Model of
Urban
Network of
Intersecting
Canyons and
Highways

MUNICH 1.1
User's Guide

CEREA (ENPC – EDF R&D)
Youngseob Kim, Lya Lugon,
Karine Sartelet

http://cerea.enpc.fr/munich
https://gitlab.enpc.fr/cerea/munich

Last update: December 19, 2019

Contents

1 Introduction

The Model of Urban Network of Intersecting Canyons and Highways (MUNICH) is used to simulate subgrid concentrations in the urban canopy represented by the street network [Kim et al., 2018].

MUNICH consists of two main components:

- street-canyon component, which represents the atmospheric processes in the volume of the urban canopy.
- street-intersection component, which represents the processes in the volume of the intersection.

These components are designed to connect to the Polair3D model at roof level and are also interconnected [Kim et al., 2018].

2 Requirements

MUNICH is designed to run under Unix or Linux-based systems. And it is based on three scientific languages: C++, Fortran 77/90 and Python. Tested C++ compilers are GNU gcc 4.x, 5.x, 7.x. GNU gcc 6.x series is not tested. Corresponding Fortran compilers are tested: GNU gfortran 4.x, 5.x, 7.x. For any compiling issue, please report to `munich-help@liste.enpc.fr`.

Python version 2.7 is supported.

2.1 External Libraries and Python Modules

With regard to software requirements, below is a list of requirements

- the C++ library Blitz++ (`https://github.com/blitzpp/blitz`): version 0.10 is supported.
- NumPy: any recent version. Make sure that your versions of NumPy and Matplotlib (see below) are compatible.
- Matplotlib: any recent version and corresponding pylab version (usually, pylab is included in Matplotlib package). It is recommended to install the corresponding version of Basemap in order to benefit from AtmoPy map-visualizations. Basemap is a toolkit available on Matplotlib website (`https://matplotlib.org/`), but usually not included in Matplotlib package.
- SciPy: any recent version.

2.2 Build

MUNICH is supposed to be compiled with SCons. Please make sure that SCons has already been installed in your system `http://www.scons.org`. SCons is supposed to find the compilers and to properly determine all dependencies, on any platform.

```
cd processing/photochemistry
scons
```

If you experience anyway problems, please contact the MUNICH mailing list `munich-help@liste.enpc.fr`.

2.3 Parallel Computing

The chemistry part of MUNICH is parallelized with OpenMPI v1.10. The street segments are partitioned for available cores.

A build option `mpi=yes` should be added.

```
cd processing/photochemistry
scons mpi=yes
```

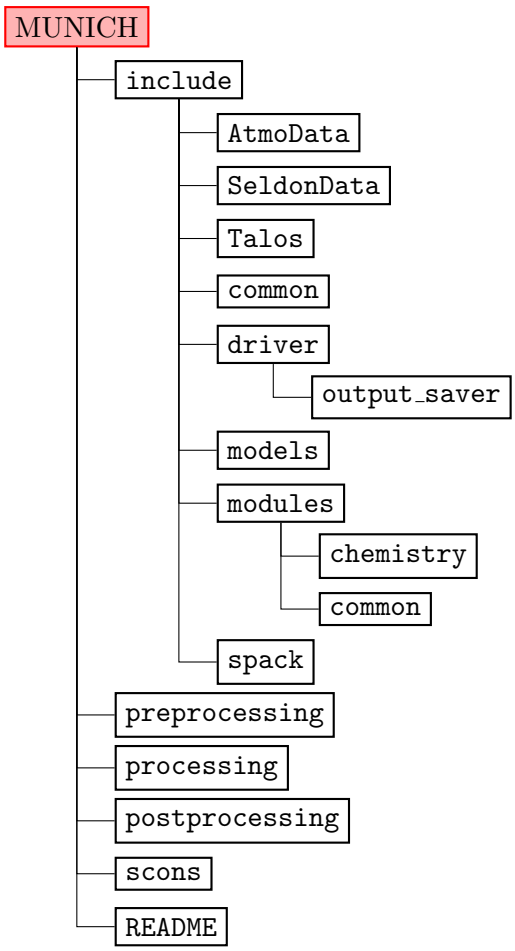
2.4 Run

If a compiling is successful, `munich` is generated. It is run with the main configuration file `munich.cfg`.

```
cd processing/photochemistry
munich munich.cfg
```

3 Source code

The following diagram shows the tree structure of the source code.



