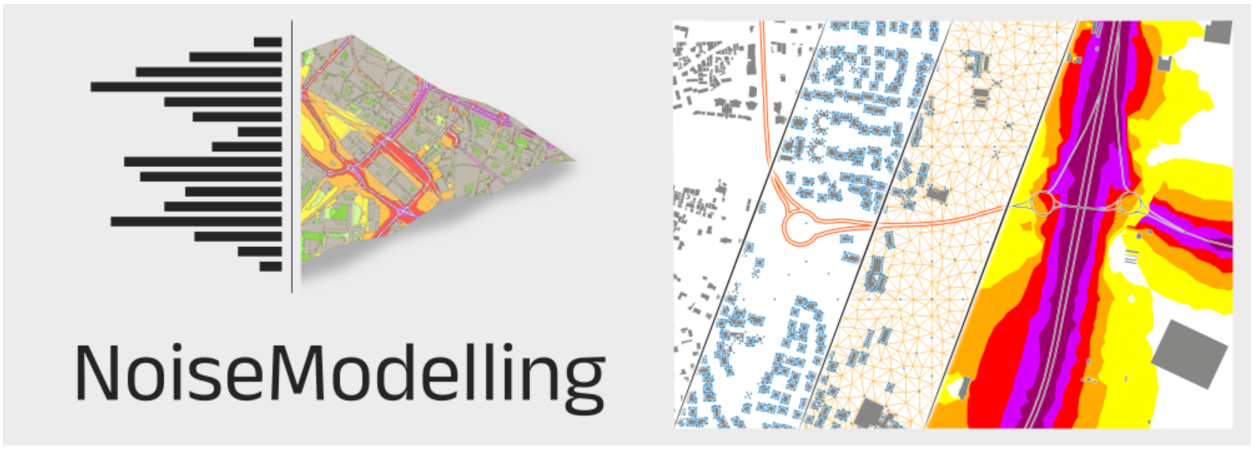
NoiseModelling Documentation

Release 4.0.6-SNAPSHOT

Aumond P., Fortin N., Le Bescond V., Petit G.

Sep 25, 2023

NoiseModelling presentation



Welcome on the **official NoiseModelling v4.0 User Guide**.

NoiseModelling is a library capable of producing noise maps. It can be freely used either for research and education, as well as by experts in a professional use.

A general overview of the model (v3.4.5 - September 2020) can be found in this video.

- for **more information** on NoiseModelling, visit the offical NoiseModelling website
- to **contribute to NoiseModelling** source code, follow the "*Get Started*" page
- **to contact the support / development team,**
  - open an issue or a write a message *(we prefer these two options)*
  - send us an email at contact@noise-planet.org

CHAPTER 1

What's new with the V4.0?

Since the release v4.0, NoiseModelling implements the CNOSSOS-EU standard method for the noise emission (road and rail (for France)) and with noise propagation (read "*Numerical Model*" and "*Validation*" pages for more information).

1.1 Optimizations

- H2 and H2GIS versions have been upgraded (respectively to v2.0.202 and v2.0.0)
- Triangulation library Poly2Tri has been replaced by Tinfoor
- Triangulation to accelerate the propagation is not used anymore (only used in DEM intersections test)

1.2 Packaging

On the NoiseModelling latest release page, three packages of NoiseModelling are proposed:

- `NoiseModelling_4.0.0.zip` : cross-platform version, with GUI (Graphic User Interface)
- `NoiseModelling_4.0.0_install.exe` : windows installer, with GUI
- `NoiseModelling_4.0.0_without_gui.zip` : version without GUI. Usefull to run NoiseModelling using command lines (read "*Pilot NoiseModelling with scripts*" page for more info)

CHAPTER 2

Authors

NoiseModelling project is leaded by acousticians from the *Joint Research Unit in Environmental Acoustics* (UMRAE, Université Gustave Eiffel - Cerema) and Geographic Information Science specialists from Lab-STICC laboratory (CNRS - DECIDE Team).

The NoiseModelling team owns the majority of the authorship of this application, but any external contributions are warmly welcomed.

CHAPTER 3

Licence

NoiseModelling and its documentation are distributed for free under GPL v3 *License*.

CHAPTER 4

Publications

NoiseModelling was initially developed in a research context, which has led to numerous scientific publications. For more information, have a look to "*Scientific production*" page. To quote this tool, please use the bibliographic reference below:

Note: Erwan Bocher, Gwenaël Guillaume, Judicaël Picaut, Gwendall Petit, Nicolas Fortin. *NoiseModelling: An Open Source GIS Based Tool to Produce Environmental Noise Maps*. ISPRS International Journal of Geo-Information, MDPI, 2019, 8 (3), pp.130. (10.3390/ijgi8030130)

CHAPTER 5

Fundings

Research projects:

- ANR Eval-PDU (ANR-08-VILL-0005) 2008-2011
- ANR VegDUD (ANR-09-VILL-0007) 2009-2014
- ANR CENSE (ANR-16-CE22-0012) 2017-2021
- Nature4cities (N4C) project, funded by European Union's Horizon 2020 research and innovation programme under grant agreement N°730468
- PlaMADE 2020-2022

Institutional (public) fundings:

- Université Gustave Eiffel (formerly Ifsttar, formerly LCPC), CNRS, Cerema, Université Bretagne Sud, Ecole Centrale de Nantes

Private fundings:

- Airbus Urban Mobility

Warning:

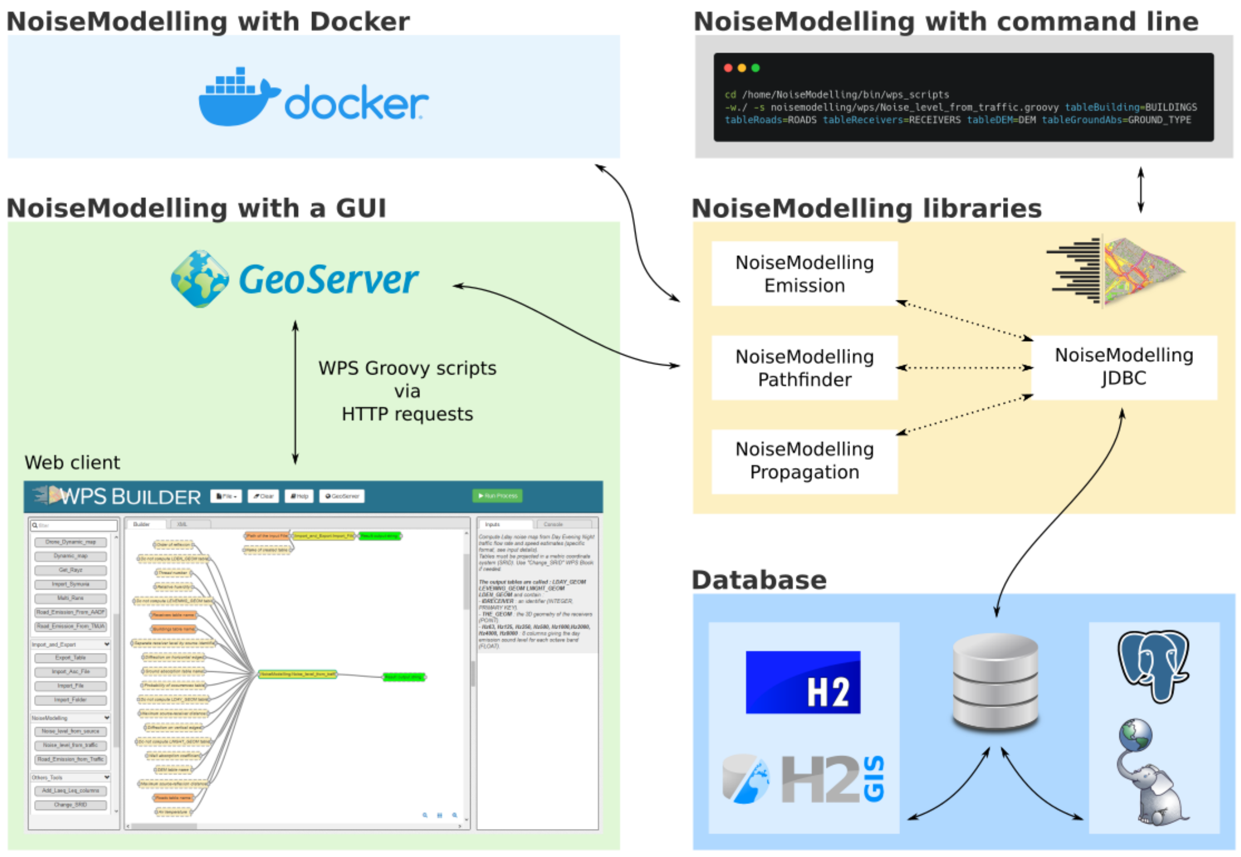
- The official documentation is available in English only
- Some illustrations may refer to previous versions of NoiseModelling
- If you observe some mistakes or errors, please open an issue here or contact us at contact@noise-planet.org
- You are also welcome to contribute to the documentation (click on *"Edit on Github"* - top of the page)

5.1 Architecture

NoiseModelling is the name of the application that allows to calculate noise maps (notably through a Graphical User Interface). But did you know that it is also the name of different calculation libraries?

The documentation below presents the architecture of NoiseModelling with its different bricks and the ways to launch it:

1. NoiseModelling libraries
2. Database connection
3. NoiseModelling with a Graphical User Interface (GUI)
4. NoiseModelling with command line
5. NoiseModelling with Docker



```
cd /home/NoiseModelling/bin/wps_scripts
-w./ -s noisemodelling/wps/Noise_level_from_traffic.groovy tableBuilding=BUILDINGS
tableRoads=ROADS tableReceivers=RECEIVERS tableDEM=DEM tableGroundAbs=GROUND_TYPE
```

5.1.1 1. NoiseModelling libraries

NoiseModelling is made of 4 main librairies:

- `noisemodelling-emission` : to determine the noise emission
- `noisemodelling-pathfinder` : to determine the noise path
- `noisemodelling-propagation` : to calculate the noise propagation
- `noisemodelling-jdbc` : to connect NoiseModelling to a database

These libraries may be used independently of each other. Note that the `noisemodelling-jdbc` library *(JDBC = Java DataBase Connectivity)* is central since it allows the three others to communicate with each other as soon as the data are stored in a database *(which is the default situation)*.

5.1.2 2. Database connection

Thanks to the `noisemodelling-jdbc` library, NoiseModelling can access and communicate with databases. This system is quite adapted to store, manage and process (spatial) data. Here, the user has the choice between to database (free, open-source and powerful) couples:

- H2 / H2GIS, which is configured and embeded by default. In this case, the user has nothing to do.
- PostGreSQL / PostGIS. In this case, the user has to configure the connexion (read "*Use NoiseModelling with a PostGIS database*" page for more information).

In both cases, database can be local or remote.

5.1.3 3. NoiseModelling with a GUI

NoiseModelling has a Graphical User Interface (GUI). It is accessible through a web browser (here http://localhost:9580/geoserver/web/) and is generated by a module named "*WPS Builder*".

In order for "WPS Builder" to communicate with the NoiseModelling libraries, we use a *'bridge'* named GeoServer. This free and open-source software, allows (among other cool things) to execute WPS* scripts, written in Groovy language, via HTTP requests.

* Web Processing Service, which is a standard from the Open Geospatial Consortium (OGC).

Note: When launching NoiseModelling, Geoserver is started first. In your terminal, you will have a lot of log messages. Most of them are coming from Geoserver and are not directly linked to NoiseModelling. Unfortunately, we can not remove them.

You can see NoiseModelling with a GUI in action in the page "*Get Started - GUI*".

5.1.4 4. NoiseModelling with command lines

You can use NoiseModelling with command lines. To do so,

1. Open a terminal
2. Go in the NoiseModelling directory
3. Call the WPS .groovy script you want, with the needed arguments

Note: The `.groovy` script may be simple (the ones already provided with NoiseModelling, executing one task) or complex (tailor made by users and calling one or many `.groovy` script(s)).

Note: No need to launch / start the application as we do with Geoserver. Here the NoiseModelling libraries are called directly for each instructions.

Examples can be found in the page "*Pilot NoiseModelling with scripts*".

5.1.5 5. Docker Setup

When a developer uses Docker, he creates an application or service, which he then bundles together with the associated dependencies in a container image. An image is a static representation of the application or service, its configuration and dependencies.

Available versions

The Docker images listed below have been built by NoiseModelling contributors / users. Many thanks to them!

- v4.0.1 image, built by Alexander (Aka "Xenotech81")
- v3.4.4 image, built by Tomáš Anda (Aka "tomasanda")

5.2 Numerical Model

5.2.1 Emission Numerical Model

Road traffic emission model

The emission model of the implemented road traffic is the CNOSSOS-EU model.

User can choose coefficients from the Directive 2015/996 and its amendment 2019/1010.

Rail traffic emission model

The emission model of the implemented rail traffic is the CNOSSOS-EU model.

Only french database, from SNCF, is implemented.

Without emission model

User can also add directly its own emission sound power level (LW).

5.2.2 Path finding algorithm

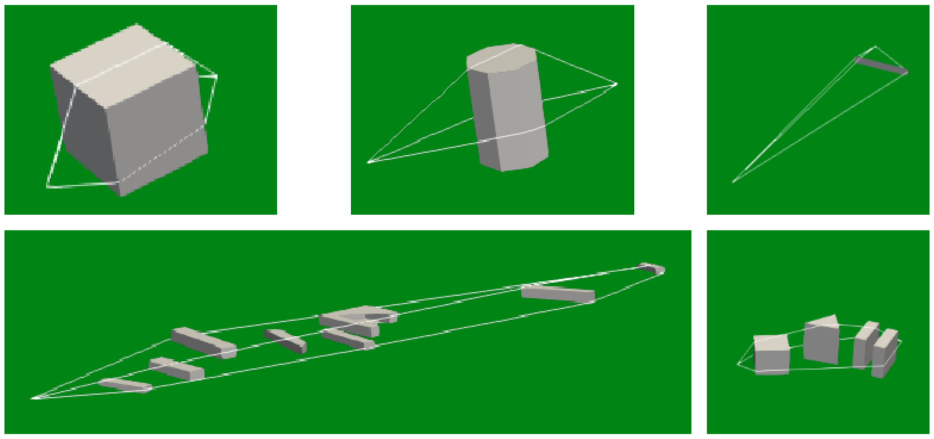
The path finding algorithm is a rubber-band like algorithm as specified in CNOSSOS-EU.

To optimize the processing time, this algorithm is taking benefit from a R-Tree spatial partioning algorithm.

Warning: Rays backwards to the source or receiver are not taken into account. For example, if a receiver is located inside a U-shaped building, only diffractions on horizontal edges will be taken into account.

5.2.3 Propagation Numerical Model

The propagation model is the CNOSSOS-EU one.



5.3 Validation

5.3.1 Acoustic model validation

Please refer to CNOSSOS-EU papers, or other scientific papers, which are independant from NoiseModelling.

Some limits are given in the CNOSSOS-EU documents below:

Note: Source : Stylianos Kephalopoulos, Marco Paviotti, Fabienne Anfosso-Lédée. Common noise assessment methods in Europe (CNOSSOS-EU). PUBLICATIONS OFFICE OF THE EUROPEAN UNION, p.75/180 2012,10.2788/31776

- Height receivers must be > 2m
- Propagation distance must be < 800 m
- Downward-refraction/ homogeneous are taken into acount
- 63 Hz to 4 000 Hz – center band
- Breakdown of the infrastructures into point sources
- Does not apply to propagation scenarios above a water body (lake, wide river, etc.).
- The effects of tunnel mouths are not dealt with by the method.
- This method considers obstacles to be equivalent to flat surfaces.

Note: Source : https://circabc.europa.eu/sd/a/9566c5b9-8607-4118-8427-906dab7632e2/Directive_2015_996_EN.pdfde

This document specifies a method for calculating the attenuation of noise during its outdoor propagation. Knowing the characteristics of the source, this method predicts the equivalent continuous sound pressure level at a receiver point corresponding to two particular types of atmospheric conditions:

- downward-refraction propagation conditions (positive vertical gradient of effective sound celerity) from the source to the receiver
- homogeneous atmospheric conditions (null vertical gradient of effective sound celerity) over the entire area of propagation.

The method of calculation described in this document applies to industrial infrastructures and land transport infrastructures. It therefore applies in particular to road and railway infrastructures. Aircraft transport is included in the scope of the method only for the noise produced during ground operations and excludes take-off and landing.

Industrial infrastructures that emit impulsive or strong tonal noises as described in ISO 1996-2:2007 do not fall within the scope of this method.

The method of calculation does not provide results in upward-refraction propagation conditions (negative vertical gradient of effective sound speed) but these conditions are approximated by homogeneous conditions when computing Lden.

To calculate the attenuation due to atmospheric absorption in the case of transport infrastructure, the temperature and humidity conditions are calculated according to ISO 9613-1:1996.

The method provides results per octave band, from 63 Hz to 8 000 Hz. The calculations are made for each of the centre frequencies.

Partial covers and obstacles sloping, when modelled, more than 15° in relation to the vertical are out of the scope of this calculation method.

A single screen is calculated as a single diffraction calculation, two or more screens in a single path are treated as a subsequent set of single diffractions by applying the procedure described further.

5.3.2 Implementation validation

A large set of unit tests are present in the code. Please consult an example dealing with CNOSSOS-EU here.

Note that all the tests entilted `TCxx` (see example) are coming from the ISO/TR 17534-4:2020 standard , which has been implemented in NoiseModelling.

5.4 Scientific production

Below is a non-exhaustive list of articles or presentations in which NoiseModelling is used.

5.4.1 Standard Noise maps

BACLET S., VENKATARAMAN S., RUMPLER R., BILLSJÖ R., HORVATH J., ÖSTERLUND P. E., , From strategic noise maps to receiver-centric noise exposure sensitivity mapping, Transportation Research Part D: Transport and Environment, 2022, vol. 102 *(Noise mapping, Road traffic noise, Population exposure, Road network sensitivity)*

GRAZIUSO G., FRANCAVILLA A. B., MANCINI S., GUARNACCIA C., Open-source software tools for strategic noise mapping: a case study, Journal of Physics: Conference Series, 2022, vol. 2162, 012014

AUMOND P., BOCHER E., ECOTIERE D., FORTIN N., GAUVREAU B., GUILLAUME G., PETIT G., Improvement of city noise map production processes and sensitivity analysis to noise models inputs, Euronoise Conference Proceedings, 2021, p. 1128

BACLET S., VENKATARAMAN S., RUMPLER R., A methodology to assess the impact of driving noise from individual vehicles in an urban environment, Resource Efficient Vehicles Conference, 2021.

