
```

The output on screen will be:

```
Reading configuration file... done.
Reading trajectory data... done.
Length of the trajectory: 48.0278
Number of points on the trajectory: 28
Writing source data... done.
```

The file `puff-discretized.dat` has been created and placed in the configuration files directory `~/polyphemus-sessions/gaussian/config/`. It contains a series of puffs representing the discretized line source.

4 Running simulations

4.1 Plume simulation with gaseous species

4.1.1 Simulation

This simulation uses the program `plume.cpp` along with the configuration file `plume.cfg`. This file contains the following information:

- `[display]` contains options about what should be displayed on screen during simulation.
- `[domain]` contains the cartesian domain where concentrations are to be computed, path to the species file and miscellaneous information.
- `[gaussian]` contains all options that can be activated: plume rise, decay, scavenging and dry deposition. It also contains paths to meteorological data file and source file.
- `[deposition]` contains the deposition model.
- `[output]` contains path to the output configuration file.

The other data files that are used here are:

- `gaussian-levels.dat` contains vertical levels where concentrations are computed.
- `gaussian-species_aer.dat` contains meteorological data.
- `gaussian-meteo_aer.dat` gives all meteorological data, scavenging coefficients and deposition velocities. It was created during preprocessing.
- `plume-source.dat` contains source data.
- `plume-saver.cfg` contains paths to results files.

For detailed description, see Polyphemus user's Guide (part **Models/GaussianPlume**).

Compile `plume.cpp` (in `../..../Polyphemus-1.3/driver/`) and run the plume simulation. Display meteorological data on screen during simulation. Only scavenging is taken into account. There is no plume rise. The output on screen will be:

	Temperature	Wind angle	Wind velocity	Stability
Case #0	15	-100	0.5	D
Case #1	10	-5	2	D
Case #2	10	20	5	D
Case #3	10	60	10	D

A quick way to make sure that the simulation has been correctly done is to use the program `get_info_float` (located in `../Polyphemus-1.3/utils/`).

Go in the directory where your results file has been placed. The file `Iodine.bin` contains concentrations for the species Iodine. The command `get_info_float Iodine.bin` returns something like:

```

Minimum: 0
Maximum: 0.486352
Mean: 0.0142579

```

For more information about this tool, see Polyphemus user's Guide (part **Using Polyphemus/Useful tools**).

Now run again the program. This time, use also radioactive decay and dry deposition with Overcamp model. Change the results file name so that the previous one won't be overwritten. Then use `get_diff_float` to compare the two results. You should obtain something like:

```

          File #0      File #1
Minima:      0          0
Maxima:      0.486352  0.486233
Means:       0.0142579 0.0136535
Standard dev.: 0.0540338 0.0517169

          Difference
Minimum:     0
Maximum:     0.0565844
Mean:        0.000604366
Standard dev.: 0.00383071

Correlation between files #0 and #1: 0.998335

```

This shows that radioactive decay and dry deposition are taken into account and that the concentration is slightly lower, but in this example it is negligible.

4.1.2 Visualizing results

To easily visualize the results of a simulation, it is recommended to use the interactive python interpreter IPython (launched with the command `ipython`). For more detailed information about how to launch and use ipython, see Polyphemus user's Guide (part **Postprocessing/Visualizing results**).

Launch IPython from the plume results directory. Then, import the modules that are needed for results visualization with the command:

```
>> import atmopy
>> from atmopy.display import *
```

Then, you need to import the concentration field you want to visualize:

```
>> d = getd(filename = 'Iodine.bin', Nt=4, Nz=2, Ny=200, Nx=200)
```

Nt is normally the number of time steps. Here, as it is a stationary simulation, it should be equal to 1. However, as there are four meteorological situations, we have here $Nt = 4$, as each situation is similar to a time step for an unstationary simulation. It would be the same if it was an unstationary simulation (puff model) with several meteorological situations. If there are 10 time steps, and 4 meteorological situations, you will put $Nt = 40$. The first ten time steps represent the first situation, from 10 to 20 you have the concentration field for the second situation, and so on...

Nz is the number of vertical levels, Ny and Nx the number of points in y and x directions. All numbers must be consistent with the simulation domain. If you are not sure, or if the command doesn't work, check the size of the binary file you are trying to visualize. It should be of size $Nt \times Nz \times Ny \times Nx \times 4$.

You now have an array named 'd' that contains the concentration field for Iodine. To check it has the right shape, use the command:

```
>> d.shape
(4, 2, 200, 200)
```

This returns (Nt, Nz, Ny, Nx) .

Now, visualize domain concentration for the first situation and add a colorbar:

```
>> contourf(d[0,0])
>> colorbar()
```

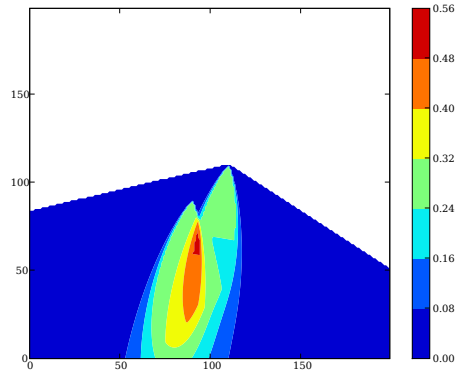


Figure 1: Plume visualization for the first meteorological situation.

You should obtain the Figure 1.

Visualize the concentration field for the other meteorological situations. You should see that the wind is turning with increasing speed.

To visualize the plume evolution according to meteorological situations, you can use atmopy function `stat.spatial_distribution`. This function applies a given function to a given field at each time step. Hence, if you type the command:

```
>> s = atmopy.stat.spatial_distribution(d,"max")
>> contourf(s[0])
```

You should see the spatial distribution of maximum concentration at level 0. In our case, it corresponds to the maximum concentration for each meteorological situation. See Figure 2.

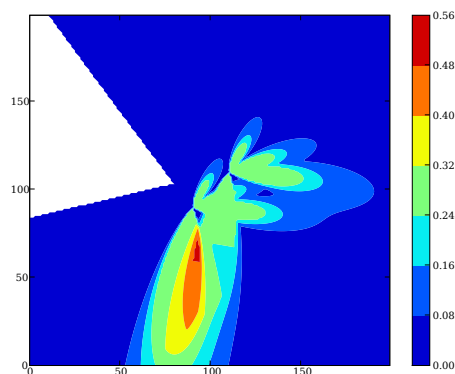


Figure 2: Plume visualization for all meteorological situations.

4.2 Puff simulation with aerosol species

4.2.1 Simulation

This simulation is run with the main configuration file `puff_aer.cfg`. It uses the same files as the plume simulation for vertical levels and species. It uses the meteorological data files created during preprocessing. The saver file, `puff-saver_aer.cfg` is different, because in addition to saving gaseous species, it also has to specify which aerosol species and diameters are saved. For more information about it, see Polyphemus user's Guide, part **Drivers/Output savers/SaverUnitDomain_aer**.

In this part, only the first meteorological situation is to be used, so you have to modify the meteorological data file. A useful trick, if you do not want to modify the file too rashly, is to simply add a symbol at the beginning of each section you do not want to be read: for example, `[situation]` becomes `[*situation]`. That way, all information contained in the modified section will not be read, but you can easily come back to the previous file. This works for all configuration files.

Compile the program `puff_aer` and run the simulation. Use `get_info_float` to check your results. You should obtain:

```
-- File "Iodine.bin"
Minimum: 0
Maximum: 4.83669e+07
Mean: 620.781

-- File "aer1_0.bin"
Minimum: 0
Maximum: 4.83669e+10
Mean: 608406

-- File "aer1_1.bin"
Minimum: 0
Maximum: 4.83669e+10
Mean: 608221

-- File "aer2_0.bin"
Minimum: 0
Maximum: 4.83669e+10
Mean: 640519

-- File "aer2_1.bin"
Minimum: 0
Maximum: 4.83669e+10
Mean: 639552
```

4.2.2 Visualizing results

Without quitting `ipython`, you can move to the result directory for `puff` case with aerosol species. Use `getd` to import the concentration field of a given species. Then, visualize the puff for different time steps with `contourf`.

To visualize several time steps on the same figure, just use `contourf` several times. If you want to clear the figure, use the command `clf()`. Figure 3 gives an example of what you can obtain.

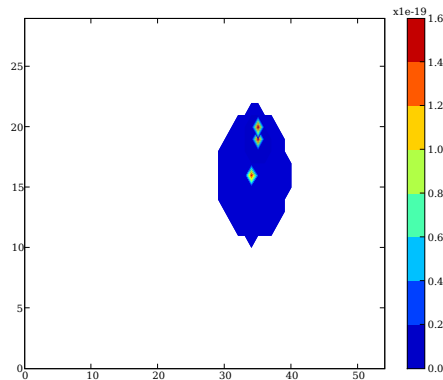


Figure 3: Puff visualization at time steps 0, 30 and 79 for species aer1 and first diameter.

Add two more diameters for particles: $8\ \mu\text{m}$ and $10\ \mu\text{m}$. Use the same data for scavenging and deposition as for particles of diameter $5\ \mu\text{m}$. Rerun the simulation. Save concentrations for all species and diameters. Check your results.

4.3 Puff simulation with line sources

4.3.1 Simulation

This simulation is run with the program `puff` and the main configuration file `puff.cfg`. Use only the first meteorological situation. Also note that, except from using a line source, this simulation is simpler: there are no loss processes, hence no need to compute coefficients. The species file is reduced to the section `[species]`, so it is written directly in the main configuration file. The source file contains a discretized line source similar to the one created during preprocessing.

Run the simulation.

4.3.2 Visualizing results

Now, go into the directory for `puff` case with line source. First visualize the concentration field of species `CO2` for 80 time steps.

Figure 4 gives an example of what you can obtain.

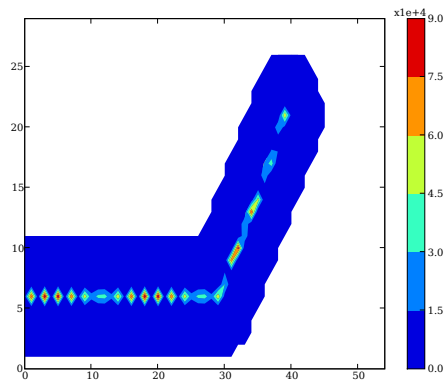


Figure 4: Puff line source for species CO2 at time step 79.