The folder `include` is where all source code files are stored.

AtmoData is a tool for data processing in atmospheric sciences.

SeldonData: C++ library to perform data processing (interpolations, input/output operations);

Talos: C++ library to manage configuration files, dates and string processing;

common: mostly, functions used to parse and manage the arguments of preprocessing programs;

driver contains StreetDriver and subroutines for output saver.

models contains models including two major models StreetNetworkChemistry and StreetNetworkTransport which are used by the driver StreetDriver

modules

- chemistry: chemical kinetic mechanisms. These routines are generated from a list of reactions and species specified in the folder spack.
- common: base modules and parallelism module

spack contains the gas-phase chemical model generator.

The folder processing contains the main program munich.cpp in its subfolder photochemistry. Some useful tools are added to generate input data and to visualize output data in the folder preprocessing and postprocessing, respectively. scons folder contains the files necessary for compiling the program.

3.1 Configuration

The main program munich in the folder processing/photochemistry is configured with three configuration files (munich.cfg, munich-data.cfg and munich-saver.cfg) and three data files (intersection.dat, street.dat and species-cb05.dat). The main configuration file munich.cfg provides the paths to the three data files. There are several output files: concentrations for each species and text-type files which contain concentrations for all species at each time step.

3.1.1 Main Configuration File: munich.cfg

The main configuration file munich.cfg gives informations on the options of the simulation. There are many sections, e.g., [display], [domain], in the file.

The section [display] is configured with StreetDriver in the folder driver

[display]	
Show_iterations	If activated, each iteration is displayed on screen.
Show_date	If activated, the starting date of each iteration is displayed on screen in format YYYY-MM-DD HH:II (For notations, please see Section 7).

The sections [domain], [data] and [output] are configured with StreetNetworkTransport model.

[domain]

`Date_min`	Starting date in any legal format (see Section 7). The date can therefore include seconds.
`Delta_t`	Time step in seconds.
`Nt`	Number of iterations of the simulation (integer).
`Species`	Path to the file that defines involved species and their chemical properties.
	`[data]`
`Data_description`	Path to the configuration file that describes input data.
`Horizontal_diffusion`	Horizontal diffusion coefficient in $m^2\,s^{-1}$.
`Isotropic_diffusion`	If activated, horizontal diffusion is set equal to vertical diffusion.
	`[output]`
`Configuration_file`	Path to the configuration for the output saver.

The section `[options]` is also configured with StreetNetworkTransport model.

	`[street]`
`Transfert_parameterization`	Parameterization to compute turbulent transfert velocity: "Sirane" or "Schulte"
`Mean_wind_speed_parameterization`	Parameterization to compute mean wind speed within the street-canyon: "Sirane" or "Lemonsu"
`With_horizontal_fluctuation`	If the horizontal fluctuation is taken into account.
`Intersection`	File containing the input data for intersections.
`Street`	File containing the input data for streets.
`Minimum_Street_Wind_Speed`	Minimum wind speed within the streets.
`With_local_data`	If meteo data and background concentrations are available for each street.

The section `[options]` is configured with StreetNetworkChemistry model.

	`[options]`
`With_chemistry`	Should chemistry occur?
`With_photolysis`	Should photolysis occur?
`Photolysis_tabulation_option`	Read only if `With_tabulated_photolysis`. If `1`, the tabulation generated by SPACK is used. If `2`, the binary files obtained by preprocessing tools (JProc or FastJ) are used.
`With_adaptive_time_step_for_gas_chemistry`	With adaptive time stepping for gaseous chemistry?
`Adaptive_time_step_tolerance`	Tolerance for the adaptive time step.
`Min_adaptive_time_step`	Minimum for the adaptive time step.
`Option_chemistry`	Chemistry mechanism used in the simulation. You can choose among `RACM`, `RACM2` and `CB05`.