NOURMOHAMMADI Z., LILASATHAPORNKIT T., ASHFAQ M., et al., Mapping Urban Environmental Performance with Emerging Data Sources: A Case of Urban Greenery and Traffic Noise in Sydney, Australia, Sustainability, 2021, vol. 13, n° 2, p. 605

BAEZA J. L., SIEVERT J. L., LANDWEHR A., et al., CityScope Platform for Real-Time Analysis and Decision-Support in Urban Design Competitions, International Journal of E-Planning Research (IJEPR), 2021, vol. 10, n° 4, p. 1-17

WANG Z., NOVACK T., YAN Y., ZIPF A., Quiet Route Planning for Pedestrians in Traffic Noise Polluted Environments, IEEE Transactions on Intelligent Transportation Systems, 2020

AUMOND P., FORTIN N., CAN A., Overview of the NoiseModelling open-source software version 3 and its applications, INTER-NOISE and NOISE-CON Congress and Conference Proceedings, 2020, vol. 261, n°4, p. 2005-2011

5.4.2 Dynamic Noise maps

LE BESCOND V., CAN A., AUMOND P., GASTINEAU P., Open-source modeling chain for the dynamic assessment of road traffic noise exposure, Transportation Research Part D: Transport and Environment, 2021, vol. 94, 102793 (Watch a short presentation on Youtube)

CAN A., AUMOND P., BECARIE, C., LECLERCQ, L., Dynamic approach for the study of the spatial impact of road traffic noise at peak hours, Proceedings of the 23rd International Congress on Acoustics, Aachen, Allemagne, 09-13 September, 2019

QUINTERO G., AUMOND P., CAN A., BALASTEGUI A., ROMEU J., Statistical requirements for noise mapping based on mobile measurements using bikes, Applied Acoustics, 156, 271-278, 2019

47.0
49.8
52.5
55.3
58.1
60.9
63.7
66.4
69.2
72.0
74.8
77.6
80.3
83.1
85.9
Roads
8:12 / 12:02

https://www.youtube.com/watch?v=jl8tASDr-uQ&t=133s

CAN A., AUMOND P., BECARIE C., LECLERCQ L., Approche dynamique pour l'étude de l'emprise spatiale du bruit de trafic routier aux heures de pointe, Recherche en Transport Sécurité, 2018

5.4.3 Probabilistic & Multi-sources Noise maps

ALIONTE C-G., COMEAGA D-C., Noise assessment of the small-scale wind farm, In : E3S Web of Conferences. EDP Sciences, 2019

AUMOND P., CAN A., Probabilistic modeling framework to predict traffic sound distribution, Proceedings of Euronoise, Hersonissos, Crete, 27-31 May 2018

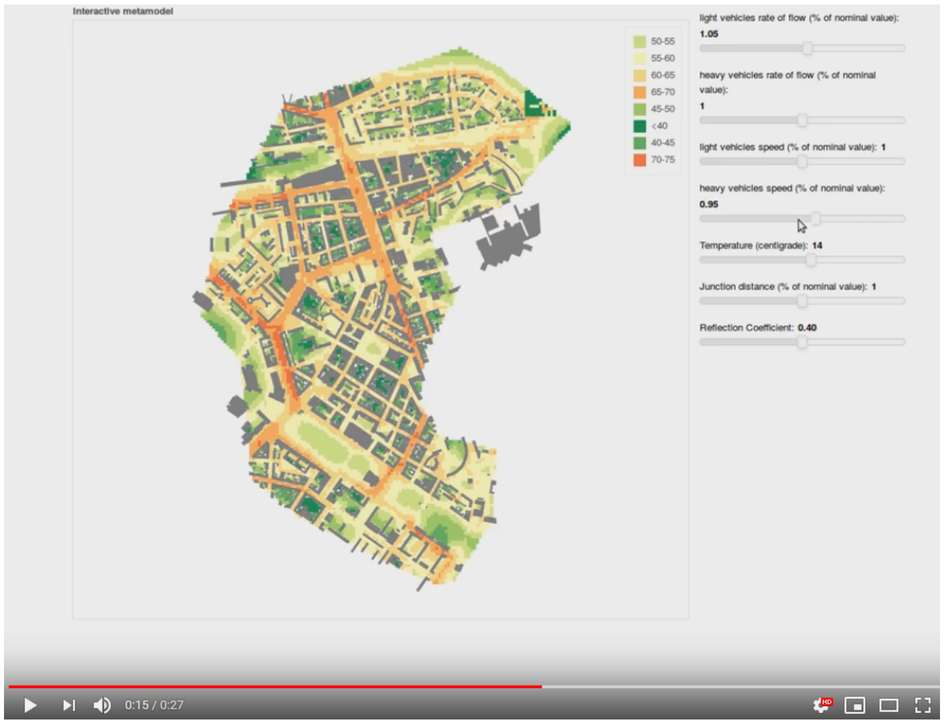
AUMOND P., JACQUESSON L., CAN A., Probabilistic modeling framework for multisource sound mapping, Applied Acoustics, 139, 34-43, 2018

5.4.4 Sensitivity Analysis & data assimilation

LESIEUR A., MALLET V., AUMOND P., CAN A., Data assimilation for urban noise mapping with a meta-model, Applied Acoustics, 2021, vol. 176, 107938,

AUMOND P., CAN A., MALLET V., GAUVREAU B., GUILLAUME G., Global sensitivity analysis of a noise mapping model based on open-source software, Applied Acoustics, 2021, vol. 176, 107899

LESIEUR A., AUMOND P., MALLET V., et al., Meta-modeling for urban noise mapping. The Journal of the Acoustical Society of America, 2020, vol. 148, no 6, p. 3671-3681



light vehicles rate of flow (% of nominal value): 1.05

heavy vehicles rate of flow (% of nominal value): 1

light vehicles speed (% of nominal value): 1

heavy vehicles speed (% of nominal value): 0.95

Temperature (centigrade): 14

Junction distance (% of nominal value): 1

Reflection Coefficient: 0.40

https://www.youtube.com/watch?v=orc5ZbN2dlY

AUMOND P., CAN A., MALLET V., GAUVREAU B., GUILLAUME G., Global sensitivity analysis for urban noise modelling, Proceedings of the 23rd International Congress on Acoustics, Aachen, Allemagne, 09-13 September, 2019

5.4.5 Auralisation

ROHRLICH F. , VERRON C. (Noise Makers), *Captation et Simulation d'Ambiances Urbaines Spatialisées*, 2018-2019

5.5 Buildings

NoiseModelling is a tool for producing noise maps. To do so, at different stages of the process, the application needs input data, respecting a strict formalism.

Below we describe the table `BUILDINGS`, dealing with buildings.

The other tables are accessible via the left menu in the `Input tables & parameters` section.

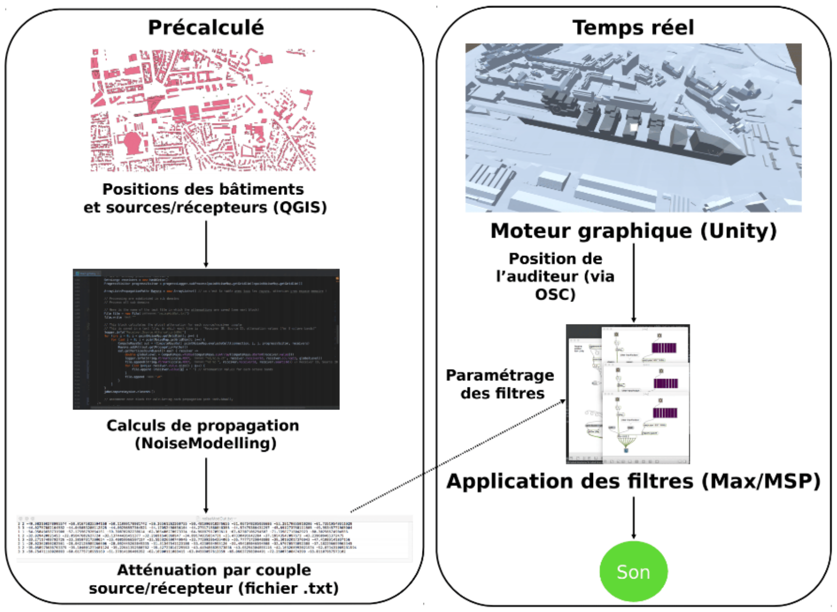

Précalculé
Positions des bâtiments et sources/récepteurs (QGIS)
Calculs de propagation (NoiseModelling)
Atténuation par couple source/récepteur (fichier .txt)
Temps réel
Moteur graphique (Unity)
Position de l'auditeur (via OSC)
Paramétrage des filtres
Application des filtres (Max/MSP)
Son

FIGURE 24 – Schéma de principe du prototype de simulateur d'ambiances urbaines. A gauche, les éléments liés à NoiseModelling, qui sont précalculés ; à droite, les éléments temps réel (Unity et Max/MSP).

5.5.1 Table definition

Warning: In the list below, the columns noted with `*` are mandatory

- **`THE_GEOM`** *
  - Description: building's geometry. It can be in 2D (stuck to the ground) or in 3D (see *Geometry modelling* section below)
  - Type: Geometry (`POLYGON` or `MULTIPOLYGON`)
- **`HEIGHT`** *
  - Description: building's height *(in meters)*
  - Type: Double
- **`POP`**
  - Description: number of inhabitant in the building
  - Type: Double

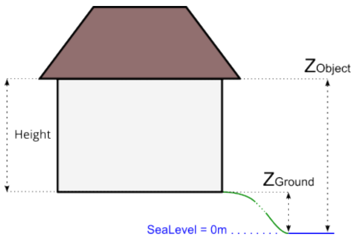
Note: If you want to generate a scene without buildings, create two fictitious buildings, placed in two corners of the scene, and assign them a height of 0 meter.

5.5.2 Geometry modelling

In NoiseModelling, the geometry of the building is used to calculate the 3D ray path of the acoustic wave. Therefore, we need to know the footprint of the building as well as the points in height (at the roof, the gutter, . . .)

To determine the 3D shape of the building we can use some of the following elements:

- `Zground` : The ground altitude, exprimed in meters and based on the 0 sea level
- `Zobject` : The altitude in the air, exprimed in meters and based on the 0 sea level
- `HEIGHT`: The height, equal to the diffirence between `Zobject` and `Zground`



In this context, geometry coordinates have to be in 3D, with:

- `X` and `Y` coordinates corresponding to the building's footprint (or the gutter/roof projection to the ground)
- `Z` = `Zobject` : coordinate corresponding to the gutter or the roof altitude(s), . . .

5.5.3 Z coordinate deduction

Depending on the information you have, NoiseModelling will adpat the process to deduce the `Zobject` information and therefore the 3D frame of the building.

Two cases are possible:

1. The geometry has no Z coordinate

There is a DEM layer

The DEM is triangulated. Then, all the vertices of the building are projected onto the triangle below it in order to determine their altitudes. Finally, the minimum altitude is taken and assigned to the whole building: `Zground` = Minimum DEM Z value. Then:

- If `HEIGHT` > 0 then `Zobject` = `Zground` + `HEIGHT`
- If `HEIGHT` = 0 then `Zobject` = `Zground` and Warning message *"Be carreful, some buildings are 0 meter high"*
- If `HEIGHT` null or < 0 then Error message *"Not possible to determine Z coordinates"*

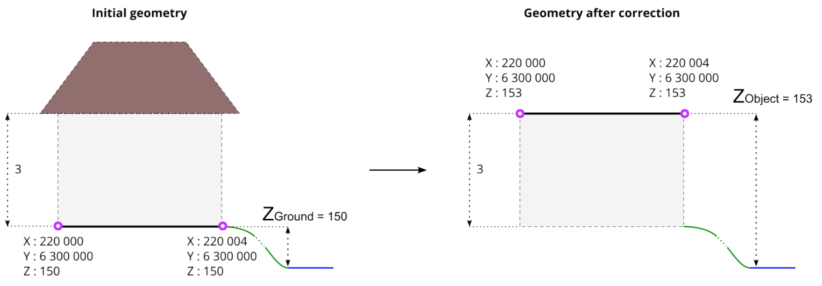
There is no DEM layer

- If `HEIGHT` > 0 then `Zobject` = `HEIGHT`
- If `HEIGHT` = 0 then `Zobject` = 0 and Warning message *"Be carreful, some buildings are 0 meter high"*
- If `HEIGHT` null or < 0 then Error message *"Not possible to determine Z coordinates"*

2. The geometry has a Z coordinate

- **The Z coordinate correspond to `Zobject`**
  - It's ok, your data is already ready to be used by NoiseModelling
- **The Z coordinate correspond to `Zground`**
  - You are invited to correct `Z` value(s) by changing the information by yourself or by using the dedicated WPS block called `Correct_building_altitude`

Below is an example with a initial geometry (coordinates are exprimed in French Lambert 93 (EPSG:2154) system) with a `Zground` value coupled with `HEIGHT` information. After correction, the geometry has a correct Z value, which corresponds to `Zobject`.



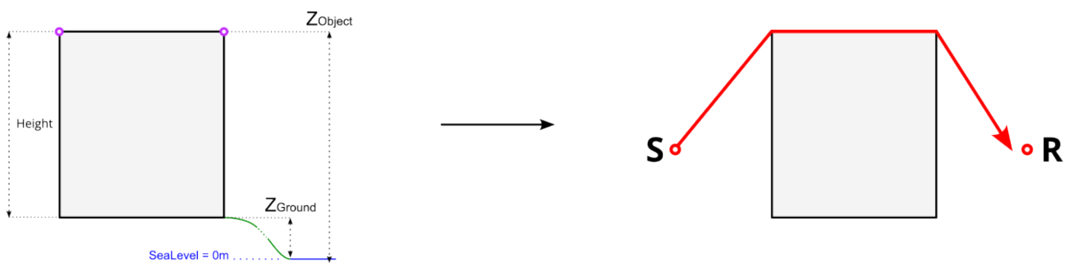
5.5.4 Ray path

Depending on the building modelisation and the `Zobject` you have, the acoustic wave path will differ.

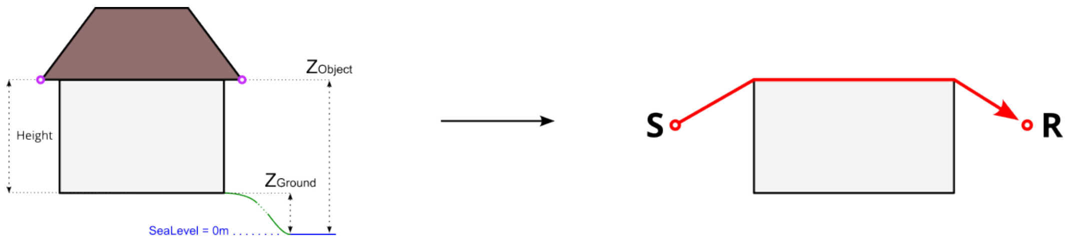
In the 4 examples below,

- the left-hand side is dealing with the building's modelisation. Pink circles represents the vertices of the geometry
- the right-hand side represents the corresponding path of the ray (in red), from the sound source (S) to the receiver (R) and how the building (in grey) is "understood" by NoiseModelling.

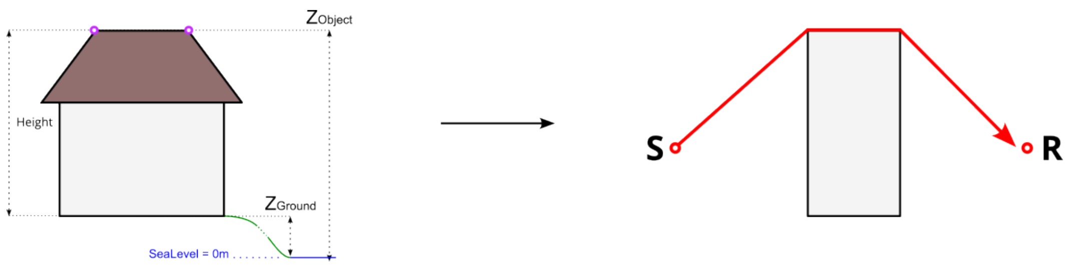
Case 1 : there is no roof



Case 2 : `Zobject` is on the gutter level



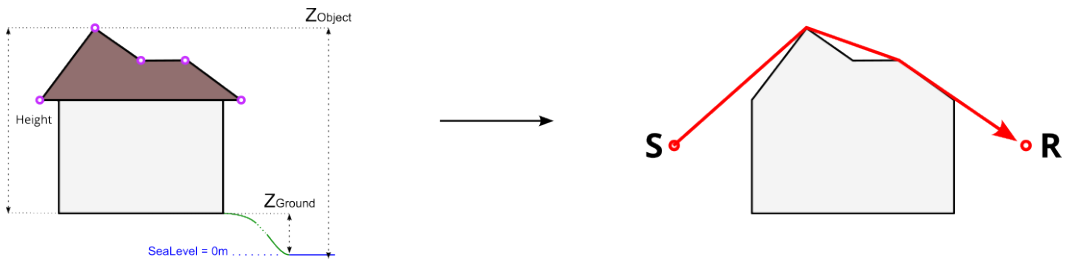
Case 3 : `Zobject` is on top ot the roof



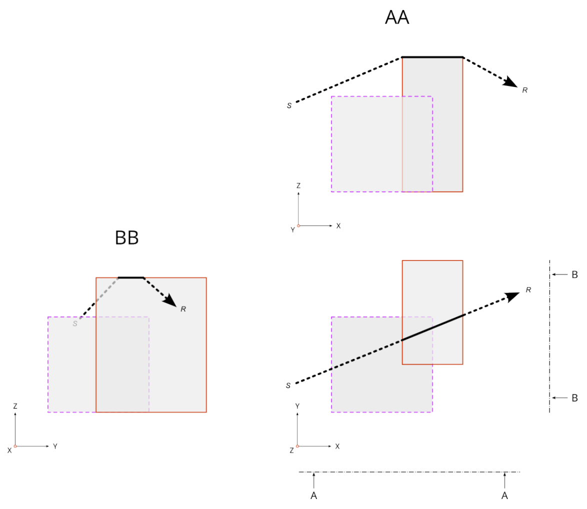
Case 4 : Complex roof shape

5.5.5 Topology

In the table `BUILDINGS` there is no topological constraint. Even if it is not recommended, this means that NoiseModelling accepts that the buildings overlap. In this case, the highest points and edges will be retained for the determination of the wave path.



The figure below illustrate this possibility with two buildings that overlaps. The wave is going from the source `S` to the receveiver `R`.



5.6 Roads

NoiseModelling is a tool for producing noise maps. To do so, at different stages of the process, the application needs input data, respecting a strict formalism.

Below we describe the table `ROADS`, dealing with the roads network.

The other tables are accessible via the left menu in the `Input tables & parameters` section.

5.6.1 Table definition

Warning:

- In the list below, the columns noted with `*` are mandatory

- This description is only valid for `Noise_level_from_traffic` and `Road_Emission_from_Traffic` WPS scripts. For the other WPS scripts, it is necessary to refer to the description of their input data

Note: In the list below, some columns are suffixed with the letters `D`, `E` and `N`. This correspond to `Day` (6-18h), `Evening` (18-22h) and `Night` (22-6h) periods. A column is expected for each of them.

- **`THE_GEOM`** *
  - Description: Geometry of the roads (`LINESTRING` or `MULTILINESTRING`)
  - Type: Geometry
- **`PK`** *
  - Description: An identifier (PRIMARY KEY)
  - Type: Integer
- **`LV_D`, `LV_E`, `LV_N`**
  - Description: Hourly average light vehicle count
  - Type: Double
- **`MV_D`, `MV_E`, `MV_N`**
  - Description: Hourly average medium heavy vehicles, delivery vans > 3.5 tons, buses, touring cars, *etc.* with two axles and twin tyre mounting on rear axle count
  - Type: Double
- **`HGV_D`, `HGV_E`, `HGV_N`**
  - Description: Hourly average heavy duty vehicles, touring cars, buses, with three or more axles count
  - Type: Double
- **`WAV_D`, `WAV_E`, `WAV_N`**
  - Description: Hourly average mopeds, tricycles or quads 50 cc count
  - Type: Double
- **`WBV_D`, `WBV_E`, `WBV_N`**
  - Description: Hourly average motorcycles, tricycles or quads > 50 cc count
  - Type: Double
- **`LV_SPD_D`, `LV_SPD_E`, `LV_SPD_N`**
  - Description: Hourly average light vehicle speed *(km/h)*
  - Type: Double
- **`MV_SPD_D`, `MV_SPD_E`, `MV_SPD_N`**
  - Description: Hourly average medium heavy vehicles speed *(km/h)*
  - Type: Double
- **`HGV_SPD_D`, `HGV_SPD_E`, `HGV_SPD_N`**
  - Description: Hourly average heavy duty vehicles speed *(km/h)*

  - Type: Double
- **`WAV_SPD_D`, `WAV_SPD_E`, `WAV_SPD_N`**
  - Description: Hourly average mopeds, tricycles or quads 50 cc speed *(km/h)*
  - Type: Double
- **`WBV_SPD_D`, `WBV_SPD_E`, `WBV_SPD_N`**
  - Description: Hourly average motorcycles, tricycles or quads > 50 cc speed *(km/h)*
  - Type: Double
- **`PVMT`**
  - Description: CNOSSOS road pavement identifier *(Default* `DEF` *)* (See NM possible values)
  - Type: Varchar
- **`TEMP_D`, `TEMP_E`, `TEMP_N`**
  - Description: Average Day, Evening and Night Celsius temperature (°C) *(Default 20)*
  - Type: Double
- **`TS_STUD`**
  - Description: A limited period (`Ts`) (in months) over the year where a average proportion (`pm`) of light vehicles are equipped with studded tyres [0-12]
  - Type: Double
- **`PM_STUD`**
  - Description: Average proportion of vehicles equipped with studded tyres during `TS_STUD` period [0-1]
  - Type: Double
- **`JUNC_DIST`**
  - Description: Distance to the junction *(in meters)*. When approaching less than 100m from a junction, it is advisable to subdivide the section into 10m pieces and calculate the distance from the centroid of this sub-section to the junction. This allows for a finer calculation.
  - Type: Double
- **`JUNC_TYPE`**
  - **Description: Integer defining the type of junction**
    * `0` : None
    * `1` : A crossing with traffic lights
    * `2` : A roundabout
  - Type: Integer
- **`SLOPE`**
  - Description: Slope (in %) of the road section. If the column is not filled in, the `LINESTRING` Z-values will be used to calculate the slope and the traffic direction (`WAY` column) will be force to `3` (bi-directional)
  - Type: Double
- **`WAY`**

- **Description: Integer defining the way of the road section.**
  - `1` = One way road section and the traffic goes in the same way that the slope definition you have used
  - `2` = One way road section and the traffic goes in the opposite way that the slope definition you have used
  - `3` = Bi-directional traffic flow, the flow is split into two components and correct half for uphill and half for downhill
- Type: Integer

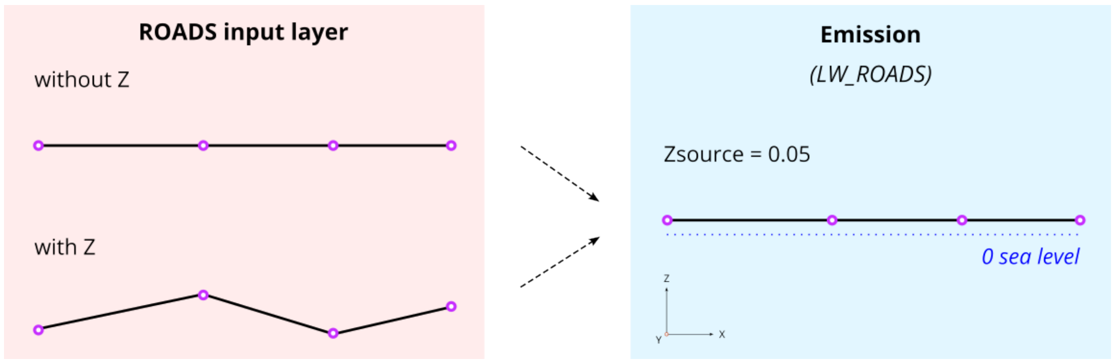
5.6.2 Geometry modelling

In NoiseModelling, road geometries are used as a medium for road noise emission and propagation.

Emission

According to CNOSSOS-EU, emissions from road traffic should be 5cm above the ground.

You can create your own emmission layer or use the dedicated NoiseModelling block called `Road_Emission_from_Traffic.groovy`. In this script, the table `ROADS` is used to create the emission table `LW_ROADS`. As a consequence, whether or not your roads have a Z value in `ROADS`, NoiseModelling forces a `Zsource` value of 5cm in `LW_ROADS`.

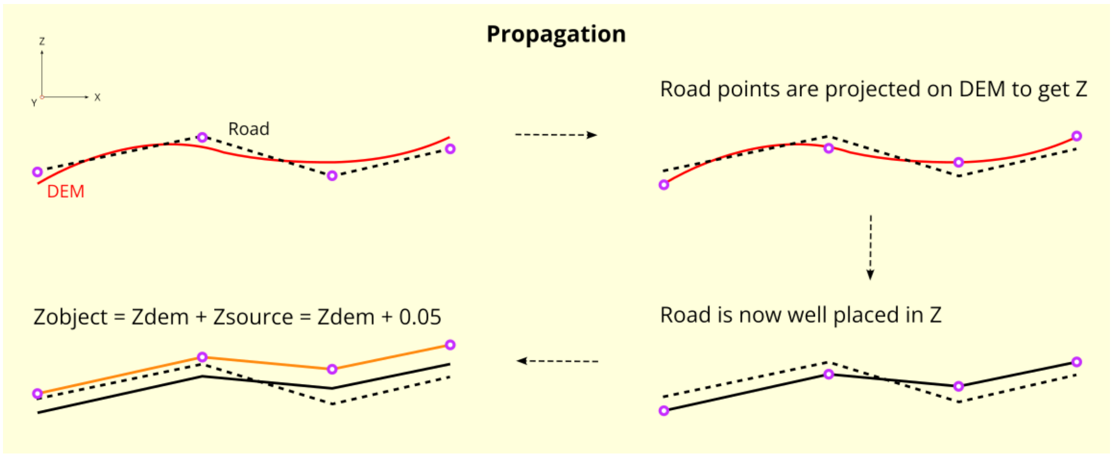


Warning: Whether you have Z values, the emission layer must be at an altitude of 5cm (above sea level) : `Zsource` = 0.05

Note: Z values in the input layer are only used to calculate the slope

Propagation

Whether you use your own sources or those calculated by NoiseModelling, the propagation step will consist of deducing the altitude from the DEM and adding the emission height (5cm).


Propagation
Z
X
Y
Road
DEM
Road points are projected on DEM to get Z
Road is now well placed in Z
Zobject = Zdem + Zsource = Zdem + 0.05

Warning:

- `Zobject = Zdem + Zsource = Zdem + 0.05`
- If there is no DEM, the altitude will be equal to 5cm (`Zobject = 0.05`)
- If your `ROADS` table has accurate Z values, you are invited to enrich your DEM with this information before doing the propagation step. See *DEM* section for more information.

Note: Z values in the input layer are only used to calculate the slope. They are not used to force the DEM

In this context, the roads geometry can be in 2D or in 3D. In both cases, Z information is not taken into account during emission or propagation steps.

5.7 Railways

NoiseModelling is a tool for producing noise maps. To do so, at different stages of the process, the application needs input data, respecting a strict formalism.

Below are described the tables `RAIL_SECTIONS` and `RAIL_TRAFFIC`.

The other tables are accessible via the left menu in the `Input tables & parameters` section.

Warning: In the lists below, the columns noted with `*` are mandatory

5.7.1 Railways sections

- Table name : `RAIL_SECTIONS`
- Description: contains all the sections of railways

Table definition

- **THE_GEOM** *
  - Description: Railway's geometry
  - Type: Geometry (`LINESTRING` or `MULTILINESTRING`)
- **IDSECTION** *
  - Description: A section identifier (PRIMARY KEY)
  - Type: Integer
- **NTRACK** *
  - Description: Number of tracks
  - Type: Integer
- **TRACKSPD** *
  - Description: Maximum speed on the section *(in km/h)*
  - Type: Double
- **TRANSFER**
  - **Description: Track transfer function identifier**
    - `1` = Mono-bloc sleeper on soft rail pad
    - `2` = Mono-bloc sleeper on medium rail pad
    - `3` = Mono-bloc sleeper on stiff rail pad
    - `4` = Bi-bloc sleeper on soft rail pad
    - `5` = Bi-bloc sleeper on medium rail pad
    - `6` = Bi-bloc sleeper on stiff rail pad
    - `7` = Wooden sleeper (Traverse en bois)
  - Type: Integer
- **ROUGHNESS**
  - **Description: Rail roughness identifier**
    - `1` = Classic lines
    - `2` = TGV (for France) lines
  - Type: Integer
- **IMPACT**
  - **Description: Impact noise coefficient identifier**
    - `0` = No impact
    - `1` = Single joint, switch or crossing per 100 m

    - Type: Integer

- **CURVATURE**
    - **Description: Section's curvature identifier**
        * 0 = R > 500 m
        * 1 = 300 m < R < 500 m
        * 2 = R < 300 m
    - Type: Integer
- **BRIDGE**
    - **Description: Bridge transfer function identifier**
        * 0 = Any type of track or bridge except metal bridges with unballasted tracks
        * 1 = Metal bridges with unballasted tracks + 5dB
    - Type: Integer
- **TRACKSPC**
    - Description: Commercial speed on the section *(in km/h)*
    - Type: Double
- **ISTUNNEL**
    - Description: Indicates whether the section is a tunnel or not (0 = no / 1 = yes)
    - Type: Boolean

Geometry modelling

The modeling of the geometry is identical to the road's one (see "*Roads*" page). The only difference is that the affected height is not 5cm by default. It depends on the model used (*e.g* in CNOSSOS: rolling noise = 0.05m / aerodynamic noise = 4m).

5.7.2 Railways traffic

- Table name : RAIL_TRAFFIC
- Description: contains all the railways traffic

Table definition

- **IDTRAFFIC** *
    - Description: A traffic identifier (PRIMARY KEY)
    - Type: Integer
- **IDSECTION** *
    - Description: A section identifier, refering to RAIL_SECTIONS table
    - Type: Integer

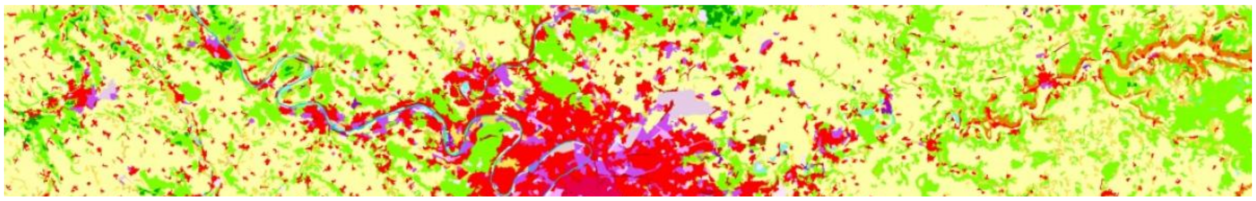
- **TRAINTYPE** *
  - Description: Type of vehicle, listed in the Rail_Train_SNCF_2021.json file *(mainly for french SNCF)*
  - Type: Varchar
- **TRAINSPD** *
  - Description: Maximum train speed *(in km/h)*
  - Type: Double
- **TDAY**
  - Description: Hourly average train count, during the day (6-18h)
  - Type: Integer
- **TEVENING**
  - Description: Hourly average train count, during the evening (18-22h)
  - Type: Integer
- **TNIGHT**
  - Description: Hourly average train count, during the night (22-6h)
  - Type: Integer

5.8 Ground surfaces

NoiseModelling is a tool for producing noise maps. To do so, at different stages of the process, the application needs input data, respecting a strict formalism.

Below we describe the table `GROUND`, dealing with the land use type, with an associated ground absorption coefficient (G).

The other tables are accessible via the left menu in the `Input tables & parameters` section.



5.8.1 Table definition

Warning: In the list below, the two columns are mandatory

- **THE_GEOM**
  - Description: 2D geometry of the surfaces (`POLYGON` or `MULTIPOLYGON`)
  - Type: Geometry
- **G**

  - Description: acoustic ground's absorption (from `0` : very hard to `1` : very soft - *see table below*)
  - Type: Double

Table 1: G values for different types of ground (extracted from "Common Noise Assessment Methods in Europe (CNOSSOS-EU)", p86)

Description	G
Very soft (snow or moss-like)	1
Soft forest floor (short, dense heather-like or thick moss)	1
Uncompacted, loose ground (turf, grass, loose soil)	1
Normal uncompacted ground (forest floors, pasture field)	1
Compacted field and gravel (compacted lawns, park area)	0.7
Compacted dense ground (gravel road, car park)	0.3
Hard surfaces (most normal asphalt, concrete)	0
Very hard and dense surfaces (dense asphalt, concrete, water)	0

5.8.2 Topology

At a given point, there can only be one value of G. Consequently, in the `GROUND` table, the geometries must not overlap.

5.9 DEM

NoiseModelling is a tool for producing noise maps. To do so, at different stages of the process, the application needs input data, respecting a strict formalism.

Below we describe the table `DEM`, dealing with the Digital Elevation Model matrix.

The other tables are accessible via the left menu in the `Input tables & parameters` section.

Note: If your DEM is in raster, please use the `Import_Asc_File` WPS script which will format your DEM in the right format

5.9.1 Table definition

Warning: In the list below, the column noted with `*` is mandatory

- **`THE_GEOM`** *
  - Description: 3D point of the matrix (`POINT` or `MULTIPOINT`). Z coordinate represent the altitude from the 0 sea level.
  - Type: Geometry

5.9.2 DEM enrichment

If you have input data with a good elevation quality (better than the DEM one) / higher density and if you are comfortable with GIS tools, you are invited to enrich your DEM so that it takes into account the structuring elements of the territory.

Note: You can find dedicated scripts (*e.g* `Enrich_DEM_with_lines`, `...`) in the `Geometric Tools` section of the left-side menu of NoiseModelling

Below is an example of DEM enrichment using road network:

1. Roads (red lines) are inserted into the DEM (blue points),
2. Roads are densified in order to have more points (red) (for example a new point every 5m along the road). For each new point, the altitude (`Za1`, `Za2`, `...`) is deduced from a linear interpolation between input vertices (`Za`, `Zb`, `Zc`, `...`),
3. We generate the road platform (pink area), using the road's width or an arbitrary distance (*e.g* 3m). The densified points (green), which keep the interpolated altitudes, are placed along this new plateform,
4. All the DEM points that intersects the road platform are removed from the layer.

5.10 Directivity

NoiseModelling is a tool for producing noise maps. To do so, at different stages of the process, the application needs input data, respecting a strict formalism.

Below we describe the table `DIRECTIVITY`, containing all the directivity parameters.

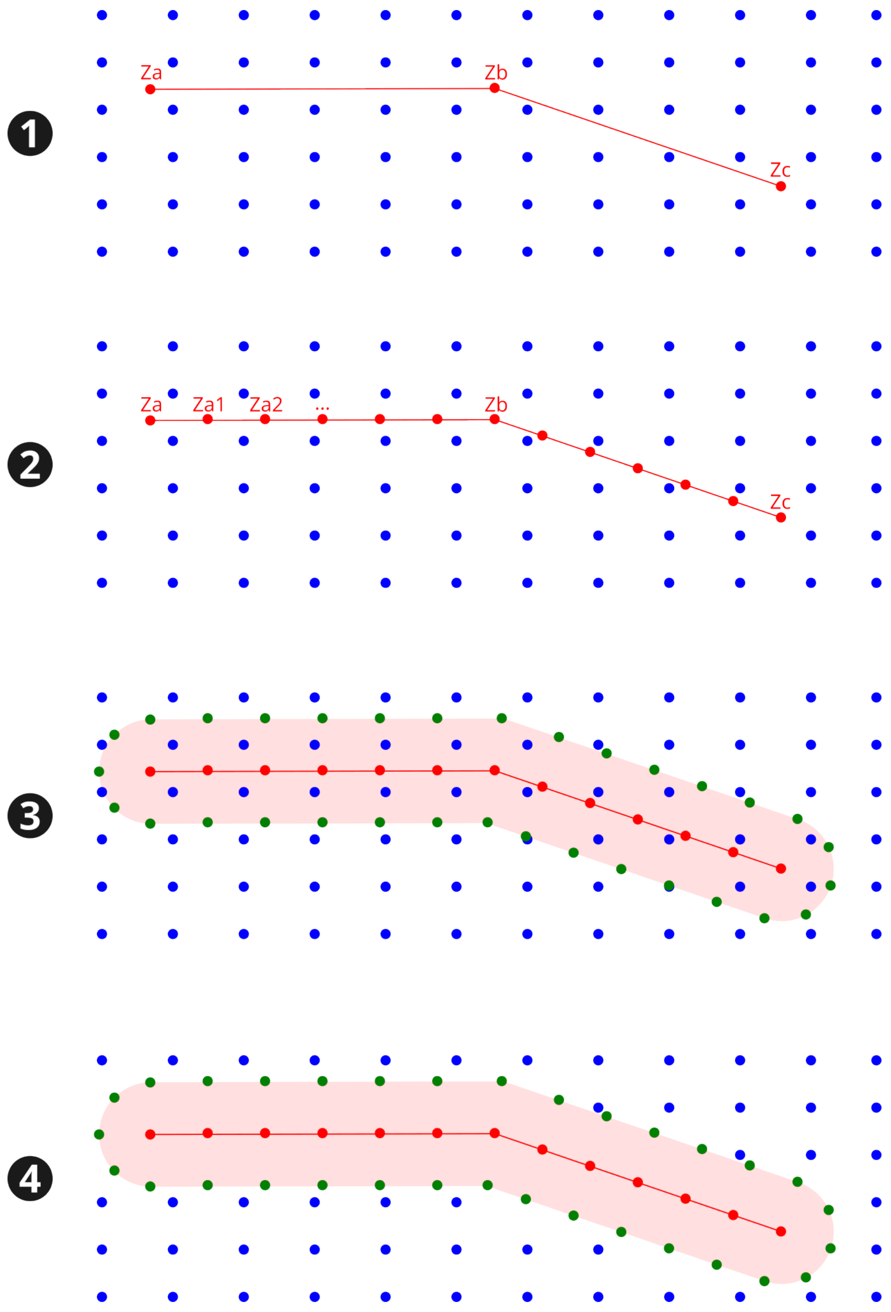
The other tables are accessible via the left menu in the `Input tables` section.