3.1.2 Data Description File: `munich-data.cfg`

This configuration file describes input data files (binary files). It is divided into sections: for emission, for meteorological fields, etc. A section roughly looks like this:

```
Directory: /net/libre/yomi/kimy/StreetInGrid/Trafipollu/sing-voc/data-munich/

[meteo]

Date_min: 20140316_00
Delta_t = 3600.
Nt = 2568

Fields: WindDirection WindSpeed PBLH UST LMO WindDirectionInter
        WindSpeedInter PBLHInter USTInter LMOInter Attenuation
        SpecificHumidity SurfacePressure SurfaceTemperature Rain
Filename: <Directory>/meteo/&f.bin
```

It is assumed that all binary files start at the same date, and this date is `Date_min` (see dates formats in Section 7). The time step is `Delta_t`, in seconds.

Then a list of fields is provided after `Fields`. These are fields that the model needs, and their names are determined by the model. Below, all fields required by the model (depending on its options) are listed. A generic path (full file name) is then provided (entry `Filename`). In this path, the shortcut '`&f`' refers to a field name.

Note that:

1. entries `Fields`, `Filename` and additional paths *must* be at the *end of the section*, and *in this order*;
2. *at least one element* (possibly not a required field) must be provided to `Fields` and at least one element (possibly not a path) to `Filename`; for instance:

   ```
   Fields: ---
   Filename: --- # means no generic path.
   ```

   but:

   ```
   Fields:    # Illegal: one element required.
   Filename:     # Illegal: one element required.
   ```

In most sections, `Fields` is used to specify all chemical species involved in the process, e.g.:

```
[emission]

Date_min: 20140316_00
Delta_t = 3600.
Nt = 2568

Fields: ALDX API CH4 ETH ETHA ETOH FORM IOLE ISOP LIM MEOH OLE PAR TERP TOL XYL NO NO2
Filename: <Directory>/emission/&f.bin

ALD2  /net/libre/yomi/kimy/StreetInGrid/Trafipollu/sing-voc/data-munich/ALD2-modified.bin
CO    0.002
```

If a few fields are not stored in a file with a generic path, their specific paths can be provided after the entry `Filename`. This is the case for `ALD2` in the above example.

Notice that `CO` is not associated with a path but with a numerical value. This is a feature: a binary file may be replaced with a numerical value. In this case, the field (in the example, CO emission) is set to a constant value (in every location and at every time step). This works with any field, including meteorological fields (section `[meteo]`).

In `munich-data.cfg`, several sections are required. Several sections have to be included only if given options are activated. In the following table, all possible sections are listed, with their entries.

Section	Entries	Comments
`[emission]`	`Date_min`, `Delta_t`, `Nt` `Fields`, `Filename`	
`[meteo]`	`Date_min`, `Delta_t`, `Nt`, `Fields`, `Filename`	Required fields are: `WindDirection`, `WindSpeed`, `PBLH`, `UST`, `LMO`, `WindDirectionInter`, `WindSpeedInter`, `PBLHInter`, `USTInter`, and `LMOInter`. `Attenuation`, `SpecificHumidity`, `SurfacePressure`, `SurfaceTemperature`, `Rain` with option `with_chemistry`.
`[background_ concentration]`	`Date_min`, `Delta_t`, `Nt` `Fields`, `Filename`	
`[deposition]`	`Fields`	
`[scavenging]`	`Fields`	

A section `[photolysis_rates]` may be required (if the chemical mechanism includes photolysis reactions). Depending on the chosing option for photolysis rates, different fields are read. If photolysis rates are tabulated, they depend on days, time angle, latitude and altitude. During the time integration, they are linearly interpolated in all cells. The following fields describe the tabulation parameters that must be filled.

Section	Entries	Comments
`[photolysis_rates]`	`Ndays`	Number of steps.
	`Time_angle_min`	Starting time angle in hours.
	`Delta_time_angle`	Time angle step in hours.
	`Ntime_angle`	Number of time angles.
	`Latitude_min`	First latitude in degrees.
	`Delta_latitude`	Step along latitude in degrees.
	`Nlatitude`	Number of latitude steps.
	`Altitudes`	List of altitudes in meters at which photolysis rates are provided.
	`Date_min`	Starting dates of photolysis rates.
	`Delta_t`	Time step (in days if tabulated photolysis rates).
	`Fields, Filename`	Photolysis reaction names and the paths to the files in which photolysis rates are stored.