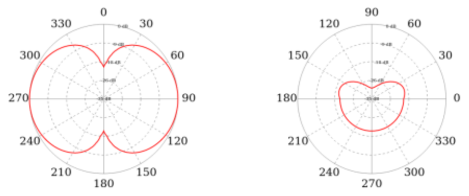
Note: If you want to see how to use this table, have a look to the tutorial "*Noise Map from Point Source - GUI*" , in the section `Step 5 (bonus): Change the directivity`

5.10.1 Table definition

Warning: In the list below, the columns noted with `*` are mandatory

- **`DIR_ID`** *
  - Description: identifier of the directivity sphere
  - Type: Integer
- **`THETA`**
  - Description: vertical angle in degrees, 0 (front), -90 (bottom), 90 (top), from -90 to 90
  - Type: Double
- **`PHI`**
  - Description: horizontal angle in degrees, 0 (front) / 90 (right), from 0 to 360


❶
Za
Zb
Zc
❷
Za
Za1
Za2
...
Zb
Zc
❸
❹


0
30
60
90
120
150
180
210
240
270
300
330
90
60
30
0
330
300
270
240
210
180
150
120

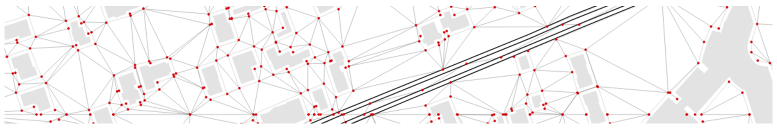
  - Type: Double
- **LW63**
  - Description: attenuation levels in dB for 63 Hz
  - Type: Double
- **LW125**
  - Description: attenuation levels in dB for 125 Hz
  - Type: Double
- **LW250**
  - Description: attenuation levels in dB for 250 Hz
  - Type: Double
- **LW500**
  - Description: attenuation levels in dB for 500 Hz
  - Type: Double
- **LW1000**
  - Description: attenuation levels in dB for 1000 Hz
  - Type: Double
- **LW2000**
  - Description: attenuation levels in dB for 2000 Hz
  - Type: Double
- **LW4000**
  - Description: attenuation levels in dB for 4000 Hz
  - Type: Double
- **LW8000**
  - Description: attenuation levels in dB for 8000 Hz
  - Type: Double

5.11 Receivers

NoiseModelling is a tool for producing noise maps. To do so, at different stages of the process, the application needs input data, respecting a strict formalism.

Below we describe the table `RECEIVERS`, dealing with the receivers.

The other tables are accessible via the left menu in the `Input tables & parameters` section.



5.11.1 Table definition

Warning: The two following columns are mandatory

- **PK**
  - Description: receiver's unique identifier.
  - Type: Integer - Primary Key
- **THE_GEOM**
  - Description: 3D receiver's geometry. Z coordinate correspond to the receiver's height (relative to ground altitude)
  - Type: Geometry (`POINT` or `MULTIPOINT`)

If you are working with receivers based on buildings (*e.g* 50 cm around the building's facades - see `Building_grid` script), your `RECEIVERS` table will need this additional column:

- **BUILD_PK**
  - Description: building's Primary Key (`PK`), allowing to link the receivers with their building
  - Type: Integer

5.11.2 Parameters

Below are listed the most important input parameters that may be found in the scripts dealing with receivers generation (*e.g* `Building_grid`, `Delaunay_grid`, . . .) (see `Receivers` section in the left-side menu of NoiseModelling).

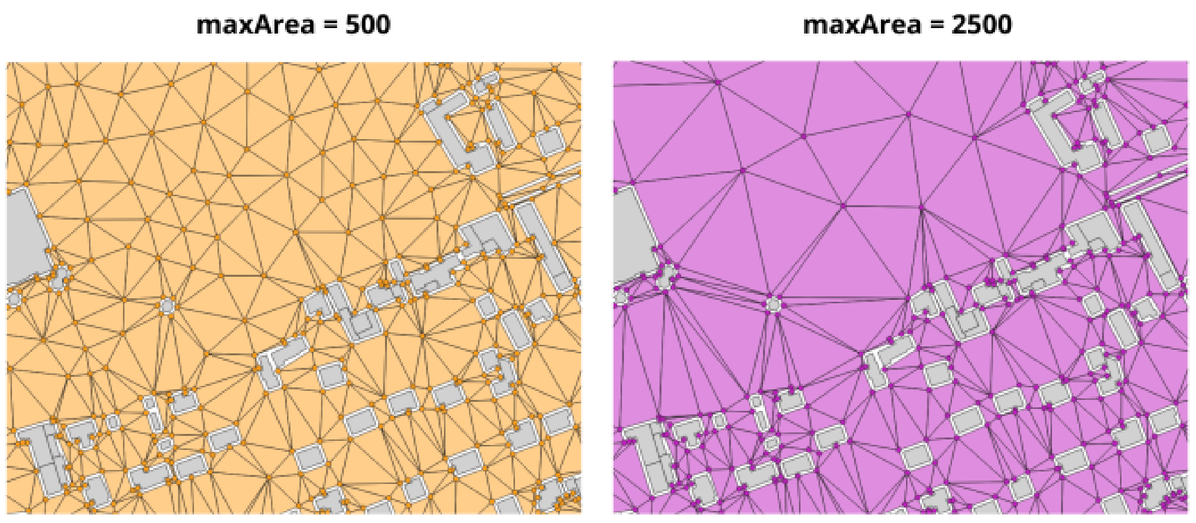
These parameters can be mandatory or optional. When necessary, we indicates the default values and those we recommend (from an acoustic point of view).

Maximum area

- Parameter name: `maxArea`
- Description: Set Maximum Area. No triangles larger than the provided area will be created. Smaller area will create more receivers (square meters)
- Type: Double
- Default value: `2500`
- Recommanded value: `2500`

Maximum cell size

- Parameter name: `maxCellDist`
- Description: Maximum distance used to split the domain into sub-domains. In a logic of optimization of processing times, it allows to limit the number of objects (buildings, roads, . . .) stored in memory during the Delaunay triangulation (meters)
- Type: Double
- Default value: `600`



- Recommanded value:

Road width

- Parameter name: `roadWidth`
- Description: Set Road Width. No receivers closer than road width distance will be created (meters)
- Type: Double
- Default value: `2`
- Recommanded value:

Height

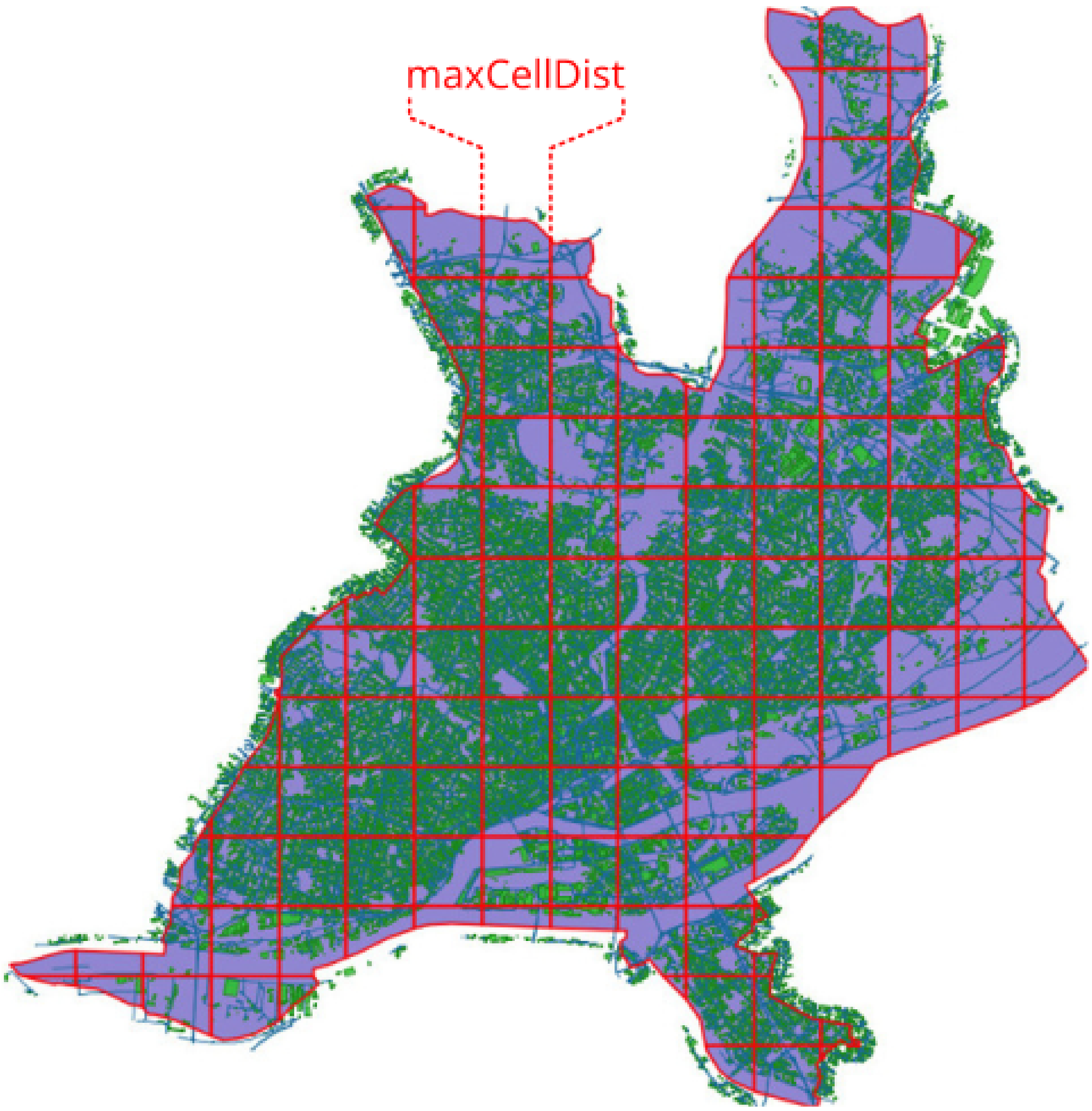
- Parameter name: `height`
- Description: Receiver height relative to the ground (meters)
- Type: Double
- Default value: `4`
- Recommanded value:

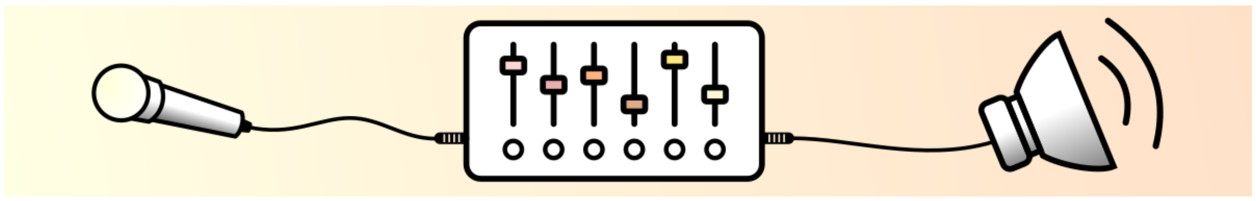
5.12 Acoustic parameters

In the different WPS scripts of NoiseModelling, you will find many input parameters, mandatory or optional.

Below we list the most important ones, indicating, where necessary, the default values and those we recommend (from an acoustic point of view).

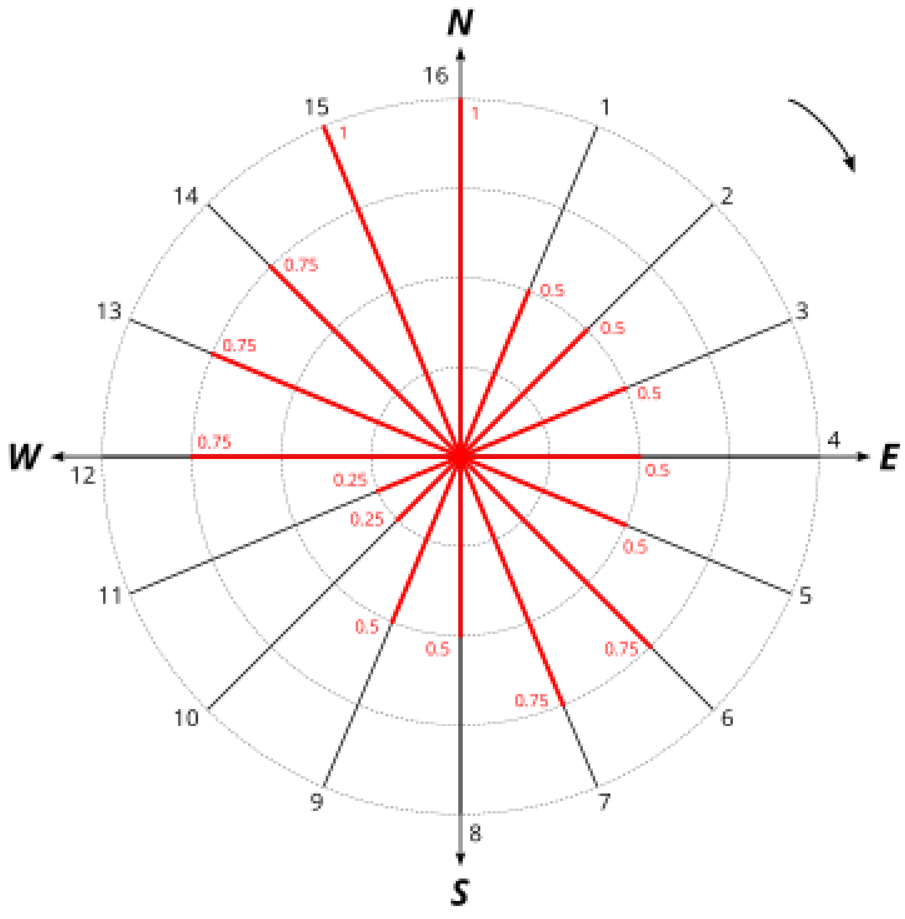
The following parameters may be found in the scripts dealing with noise emission or propagation (*e.g* `Noise_level_from_traffic`, `Noise_level_from_source`\`, ...)


maxCellDist



5.12.1 Probability of occurrences

- Parameter name: `confFavorableOccurrencesXXXXX` (with `XXXXX` = evening, day, night, . . .)
- Description: Comma-delimited string containing the probability ([0,1]) of occurrences of favourable propagation conditions. Follow the clockwise direction. The north slice is the last array index (n°16 in the schema below) not the first one
- Type: Double
- Default value: `0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5`
- Recommanded value:



confFavorableOccurrences = 0.5, 0.5, 0.5, 0.5, 0.5, 0.75, 0.75, 0.5, 0.5, 0.25, 0.25, 0.75, 0.75, 0.75, 1, 1

5.12.2 Relative humidity

- Parameter name: `confHumidity`
- Description: Humidity for noise propagation (%) [0,100]
- Type: Double
- Default value: `70`

- Recommanded value: depends on the average conditions at the location where you perform the simulation

5.12.3 Air temperature

- Parameter name: `confTemperature`
- Description: Air temperature (°C)
- Type: Double
- Default value: `15`
- Recommanded value: depends on the average conditions at the location where you perform the simulation

5.12.4 Order of reflexion

- Parameter name: `confReflOrder`
- Description: Maximum number of reflections to be taken into account. Warning: adding 1 order increases the processing time significantly
- Type: Integer
- Default value: `1`
- Recommanded value: `1` or `2`

5.12.5 Diffraction on horizontal edges

- Parameter name: `confDiffHorizontal`
- Description: Compute or not the diffraction on horizontal edges
- Type: Boolean
- Default value: `False`
- Recommanded value: `True`

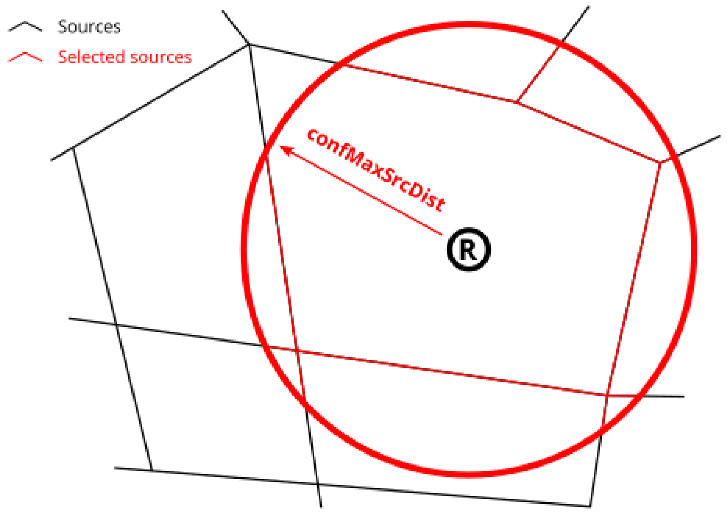
5.12.6 Diffraction on vertical edges

- Parameter name: `confDiffVertical`
- Description: Compute or not the diffraction on vertical edges. Following Directive 2015/996, enable this option for rail and industrial sources only
- Type: Boolean
- Default value: `False`
- Recommanded value:

5.12.7 Maximum source-receiver distance

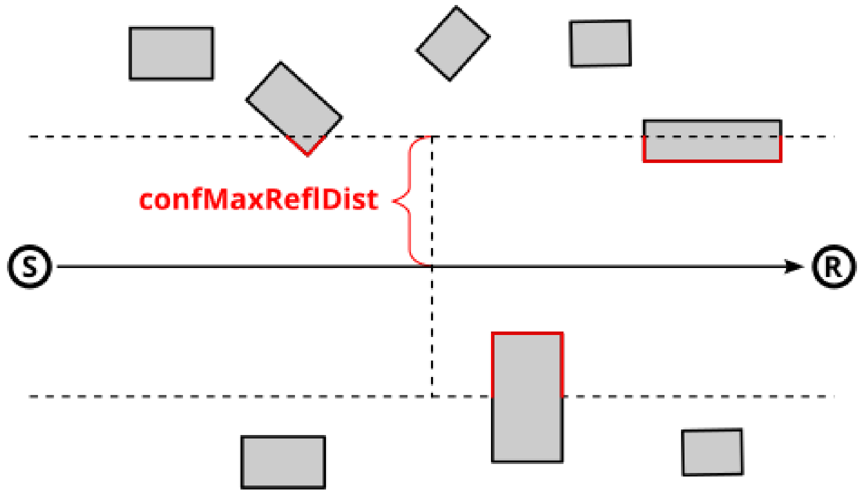
- Parameter name: `confMaxSrcDist`
- Description: Maximum distance between source and receiver (meters)
- Type: Double

- Default value: `150`
- Recommanded value: Between `500` and `800`



5.12.8 Maximum source-reflexion distance

- Parameter name: `confMaxReflDist`
- Description: Maximum search distance of walls / facades from the "Source-Receiver" segment, for the calculation of specular reflections (meters)
- Type: Double
- Default value: `50`
- Recommanded value: Between `350` and `800`



5.12.9 Wall absorption coefficient

- Parameter name: `paramWallAlpha`
- Description: Wall absorption coefficient [0,1] (between `0` : "fully absorbent" and `1` : "fully reflective")
- Type: Double
- Default value: `0.1`
- Recommanded value: `0.1`

5.12.10 Separate receiver level by source identifier

- Parameter name: `confExportSourceId`
- Description: Keep source identifier in output in order to get noise contribution of each noise source
- Type: Boolean
- Default value: `False`
- Recommanded value:

5.12.11 Thread number

- Parameter name: `confThreadNumber`
- Description: Number of thread to use on the computer
- Type: Integer
- Default value: `0` (`0` = Automatic. Will check the number of cores and apply -1. (*e.g*: 8 cores = 7 cores will be used))
- Recommanded value: `0`

5.13 Requirements

5.13.1 Java environment

Since NoiseModelling is developped with the Java langage, you will need to install the Java Runtime Environment (JRE) on your computer to use the application.