3.1.3 Output Description File: `munich-saver.cfg`

Some parameters must be provided to save output results. The file type of output files is binary as input data.

```
Output_dir: results/

[save]

# Put "all" to output all species.
Species: all

Date_beg: -1 # Put -1 to start from the simulation initial date.
Date_end: -1 # Put -1 to end at the simulation final date.
Interval_length: 36 # 1 for all steps.
Averaged: yes
# Save initial concentrations in case concentrations are not averaged?
Initial_concentration: no

# Choices: streeet
Type: street

Levels: 0

Output_file: <Output_dir>/&f.bin

Text_file: no
```

[save]	
`Species`	Chemical species to be saved. If it is set to `all`, concentrations for all species are saved.
`Date_beg`	The date from which the concentrations are saved. If concentrations are averaged, the first step at which concentrations are actually saved if not `Date_beg`, but `Date_beg` plus the number of steps over which concentrations are averaged. If the value - 1 is supplied, `Date_beg` is set at the start of the simulation.
`Date_end`	The last date at which concentrations may be saved. If the value - 1 is supplied, `Date_end` is set at the end of the simulation.
`Interval_length`	The number of steps between saves.
`Type`	The type of saver, it is `street` for MUNICH simulations.
`Output_file`	The full path of output files, in which `&f` will be replaced by the name of the chemical species. Note that the directory in which the files are written must exist before the simulation is started.
`Text_file`	output files in a text type are provided for an easy and fast file checking
`Levels`	it is not used.

Note that `Species`, `Date_beg`, `Date_end`, `Interval_length` must appear before `Type`. After `Type`, put additional options relevant for the chosen output saver.

3.2 Input data files

3.2.1 `intersection.dat` and `street.dat`

- `intersection.dat`: intersection id, longitude and latitude of the intersection, number of streets whice are connected to the intersection followed by the connected street id.

```
#id;lon;lat;number_of_streets;1st_street_id;2nd_street_id;...
1;2.49961040621;48.8639959388;6;1;605;852;3;8;11;
2;2.49977706824;48.8650938211;1;1;
6;2.49810836399;48.8642684425;1;3;
7;2.50328738802;48.8643530022;1;4;
8;2.50263219496;48.8630192684;4;4;5;10;19;
10;2.50111432732;48.8635255708;4;5;7;664;851;
11;2.48158743003;48.8656369541;1;6;
12;2.48236756109;48.8639495836;3;6;749;710;
13;2.50067262662;48.8626783551;4;7;8;23;676;
```

- `street.dat`: street id, two intersection id which connect the street, street length, width and height.

```
#id;begin_inter;end_inter;length;width;height
1;1;2;122.686160495;7.5;6.9
3;1;6;107.94798805;7.5;6.9
4;7;8;155.856467313;7.5;6.9
5;8;10;124.490042998;7.5;6.9
6;11;12;196.113095869;41.0;10.2
7;13;10;99.5930099431;7.5;6.9
8;1;13;169.563212564;7.5;6.9
```

3.2.2 Input data files: `species-cb05.dat`

CB05 chemical kinetic mechanism is implemented in MUNICH. Chemical species are listed in the section `[species]` and of a configuration file.

Please do not make a change in this `[species]` section. The order of the chemical species in this section is determined by SPACK during compiling. If you need to change chemical kinetic mechanism, e.g., add species and/or reactions, please see SPACK Guide (`https://www.cerea-lab.fr/dossiers/racine/articles/guide-0.pdf`).

```
[species]

O1D  ETOH CH4  ETHA  TOL
XYL  SO2  SULF N2O5  PAN
MEOH TO2  HNO4 PANX  HONO
ROR  H2O2 HCO3 MEPX  FACD
ROOH PACD ETH  OLE  IOLE
...

[molecular_weight]

# Unit: g / mol.
```

```
O1D  16.     ETOH  46.07 CH4   16.04  ETHA  30.07    TOL   92.14
XYL  106.16  SO2   64.06   SULF  98.08  N2O5  108.01   PAN   121.05
MEOH   32.04  TO2   141.15  HNO4  79.01  PANX  120.04   HONO  47.01
ROR    16. H2O2  34.01    HCO3  63.03  MEPX  48.04    FACD  46.03
ROOH   47.03  PACD  76.05   ETH   28.05  OLE   27.05    IOLE  56.11
...
```