Warning: Only version 11.x of Java is compatible with NoiseModelling 4.x. Unfortunatelay, former or newer versions are not compatible with NoiseModelling 4.x.

Windows

If you are launching NoiseModelling thanks to the `NoiseModelling_xxx_install.exe` file, the JRE is already inside, so **you don't have anything to do**.

If you are not using the `.exe` file, you have to launch NoiseModelling thanks to the `` `...\bin\startup_windows.bat `` file (in the `NoiseModelling_xxx.zip` release file). In this case, Java v11.x has to be installed before.

1. Download and install Java: choose between OpenJDK or Oracle versions.
2. You can check if `JAVA_HOME` environnement variable is well settled to your last installed Java folder using `echo %JAVA_HOME%` in your command prompt. You should have a result similar to `C:\\Program Files (x86)\\Java\\jre1.8.x_x\\`.
3. If you don't have this result, it is probably because your `JAVA_HOME` environnement variable is not well settled. To set you `JAVA_HOME` environnement variable you can adapt (with `x` the JAVA version number) you installed and use the following command line : `setx JAVA_HOME "C:\\Program Files (x86)\\Java\\jre.1.8.x_x"` in your command prompt. You can also refer to this document for example.
4. You may have to reboot your command prompt after using the precedent command line before printing again `echo %JAVA_HOME%`.

Warning: The command promprt should print `C:\\Program Files (x86)\\Java\\jre1.11.x_x\\` whithout the bin directory. If `JAVA_HOME` is settled as `C:\\Program Files (x86)\\Java\\jre1.11.x_x\\bin`, it will not work. It should also point to a JRE (Java Runtime Environment) Java environnement and not JDK.

Linux or Mac

If not already done, you have to install the Java version v11.x.

1. Download and install Java: choose between OpenJDK or Oracle versions.
2. You can check if `JAVA_HOME` environnement variable is well settled to your installed v11.x Java folder using `echo $JAVA_HOME` in your command prompt. You should have a result similar to `/usr/lib/jvm/java-11-openjdk-amd64/`.
3. If you don't have this result, it is probably because your `JAVA_HOME` environnement variable is not well settled. In this case, you are invited to follow the steps proposed here.
4. Once done, you may have to reboot your command prompt (or maybe disconnect/reconnect your session) after using the precedent command line before printing again `echo $JAVA_HOME`.

5.14 Get Started - GUI

Below we present a simple application case, allowing you to discover NoiseModelling through its Graphical User Interface .

5.14.1 Step 1: Download NoiseModelling

Download the latest realease of NoiseModelling on Github.

- Windows: you can directly download and execute the `NoiseModelling_4.x.x_install.exe` installer file *(or you can also follow Linux / Mac instructions below)*
- Linux or Mac: download the `NoiseModelling_4.x.x.zip` file and unzip it into a chosen directory

Warning: The chosen directory can be anywhere, but be sure that you have write access. If you are using the computer of your company, the Program Files folder is probably not a good idea.

Warning: For **Linux** and **Mac** users, please make sure your Java environment is well setted. For more information, please read the page *Requirements*. **Windows** users who are using the `.exe` file are not concerned since the Java Runtime Environment is **already embeded**.

Note: Only from version 3.3, NoiseModelling releases include the user interface described in this tutorial.

5.14.2 Step 2: Start NoiseModelling GUI

As seen in the page "*Architecture*", NoiseModelling can be used through a Graphic User Interface (GUI), thanks to Geoserver and WPS Builder bricks.

In this tutorial, we will use the default and already configured H2GIS database.

Those tools (Geoserver, WPS Builder and H2GIS) are already included in the archive. So you don't have to install them before.

To launch NoiseModelling with GUI, please execute :

- Windows: `NoiseModelling.exe` or `NoiseModelling_xxx\bin\startup_windows.bat`
- Linux or Mac: `NoiseModelling_xxx/bin/startup_linux_mac.sh` *(check authorize file execution in property of this file before)*

and wait until `INFO:oejs.Server:main:Started` is written in your command prompt.

Warning: Depending on your computer configuration, the NoiseModelling launch can take some time. Be patient.

NoiseModelling with GUI is now started.

Tip: NoiseModelling will be open as long as the command window is open. If you close it, NoiseModelling will automatically be closed and you will not be able to continue with the tutorial.

5.14.3 Step 3: Open NoiseModelling GUI

The NoiseModelling GUI is built thanks to the *WPS Builder* brick. To open it, just go to http://localhost:9580 using your preferred web browser.

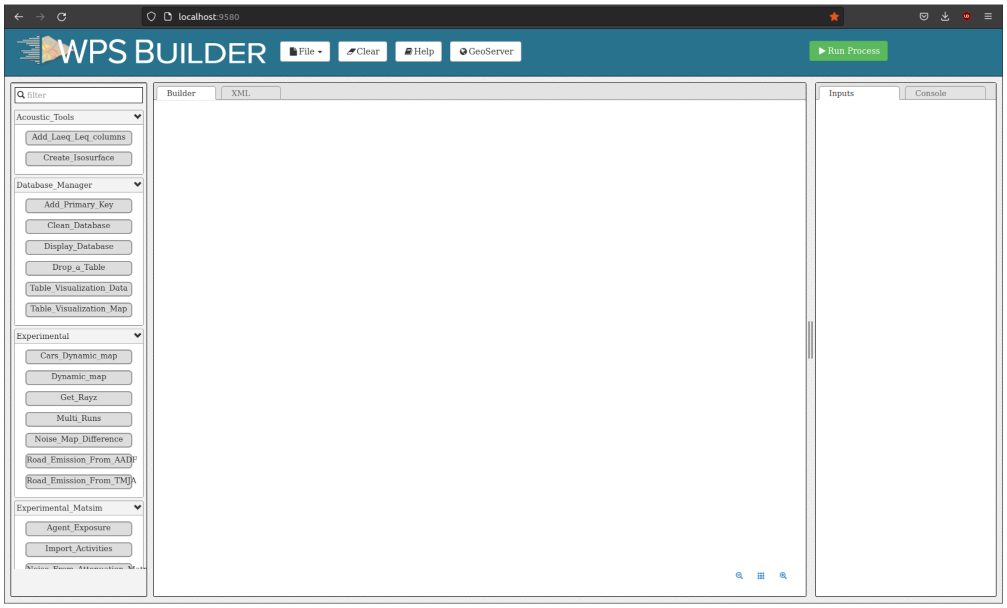
Warning: On former versions of NoiseModelling, the url was: http://localhost:8080/geoserver/web/

You are now ready to discover the power of NoiseModelling!

5.14.4 Step 4: Load input files

To compute your first noise map, you will need to load input geographic files into the NoiseModelling database.

In this tutorial, we have 5 layers, zoomed in the city center of Lorient (France): Buildings, Roads, Ground type, Topography (DEM) and Receivers.



In the `noisemodelling/data_dir/data/wpsdata/` folder, you will find the 5 files (4 shapefiles and 1 geojson) corresponding to these layers.

You can import these layers in your database using the `Import File` or `Import Folder` blocks.

- Drag `Import File` block into the Builder window
- Select `Path of the input File` box and write `data_dir/data/wpsdata/buildings.shp` in the field `PathFile` *(on the right-side column)*
- Then click on `Run Process` after selecting one of the sub-boxes of your process

Repeat this operation for the 4 other files:

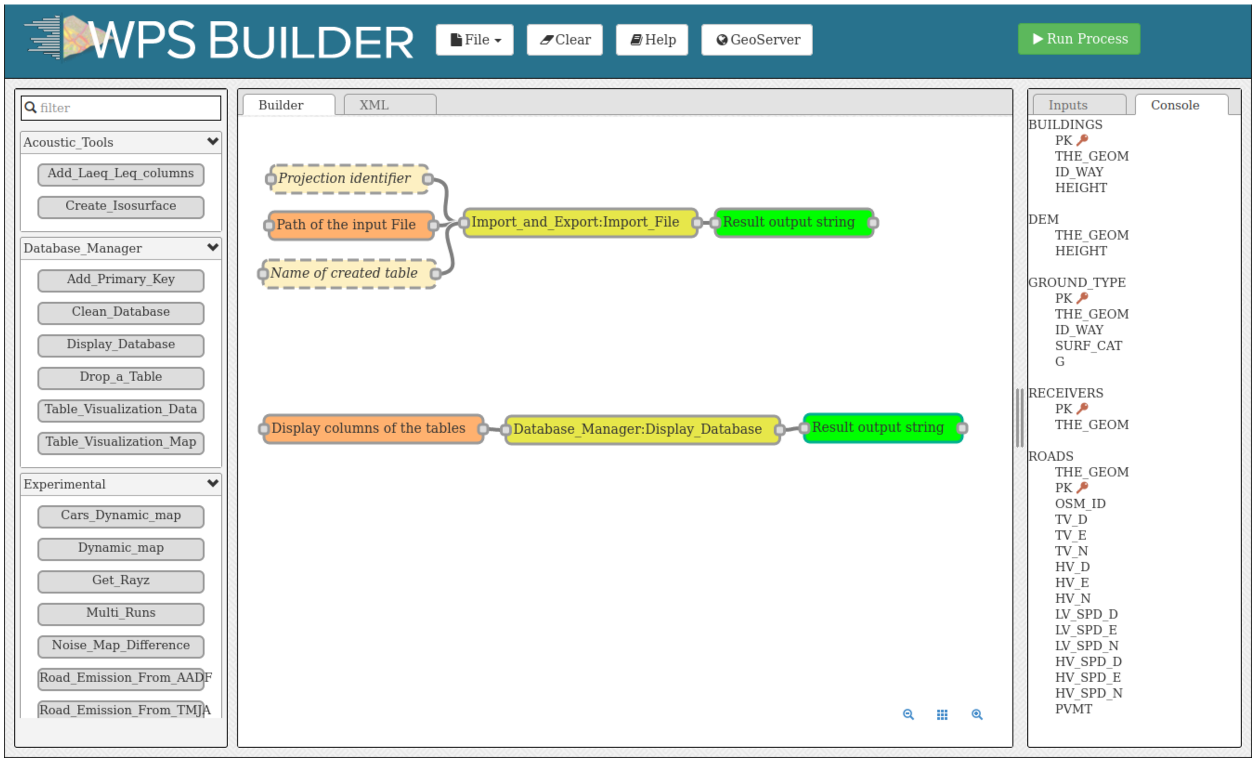
- `data_dir/data/wpsdata/ground_type.shp`
- `data_dir/data/wpsdata/receivers.shp`
- `data_dir/data/wpsdata/ROADS2.shp`
- `data_dir/data/wpsdata/dem.geojson`

Files are uploaded to database when the Console window displays `The table x has been uploaded to database`.

Note:

- If you have the message `Error opening database`, please refer to the note in Step 1.
- The process is supposed to be quick (<5 sec.). In case of out of time, try to restart NoiseModelling (see Step 2).
- Orange blocks are mandatory
- Beige blocks are optional
- If all input blocks are optional, you must modify at least one of these blocks to be able to run the process
- Blocks get solid border when they are ready to run
- Read the *WPS Builder* page for more information

Once done, you can check if the tables have been well imported in the database. To do so, drag/drop and execute the `Display_Database` WPS script (in the "Database_Manager" part). You should see on the right panel the tables list (and their columns if you checked the the option in the `Display columns of the tables` block).



```
BUILDINGS
    PK
    THE_GEOM
    ID_WAY
    HEIGHT

DEM
    THE_GEOM
    HEIGHT

GROUND_TYPE
    PK
    THE_GEOM
    ID_WAY
    SURF_CAT
    G

RECEIVERS
    PK
    THE_GEOM

ROADS
    THE_GEOM
    PK
    OSM_ID
    TV_D
    TV_E
    TV_N
    HV_D
    HV_E
    HV_N
    LV_SPD_D
    LV_SPD_E
    LV_SPD_N
    HV_SPD_D
    HV_SPD_E
    HV_SPD_N
    PVMT
```

5.14.5 Step 5: Run Calculation

To run Calculation you have to drag the block `Noise_level_from_traffic` into WPS Builder window.

Then, select the orange blocks and indicate the name of the corresponding table in your database:

- Building table name : `BUILDINGS`
- Sources table name : `ROADS2`
- Receivers table name : `RECEIVERS`

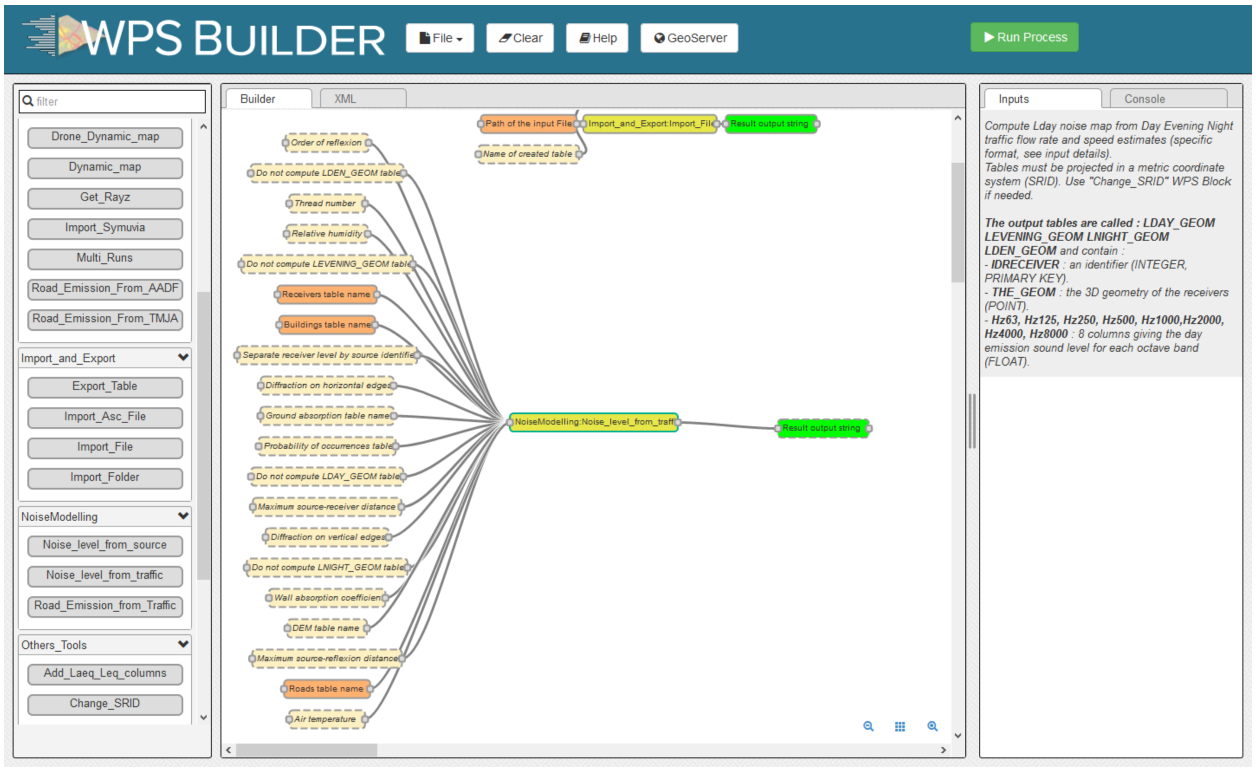
The beige blocks correspond to optionnal parameters (e.g `DEM table name`, `Ground absorption table name`, `Diffraction on vertical edges`, ...).

When ready, you can press `Run Process`.

As a result, the tables `LDAY_GEOM`, `LEVENING_GEOM`, `LNIGHT_GEOM` and `LDEN_GEOM` will be created in your database. These tables correspond to the noise levels, based on receiver points, for the 4 different period of the day.

5.14.6 Step 6: Export (& see) the results

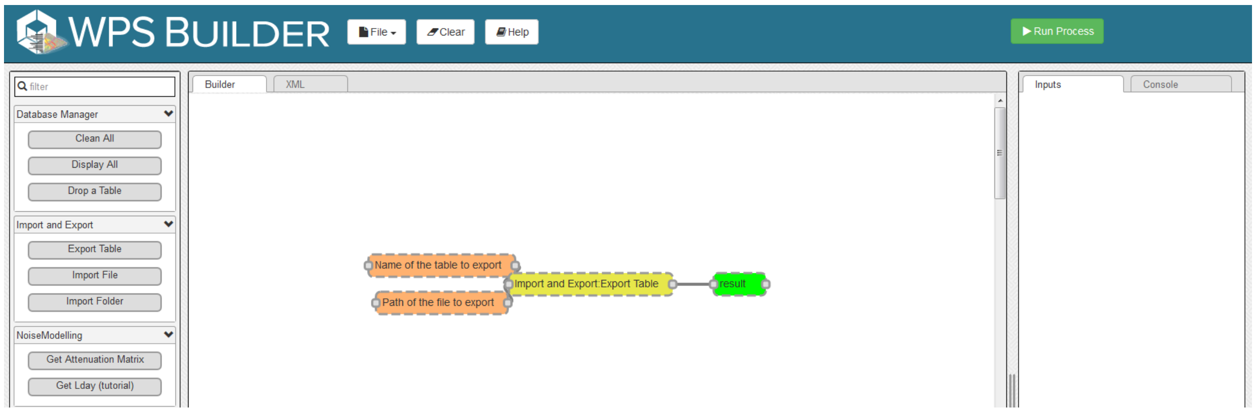
You can now export the output tables *(one by one)* in your favorite export format using `Export_Table` block, giving the path of the file you want to create.



Compute Lday noise map from Day Evening Night traffic flow rate and speed estimates (specific format, see input details).
Tables must be projected in a metric coordinate system (SRID). Use "Change_SRID" WPS Block if needed.

The output tables are called : LDAY_GEOM LEVENING_GEOM LNIGHT_GEOM LDEN_GEOM *and contain :*
*- **IDRECEIVER** : an identifier (INTEGER, PRIMARY KEY).*
*- **THE_GEOM** : the 3D geometry of the receivers (POINT).*
*- **Hz63, Hz125, Hz250, Hz500, Hz1000,Hz2000, Hz4000, Hz8000** : 8 columns giving the day emission sound level for each octave band (FLOAT).*

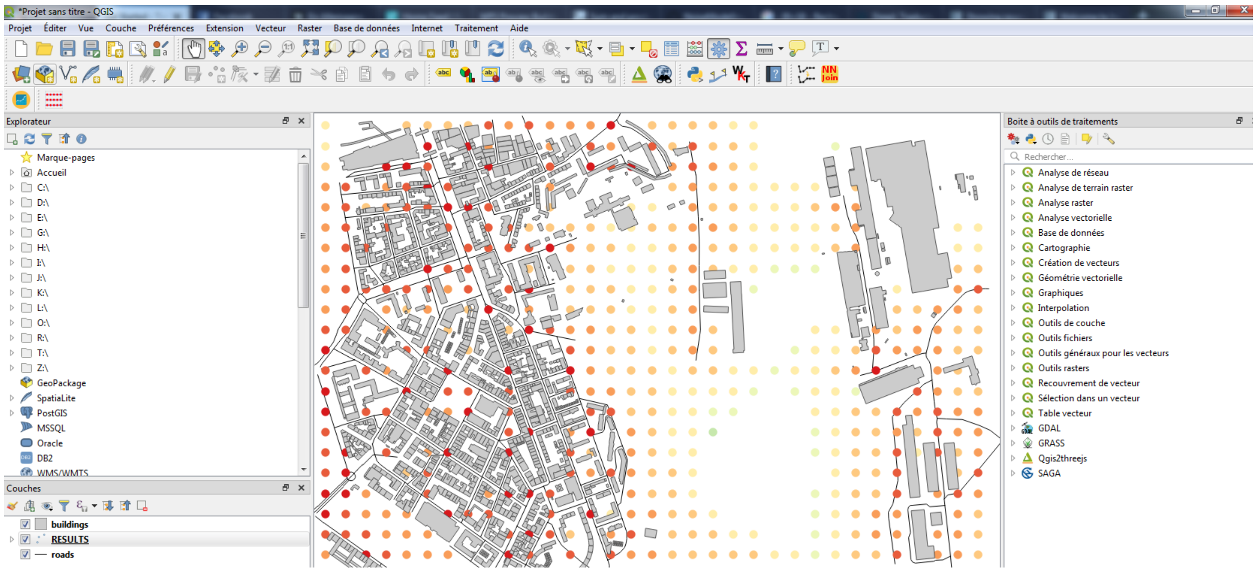
Warning: Dont' forget to add the file extension (*e.g* `c:/home/receivers.geojson` or `c:/home/lday_geom.shp`) (Read more info about file extensions here: *Tutorials - FAQ*)



For example, you can choose to export the tables in `.shp` format. This format can be read with most of GIS tools such as the free and open-source QGIS and SAGA softwares.

Note: For those who are new to GIS and want to get started with QGIS, we advise you to follow this tutorial as a start.

To obtain the following image, use the syling vector options in your GIS and assign a color gradient to `LAEQ` column of your exported `LDAY_GEOM` table.



Tip: Now that you have made your first noise map (congratulations!), you can try again, adding / changing optional parameters to see the differeneces.

5.14.7 Step 7: Know the possibilities

Now that you have finished this introduction tutorial, take the time to read the description of each of the WPS blocks present in your NoiseModelling version.

By clicking on each of the inputs or outputs, you will find a lot of information.

5.15 Noise Map from OSM - GUI

In this tutorial, we are going to produce a noise map, using OpenStreetMap (OSM) data. The exercice will be made through NoiseModelling with Graphic User Interface (GUI).

5.15.1 Prerequisites

- You need at least NoiseModelling v.3.0.6; the best is always to use last release
- We assume you already installed/configured Java and installed NoiseModelling. If not, follow Step 1 in "*Get Started - GUI*" page

Warning: If you have just finished the "*Get Started - GUI*" tutorial, please clean your database with the WPS block `Clean_Database`. Don't forget to check the `Are you sure` check box before running the process.

5.15.2 Step 1: Get OSM data

Note: OpenStreetMap data can be downloaded in various formats. The main ones are *.osm*, *.osm.gz* and *.osm.pbf* (read more). For this example, we will use *.osm.pbf* file, which is a compressed version of *.osm*.

Download OSM data

1. Go to https://extract.bbbike.org/ website. This platform is built on top of OpenStreetMap database and allows you to extract data in a very simple way.
2. In the "Format" drop-down list, choose `Protocolbuffer (PBF)`
3. Give a name to the area you will download *(this information is used to name your extraction request)*
4. Enter your email, so that BBBike will be able to send you the download link once your data are ready *(no data collection for commercial purpose)*.
5. Zoom in on the area you want to download *(be careful, depending on the zoom level, the file you will get may be very heavy)*
6. Click on the `here` icon to create the bounding box. If you click on the bbox, you can then make modification.
7. When ready, click on `extract` button.

In the email you will receive from BBBike, use the link to download your data. You will get a file called `planet_xx.xx,xx.xx.osm.pbf`

Warning: To avoid potential upcoming errors rename the file `planet_xx.xx,xx.xx.osm.pbf` to something simpler (*e.g.* `my_area.osm.pbf`).

Note: Developped by Wolfram Schneider, BBBike is a free of charge service (for non-professional purpose). If you like Wolfram's job and wants to help him support the server costs, you are invited to donate.

Import to the database

To import the `.pbf` file into the NoiseModelling database, we use the `Import_OSM` WPS block (note that this block also allows to load `.osm` or `.osm.gz` files).

1. `Target projection identifier`: enter the corresponding SRID *(see note below)* (*e.g.* `2154` for french Lambert 93)
2. `Path of the OSM file`: enter the adress of your `my_area.osm.pbf` file (*e.g.* `/home/noisemodelling/my_area.osm.pbf`)
3. If needeed, check the 4 other optionnal options
4. When ready, click on the green `Run Process` button

Once done, three tables must be created: `BUILDINGS`, `GROUND` and `ROADS`

Note: About the Coordinate System (EPSG code)

In several input files, you need to specify coordinates, *e.g* road network. You can't use the WGS84 coordinates (i.e. GPS coordinates). Acoustic propagation formulas make the assumption that coordinates are metric. Many countries and regions have custom coordinate system defined, optimized for usages in their appropriate areas. It might be best to ask some GIS specialists in your region of interest what the most commonly used local coordinate system is and use that as well for your data. If you don't have any clue about what coordinate system is used in your region, it might be best to use the Universal Transverse Mercator coordinate system. This coordinate system divides the world into multiple bands, each six degrees width and separated into a northern and southern part, which is called UTM zones (see http://en.wikipedia.org/wiki/UTM_zones#UTM_zone for more details). For each zone, an optimized coordinate system is defined. Choose the UTM zone which covers your region (Wikipedia has a nice map showing the zones) and use its coordinate system.

Here is the map : https://upload.wikimedia.org/wikipedia/commons/e/ed/Utm-zones.jpg

---

Warning:

- The current import script from OpenStreetMap may (in few specific cases) produce geometries incompatible with NoiseModelling. If an area has a problem, try to reduce the area. A much more robust version of this script will be available soon.
- As OSM does not include data on road traffic flows, default values are assigned according to the "Good Practice Guide for Strategic Noise Mapping and the Production of Associated Data on Noise Exposure - Version 2".

5.15.3 Step 2: Visualize OSM data

Now, to be sure that OSM data are corresponding to our need, we can take time to visualize them. To do so, we have various possibilities:

With NoiseModelling GUI

- The contents of the database can be viewed using `Display_Database` WPS script.
- A spatial layer can be visualized using `Table_Visualization_Map` WPS script.
- A data table can be visualized using `Table_Visualization_Data` WPS script.

With H2 or DBeaver client

While NoiseModelling is open, if you are working with the default H2/H2GIS database, you can display your database in both the H2 / H2GIS web interface and DBeaver. To do so, just follow the *Access NoiseModelling database* page.

Export tables into files

- Export a table: It is also possible to export the tables via `Export_Table` WPS script, in Shapefile, CSV or GeoJSON format.
- View the files: Then open these files into your preferred Geographic Information System (*e.g* QGIS, OrbisGIS, . . .). You can then graphically visualize your geometries layer, but also the data contained in it. Take the time to familiarize yourself with your chosen GIS.
- Add a background map: Most of the GIS allow you to add an WMS OSM background map: (see an example with QGIS)

- Change colors: Most of the GIS allow you to change layer colors (*e.g.* `GROUND` layer in green, `BUILDINGS` in gray, `ROADS` in red).

5.15.4 Step 3: Generate a Receiver table

The locations of noise level evaluation points needs to be defined.

Use `Delaunay_Grid` with the previously generated BUILDINGS table as the buildings table and ROADS as *Sources table name*. Other parameters are optional.

Don't forget to view your resulting layer in `WPSBuilder` or in your GIS to check that it meets your expectations.

This processing block will give the possibility to generate a noise map later.

5.15.5 Step 4: Associate emission noise level with roads

The `Road_Emission_from_Traffic` block is used to generate a road layer, called `LW_ROADS`, containing LW emission noise level values in accordance with the emission laws of the CNOSSOS model. The format of the input road layer can be found in the description of the WPS Block.

Don't forget to view your resulting layers *(see Step 2)* to check that it meets your expectations.

5.15.6 Step 5: Source to Receiver Propagation

The `Noise_level_from_source` block allows to generate a layer of receiver points with associated sound levels corresponding to the sound level emitted by the sources (created table `LW_ROADS`) propagated to the receivers according to the CNOSSOS-EU. propagation laws.

5.15.7 Step 6: Create Isosurfaces map

Create an interpolation of levels between receivers points using the block `Create_Isosurface`.

Set `LDEN_GEOM` as `Name of the noise table`.

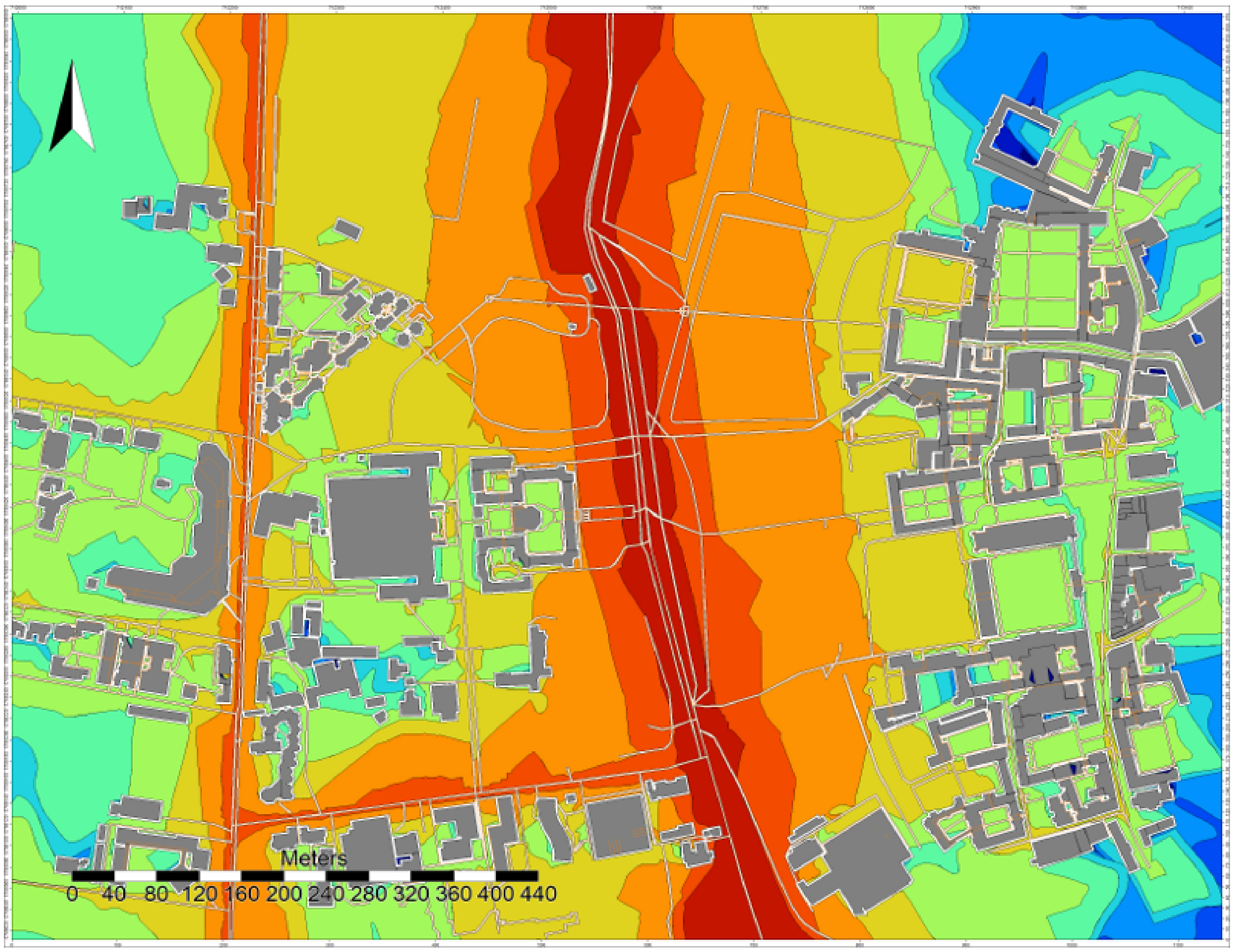
5.15.8 Step 7: View the result

Export

You can then export the output table `CONTOURING_NOISE_MAP` via `Export_Table` in Shapefile or GeoJSON format.

View

You can view this layer in your favorite GIS. You can then apply a color gradient on `ISOLVL` field; the noise level intervals are in `ISOLABEL` field.


Meters
0 40 80 120 160 200 240 280 320 360 400 440

5.16 Noise Map from Point Source - GUI

In this tutorial, we are going to produce a noise map, based on a unique source point. The exercice will be made through NoiseModelling with Graphic User Interface (GUI).

To make it more simple, we will use the data used in the *Get Started - GUI* tutorial.

This tutorial is divided in 5 steps:

1. Create the source point
2. Import data in NoiseModelling
3. Generate the noise map
4. Play with options
5. Take into account directivity

5.16.1 Step 1: Create the source point

To create the source point, we will use the free and opensource GIS software QGIS.

Note: For those who are new to GIS and want to get started with QGIS, we advise you to follow this tutorial as a start.

Load data into QGIS

Once installed, launch QGIS and load the three `buildings.shp`, `roads.shp` and `ground_type.shp` files (that are in the folder `../NoiseModelling_4.0.0/data_dir/data/wpsdata/`). To do so, you can just drag & drop these files into the `Layers` menu (bottom left of the user interface). Or you can also select them thanks to the dedicated panel opened via the `Layer / Add a layer / Add a vectorial layer... /` menu (or use `Ctrl+Maj+V` shortcut)

You should see your input data in the map as below:

Initialize the source point layer

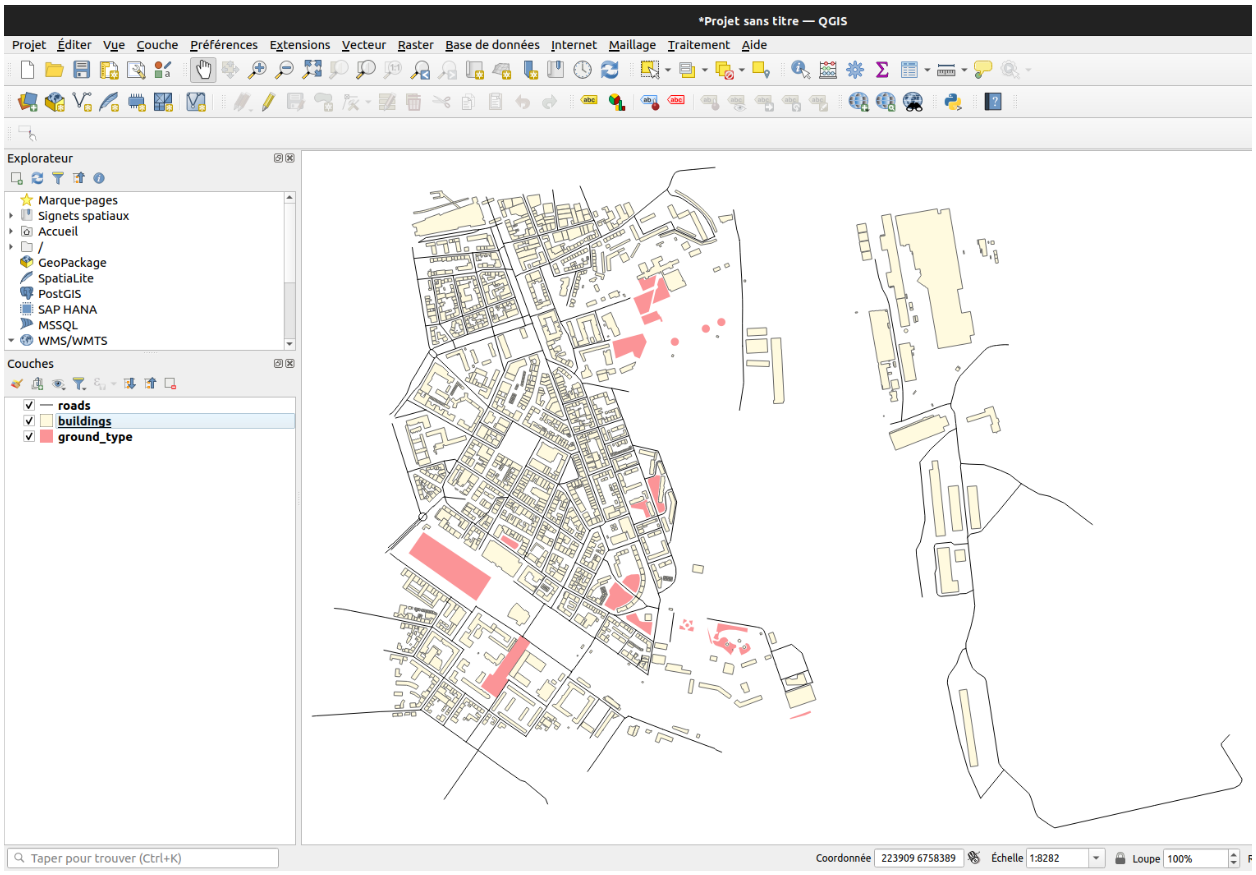
In QGIS, we will create a new empty layer called `Point_Source` in which we will add the source point.