a section `[molecular_weight]` lists the molecular weights (in $\mathrm{g\,mol^{-1}}$) of *all* species. If photolysis reactions are involved, the section `[photolysis_reaction_index]` is required. This section provides all reaction names and their indices in the list of reactions. Index begins from 0. The reaction name `NO2` corresponds to the photolysis of NO_2. The number of this reaction in CB05 is 1 and its index in the section `[photolysis_reaction_index]` is 0. For the list of reactions of CB05, please see `/include/modules/chemistry/CB05/CB05.reactions`.

```
[photolysis_reaction_index]

NO2     0   O3O3P   7   O3O1D   8   NO3NO2 13
NO3NO   14  HONO    23  H2O2    34  HNO4   49
HNO3    50  N2O5    51  ORGNIT  60  HOP    62
MHP     69  HCHOrad 72  HCHOmol 73  ALD    84
PAN     88  PACD    94  C2CHO   99 PANX   103
OPEN    133 MGLY    138 ISPD  146
```

3.3 Chemistry Module: Photochemistry

Module `Photochemistry` is the photochemical module used with MUNICH. It implements four chemical mechanisms: RACM, RACM2, CB05 and Leighton ozone photostationary state relation [Kim et al., 2009]. It uses a second-order Rosenbrock method for time integration. Computations are performed by Fortran routines (automatically generated by the chemical preprocessor SPACK, see SPACK Guide `https://www.cerea-lab.fr/dossiers/racine/articles/guide-0.pdf`) and a C++ program is used as a frame to launch all these calculations.

It only deals with gaseous species. Information about species and reactions is given below.

Chemical mechanisms	no of species	no of reactions
`RACM`	72	237 including 23 photolysis
`RACM2`	113	349 including 34 photolysis
`CB05`	52	155 including 23 photolysis
`Leighton`	4	3 including 1 photolysis

The *units* of input data (e.g., initial condition, boundary condition etc.) should be given as $\mu g/m^3$ in using `Photochemistry` module.

4 preprocessing

A python script `sing_preproc.py` can be used and run with a configuration file `sing_preproc.cfg`.

```
cd preprocessing
python sing_preproc.py sing_preproc.cfg
```

	[input]
t_min	Initial date of the simulation.
Delta_t	Time step in hours.
Nt	Number of time step.
emission_dir_weekday	Directory where emission data for weekday are stored.
emission_dir_weekend	Directory where emission data for weekend are stored.
emission_species	Species for emission: CH4 (methane), NMHC (non-methane hydro-carbon), CO (carbon monoxide), NOx (nitrogen oxide).
input_dir	Directory where input data are stored.
geog_info	File describing the geographic data.
background_concentration	File describing the background concentration.
meteo_dir	Directory where meteorological data are stored. Only WRF data can be used.
wrfout_prefix	Prefix in file names of WRF output.
	[output]
Output_dir	Directory where output data will be saved.

```
[input]

t_min: 20140316_00
Delta_t: 1 # in hr
Nt: 1

emission_dir_weekday: input/traffic/weekday/

emission_dir_weekend: input/traffic/weekend/

emission_species: CH4 NMHC CO NOx

input_dir: input/

geog_info: <input_dir>/street-geog-info.dat

background_concentration: <input_dir>/background_concentration.dat

meteo_dir: <input_dir>/meteo/
wrfout_prefix: wrfout_d04

[output]

Output_dir: output/
```

In Output_dir, the following data are generated. All files are binary except those in the directory textfile.

background - background concentration in the street canyons.

emission - traffic emission in the street canyons.