To do so, click on the `Layer / Create Layer / New Temporary Scratch Layer... /` menu. In the opened dialog, fill the detailed information below :

- `File name` : `Point_Source`
- `Geometry type` : choose `Point` in the dropdown list
- `Include Z dimension` : check the box. This way the created point will be defined with X, Y and Z coordinates
- In the projection system dropdown list, choose a metric system. Here we will choose the one used for the buildings, roads and ground_type layers, that are in metropolitan France : Lambert 93 system = `EPSG:2154` (if the system is not present in the dropdown list, use the `Globe` icon on the right to find it)

In the `New field` part, fill the information below:

- `Name` : PK . Unique id (Primary Key)

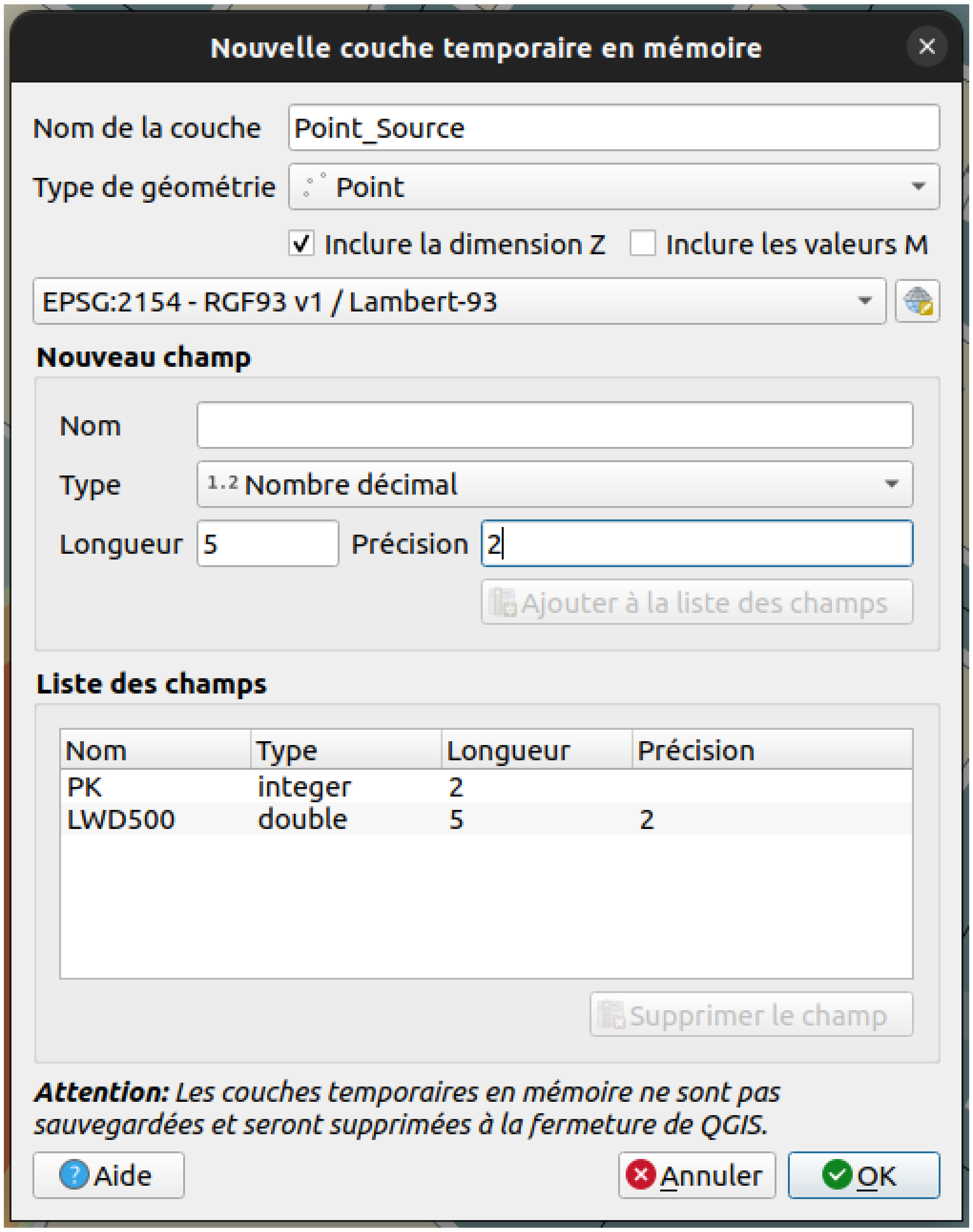

*Projet sans titre — QGIS
Projet Éditer Vue Couche Préférences Extensions Vecteur Raster Base de données Internet Maillage Traitement Aide
Explorateur
Marque-pages
Signets spatiaux
Accueil
/
GeoPackage
SpatiaLite
PostGIS
SAP HANA
MSSQL
WMS/WMTS
Couches
roads
buildings
ground_type
Taper pour trouver (Ctrl+K)
Coordonnée 223909 6758389
Échelle 1:8282
Loupe 100%

- `Type` : Integer
- `Length` : 2

Once done, click on `Add to Fields List`. Then redo this step with the following informations:

- `Name` : LWD500 . Source noise level (LW) during the day (D) at a frequency of 500 Hz
- `Type` : Decimal number
- `Length` : 5
- `Precision` : 2

You should have something like this



Nouvelle couche temporaire en mémoire

Nom de la couche	Point_Source
Type de géométrie	Point

[✓] Inclure la dimension Z [ ] Inclure les valeurs M

EPSG:2154 - RGF93 v1 / Lambert-93

Nouveau champ

Nom	
Type	1.2 Nombre décimal
Longueur	5
Précision	2

Liste des champs

Nom	Type	Longueur	Précision
PK	integer	2	
LWD500	double	5	2

Attention: *Les couches temporaires en mémoire ne sont pas sauvegardées et seront supprimées à la fermeture de QGIS.*

Once done, click on OK button. The new layer Point_Source should appear in your Layers panel.

Add a new source point

Now we have an empty layer. It's time to feed it with a point geometry.

By default, the new temporary layer is already turned into edtion mode. If not, you can activate it thanks to these two options:

- In the Layers panel, select the Point_Source layer and make a right-click. Choose Toggle Editing
- or you can click on the "Yellow pencil" icon in the toolbar

Now we can add a new point, by clicking on the dedicated icon (see illustration below) and then by clicking somewhere in the map.

To have an interesting resulting noise map, choose to place your source point next to buildings.

Click on the map where you want to create the source point. Once clicked, a new dialog appears and you are invited to fill the following attributes:

- PK: 1
- LWD500 : 90

Once done, click on OK. The source point is now visible in the map (the blue point in the illustration below).

Now, we have to save this temporary layer into a flat file. To do so, just make a right-click on the layer name and choose the Make permanent option.

In the new dialog, select GeoJSON file format and then define the path and the name of your resulting .geojson file. Press OK when ready.

Your Point_Source.geojson file is now ready to be imported in NoiseModelling.

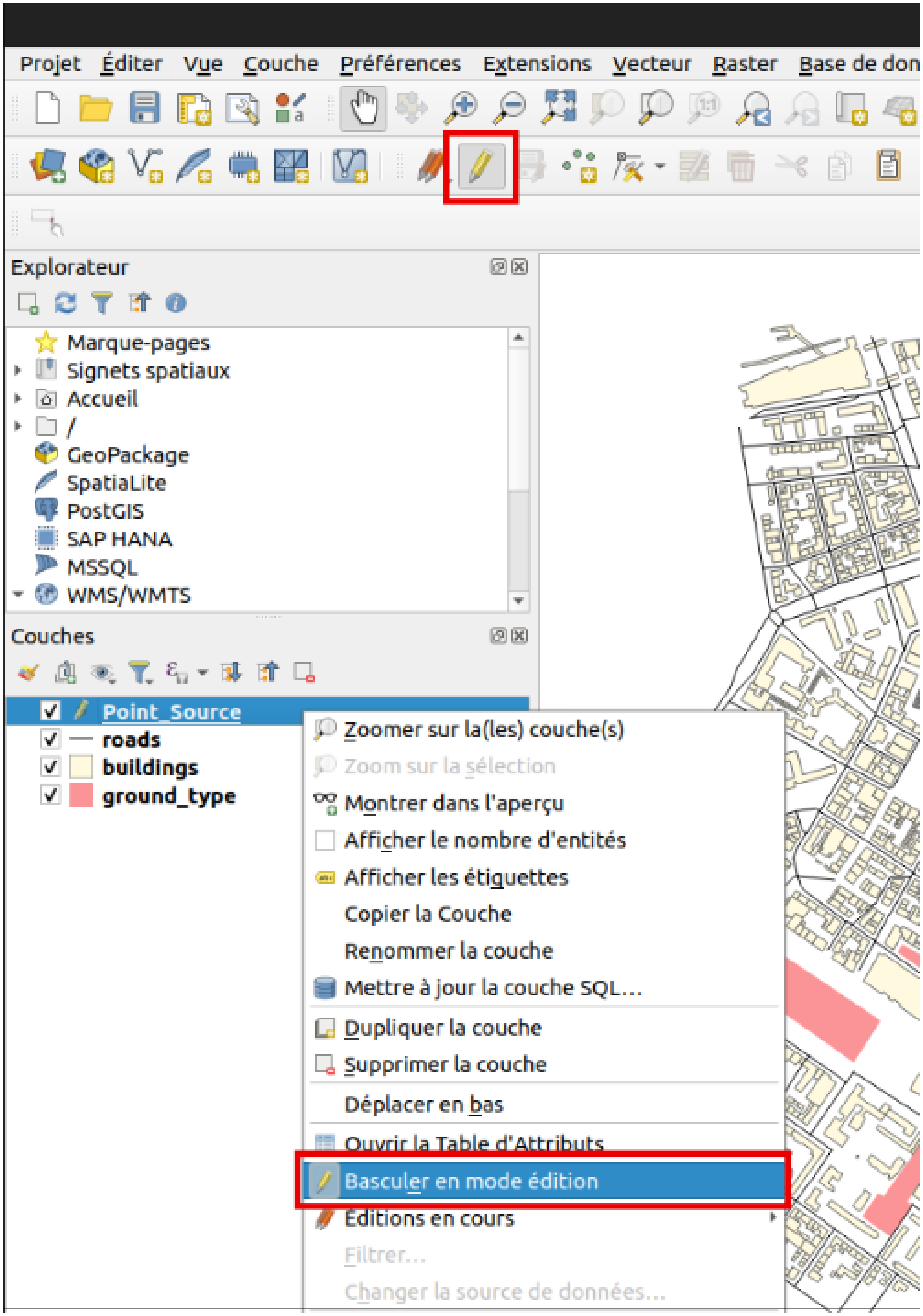
For your information, you can open .geojson files in most of text editor. If you do so with Point_Source.geojson, you will have something like this:

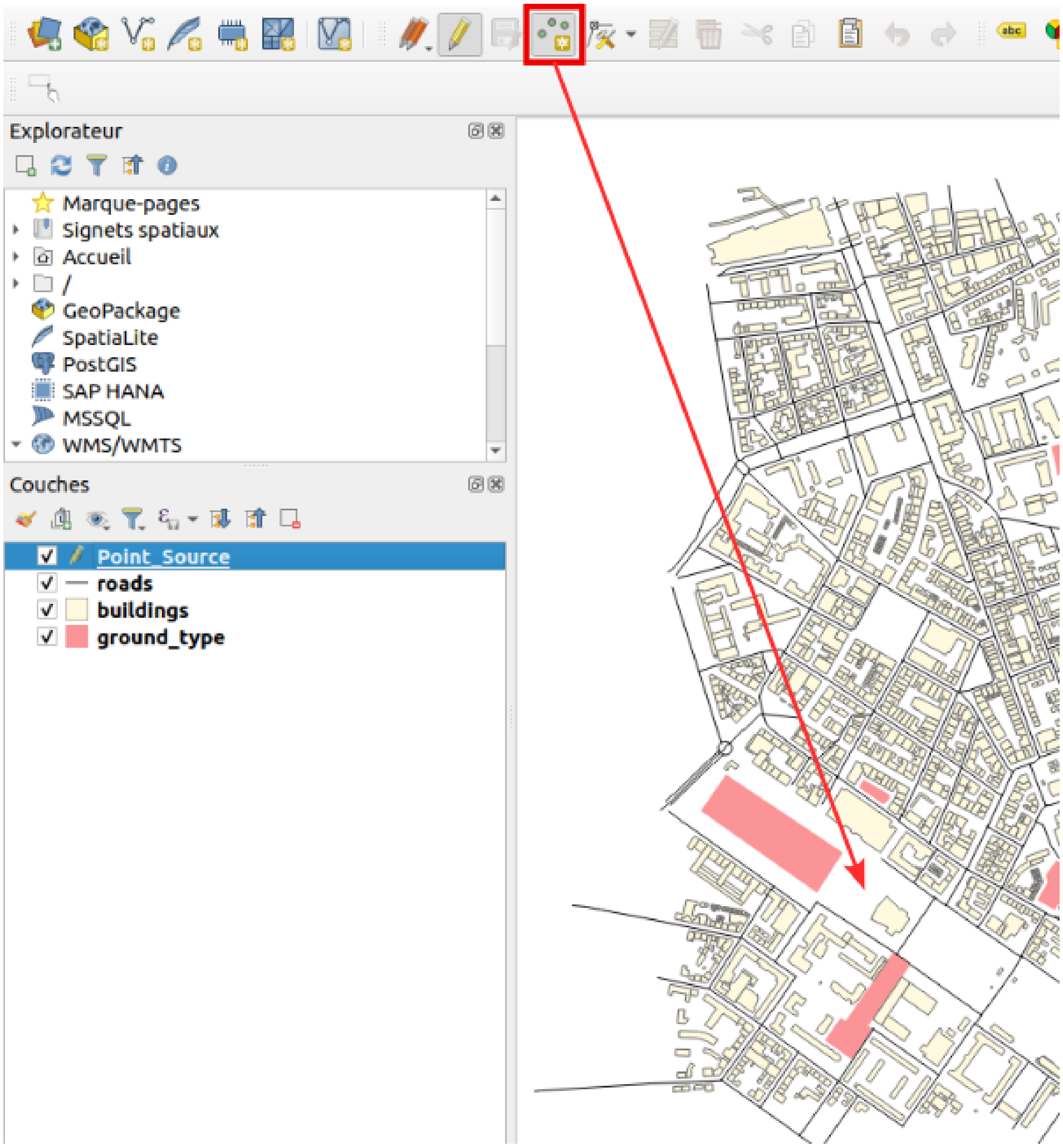
```
{
"type": "FeatureCollection",
"name": "Point_Source",
"crs": {"type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::2154" } },
"features": [{"type": "Feature", "properties": { "PK": 1, "LWD500": 90.0 },
              "geometry": {"type": "Point", "coordinates": [223771.0727, 6757583.2983,
↪ 0.0]}}
             }]
}
```

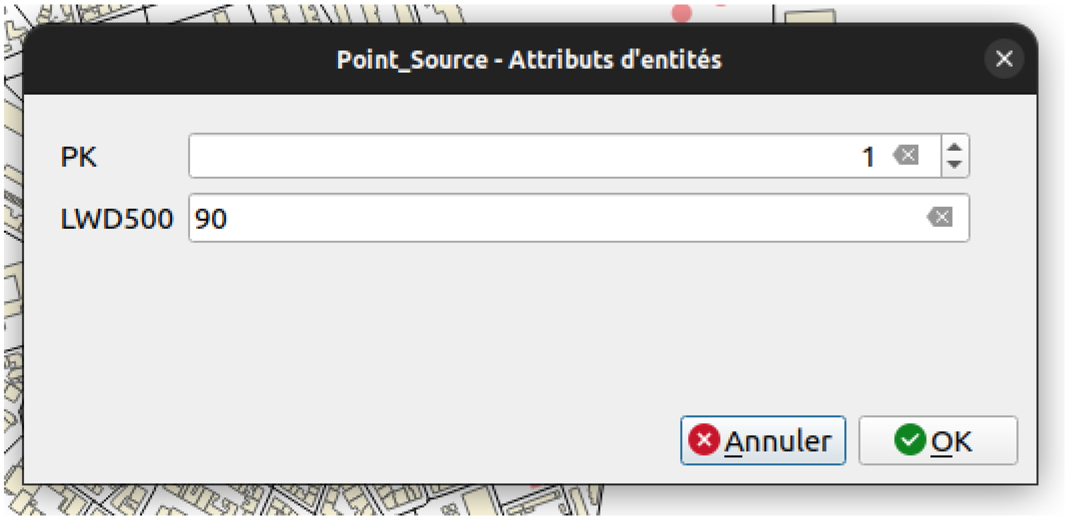
5.16.2 Step 2: Import input data in NoiseModelling

Once NoiseModelling is launched (see Step 2: Start NoiseModelling GUI in *Get Started - GUI* page), load the four BUILDINGS, ROADS and GROUND_TYPE, POINT_SOURCE layers (see Step 4: Load input files for more details).