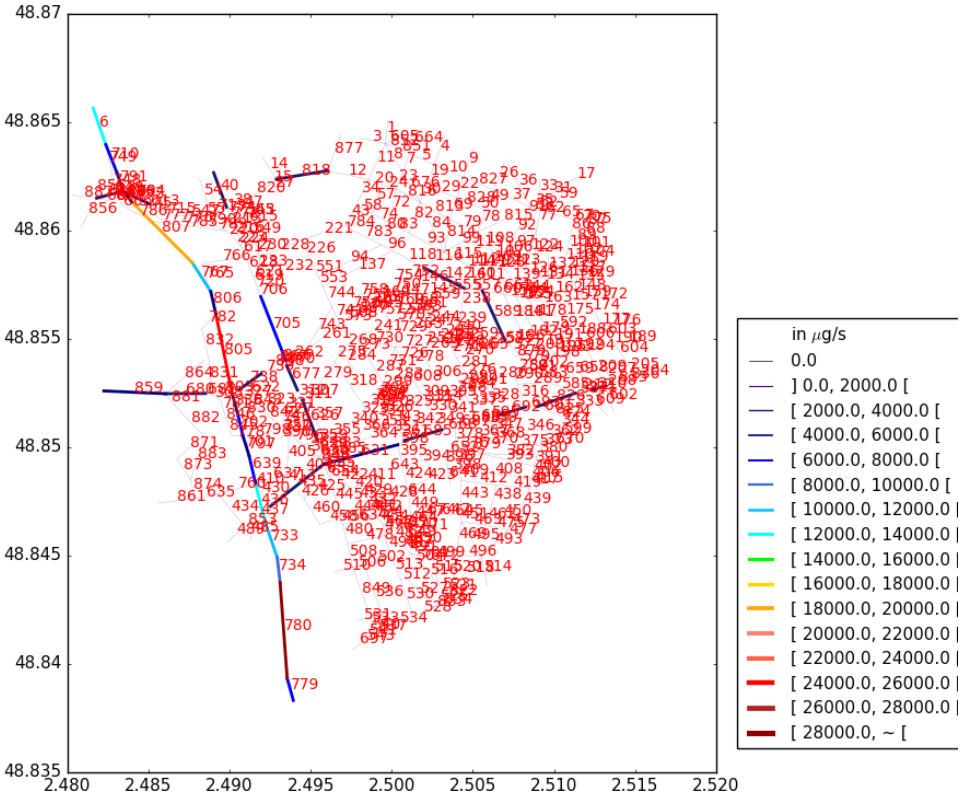


Figure 1: Emission map

`meteo` - meteorological data in the street canyons and the street intersections.

`textfile` - emission data and geographic data (`street.dat` and `intersection.dat` are used in `munich.cfg`).

`*.bin` - Averaged emission data for a given grid cells (not necessary in MUNICH).

For the species `NMHC`, emission data need to be calculated for model species. For this version, informations for CB05 chemical kinetic mechanism are included `aggregation_cb05-siream.dat`.

```
cd preprocesing/utils/speciation
python speciation_aggregation.py
```

A python script `display_emission.py` is provided to visualize emission data which are computed for a given domain.

```
cd preprocesing/utils
python display_emission.py
```

5 Postprocessing

`disp_concentrations.py` is provided to draw simulation results on the map.

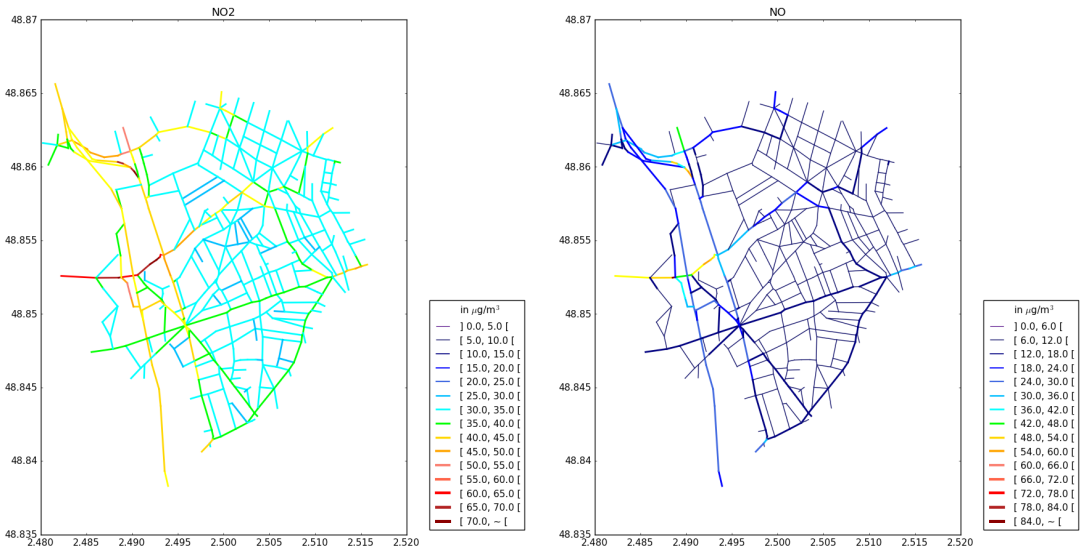


Figure 2: NO and NO_2 concentrations map obtained by `disp_concentration.py`.

```
cd postprocesing
python disp_concentrations.py disp_concentrations.cfg
```

`emission.txt` and `node.txt` are used. You need to copy them from `preprocessing/output/textfile` if they are not found in the directory.

6 Markups

In order to avoid duplications in a configuration file, Polyphemus features a markup management. A markup is denoted with surrounding '<' and '>', e.g. <path>. A markup is automatically replaced with its value whenever it is found. Its value should be provided somewhere in the configuration file with a proper field; for instance, <path> refers to the field `path`. Here is a complete example:

```
Root: /home/user
Input_directory: <Root>/input/
Output_directory: <Root>/output/
```

means:

```
Input_directory: /home/user/input/
Output_directory: /home/user/output/
```

The markup can be used before its value is defined:

```
Input_directory: <Root>/input/
Output_directory: <Root>/output/
Root: /home/user # After calls to <Root>. This is legal.
```

Any field may be used as a markup. The user may define any new markup (that is a new field). Moreover, several markup substitutions can be performed in a single value, and nested markups are properly handled:

```
Home: /home/user
Root: <Home>/Polyphemus/work
Number = 7
Input_directory: <Root>/input-<Number>/
```

is accepted and means:

```
Input_directory: /home/user/Polyphemus/work/input-7/
```

Notice that markups may also replace numbers and may be based on preexisting fields:

```
x_min = 12.5    Delta_x = 0.5    Nx = 100
y_min = <x_min> Delta_y = 1.     Ny = <Nx>
```

7 Dates

Date formats are:

```
YYYY                # Year.
YYYY-MM             # With the month.
YYYY-MM-DD          # With the day.
YYYY-MM-DD_HH       # With the hour.
YYYY-MM-DD_HH-II    # With the minute.
YYYY-MM-DD_HH-II-SS # With the second.
```

Months range from `01` to `12`. Days range from `01` to `31`. Hours range from `00` to `23`. Minutes and seconds range from `00` to `59`.

If the month is not specified (format `YYYY`), then the month is set to `01` (January). If the day is not specified (formats `YYYY` and `YYYY-MM`), it is set to `01` (first day of the month). If the hour, the minute or the second is not specified, it is set to zero (`00`).

Hyphens and underscores may be replaced with any character that is neither a delimiter nor a cipher. They can also be removed. Examples:

```
19960413
1996-04-13_20h30
1996/04/13@2030
```

Recommandation – Use hyphens around the month and around minutes. Use an underscore between the day and the hour (`YYYY-MM-DD_HH-II-SS`).

8 References

References

Kim, Y., Sartelet, K., and Seigneur, C. (2009). Comparison of two gas-phase chemical kinetic mechanisms of ozone formation over Europe. *J. Atmos. Chem.*, 62(2):89–119.

Kim, Y., Wu, Y., Seigneur, C., and Roustan, Y. (2018). Multi-scale modeling of urban air pollution: development and application of a Street-in-Grid model (v1.0) by coupling MUNICH (v1.0) and Polair3D (v1.8.1). *Geosci. Model Dev.*, 11:611–629.