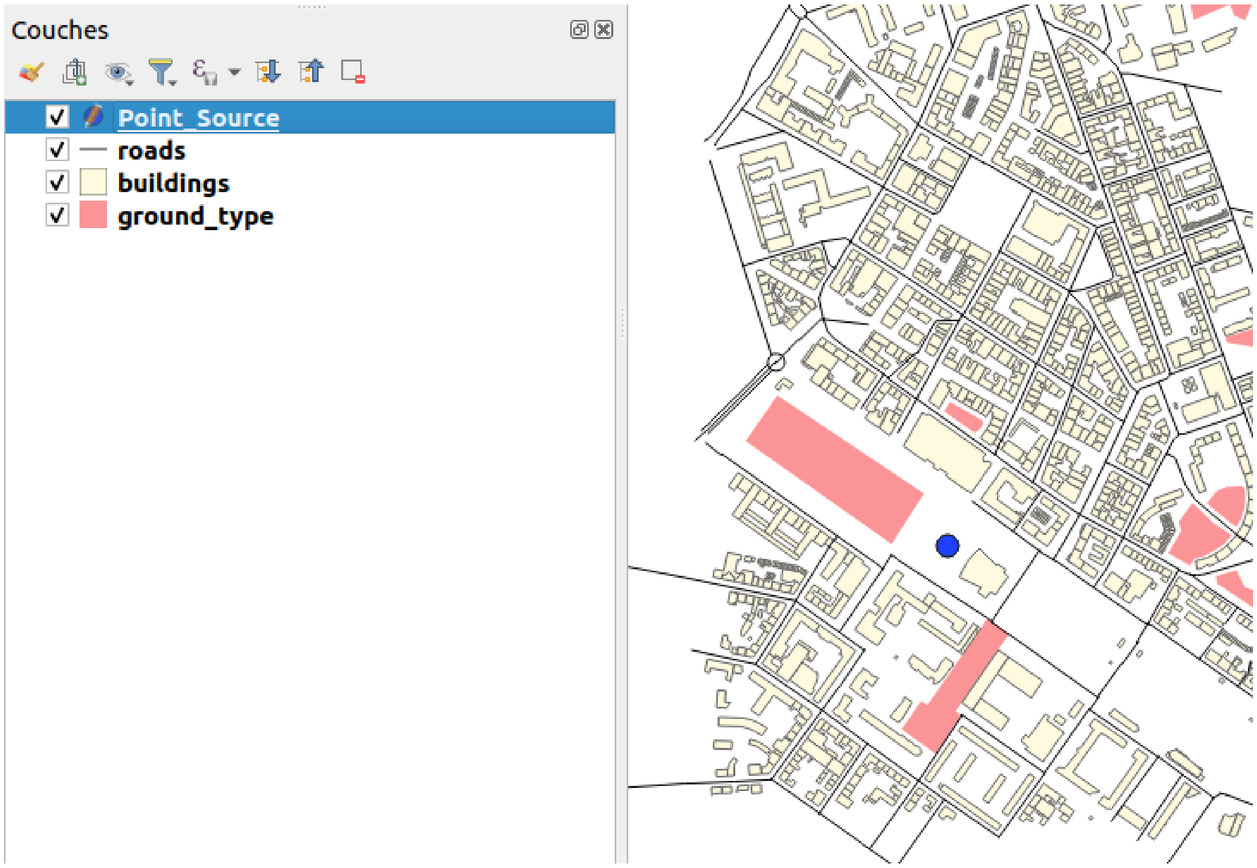
If you use the Database_Manager:Display_Database WPS script, you should see your four tables like below:


Projet
Éditer
Vue
Couche
Préférences
Extensions
Vecteur
Raster
Base de don
Explorateur
Marque-pages
Signets spatiaux
Accueil
/
GeoPackage
SpatiaLite
PostGIS
SAP HANA
MSSQL
WMS/WMTS
Couches
Point_Source
roads
buildings
ground_type
Zoomer sur la(les) couche(s)
Zoom sur la sélection
Montrer dans l'aperçu
Afficher le nombre d'entités
Afficher les étiquettes
Copier la Couche
Renommer la couche
Mettre à jour la couche SQL…
Dupliquer la couche
Supprimer la couche
Déplacer en bas
Ouvrir la Table d'Attributs
Basculer en mode édition
Éditions en cours
Filtrer…
Changer la source de données…

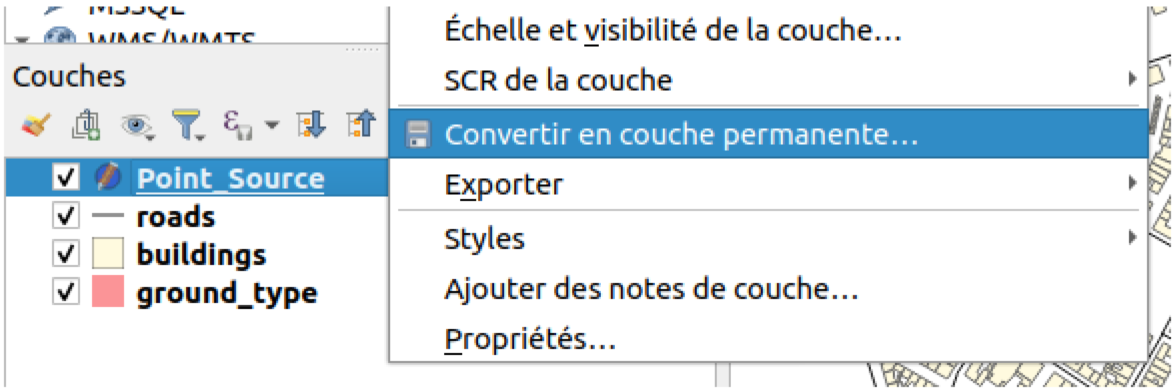

Explorateur
Marque-pages
Signets spatiaux
Accueil
/
GeoPackage
SpatiaLite
PostGIS
SAP HANA
MSSQL
WMS/WMTS
Couches
Point_Source
roads
buildings
ground_type

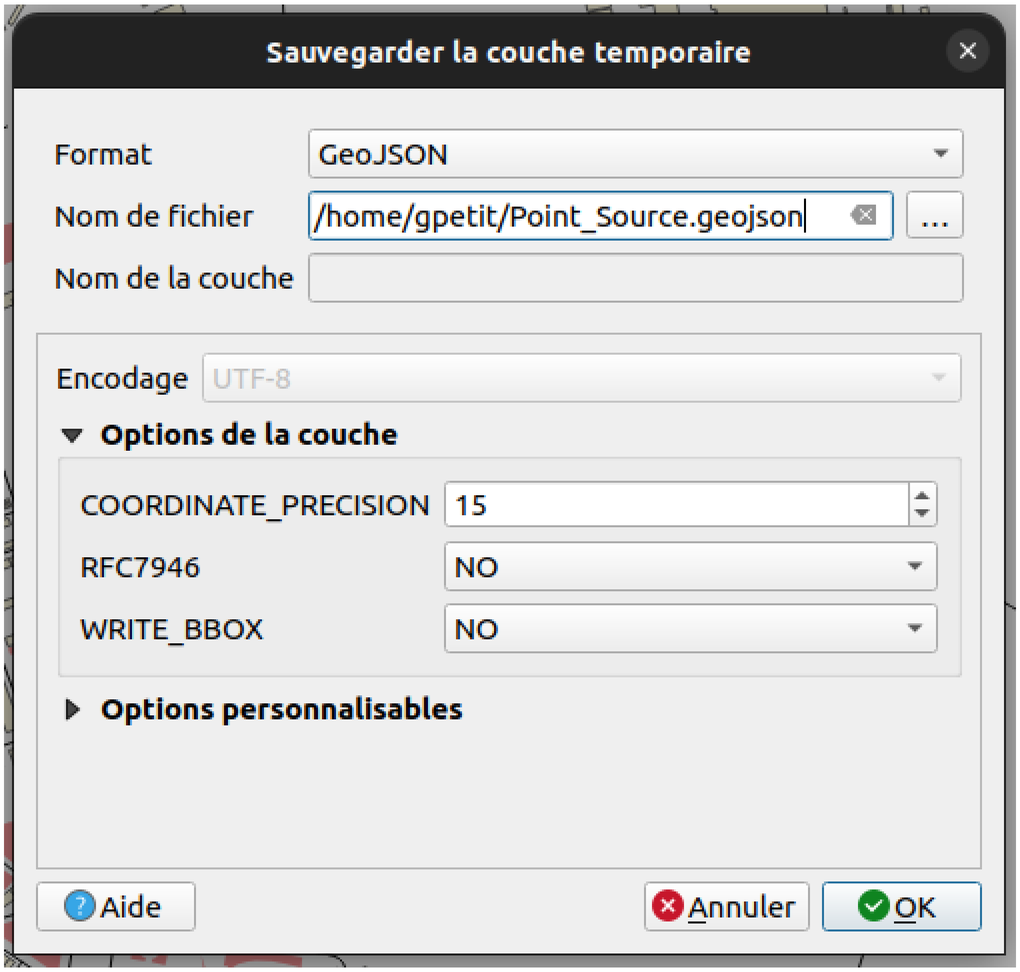

Point_Source - Attributs d'entités

PK	1
LWD500	90


Couches

- ✓ Point_Source
- ✓ roads
- ✓ buildings
- ✓ ground_type




Sauvegarder la couche temporaire

Format: GeoJSON

Nom de fichier: /home/gpetit/Point_Source.geojson

Nom de la couche:

Encodage: UTF-8

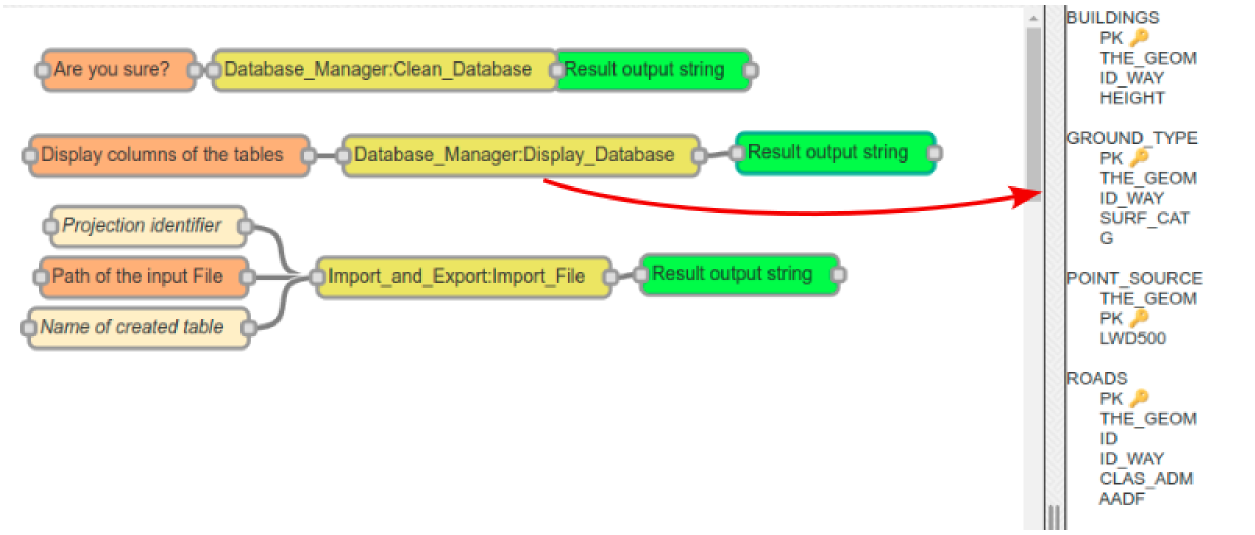
Options de la couche

COORDINATE_PRECISION: 15

RFC7946: NO

WRITE_BBOX: NO

Options personnalisables



BUILDINGS
PK
THE_GEOM
ID_WAY
HEIGHT

GROUND_TYPE
PK
THE_GEOM
ID_WAY
SURF_CAT
G

POINT_SOURCE
THE_GEOM
PK
LWD500

ROADS
PK
THE_GEOM
ID
ID_WAY
CLAS_ADM
AADF

5.16.3 Step 3: Generate the noise map

We are now ready to generate the noise map, based on a unique source point.

Create the receivers grid

Use the `Receivers:Delaunay_Grid` WPS script. Fill the two following mandatory parameters *(in orange)* and click on `Run Process` button:

- `Source table name : POINT_SOURCE`
- `Buildings table name : BUILDINGS`

Once done, you should have two new tables : `RECEIVERS` *(illustrated below with the purple small points)* and `TRIANGLES`

Calculate noise levels

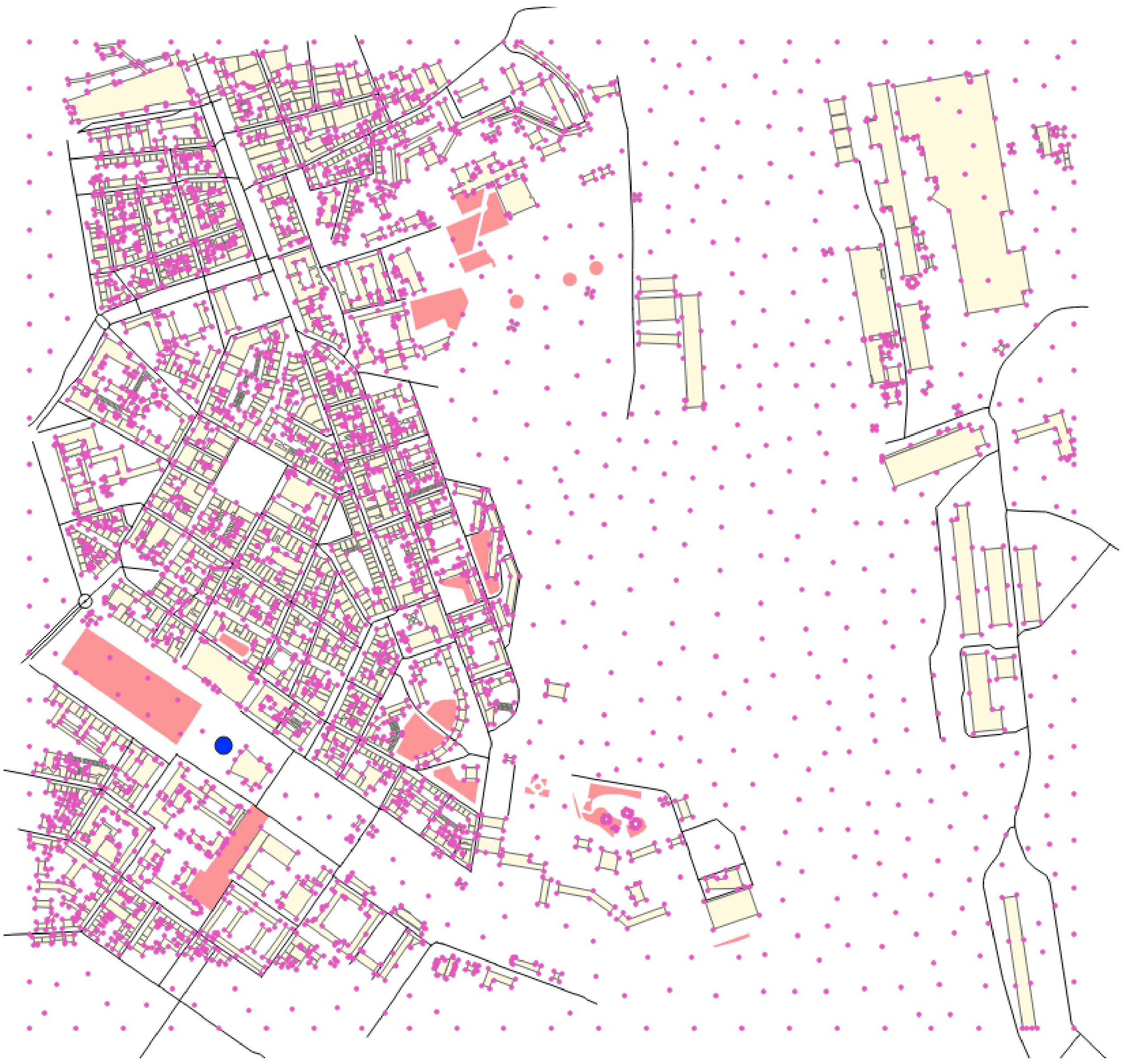
Use the `NoiseModelling:Noise_level_from_source` WPS script. Fill the three following mandatory parameters *(in orange)*:

- `Source table name : POINT_SOURCE`
- `Receivers table name : RECEIVERS`
- `Buildings table name : BUILDINGS`

Warning: For this example, since we only added information for noise level during the day (field `LWD500`), we have to skip the noise level calculation for LDEN, LNIGHT and LEVENING. To do so, check the boxes for `Do not compute LDEN_GEOM`, `Do not compute LEVENING_GEOM` and `Do not compute LNIGHT_GEOM` options.

Once ready, click on `Run Process` button.

You should then have this message: `Calculation Done ! LDAY_GEOM table(s) have been created.`



Generate noise level isosurfaces

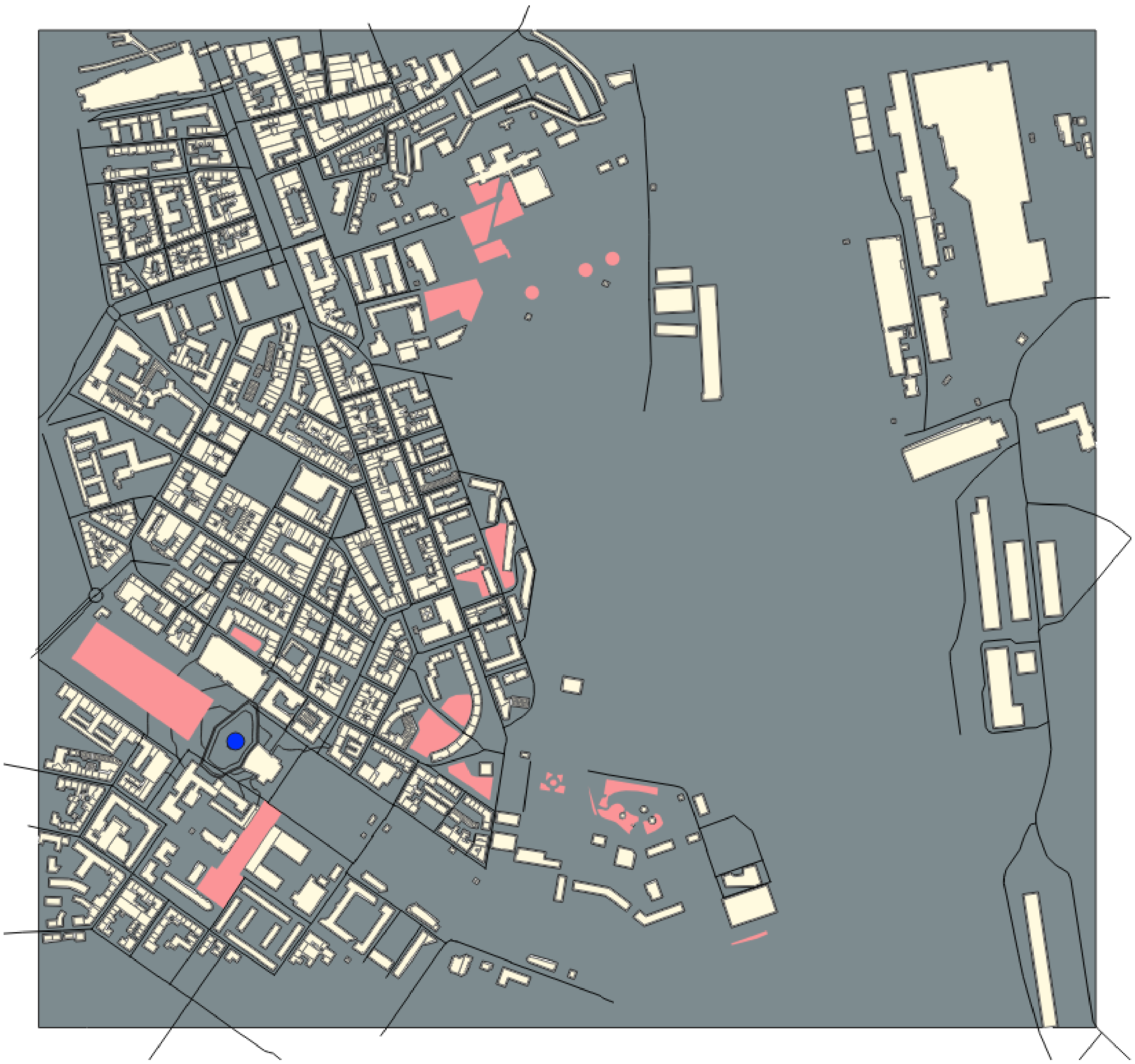
Use the `Acoustic_Tools:Create_Isosurface` WPS script. Fill the following mandatory parameter *(in orange)* and click on `Run Process` button:

- `Sound levels table`: `LDAY_GEOM`

You should have this message: `Table CONTOURING_NOISE_MAP created`

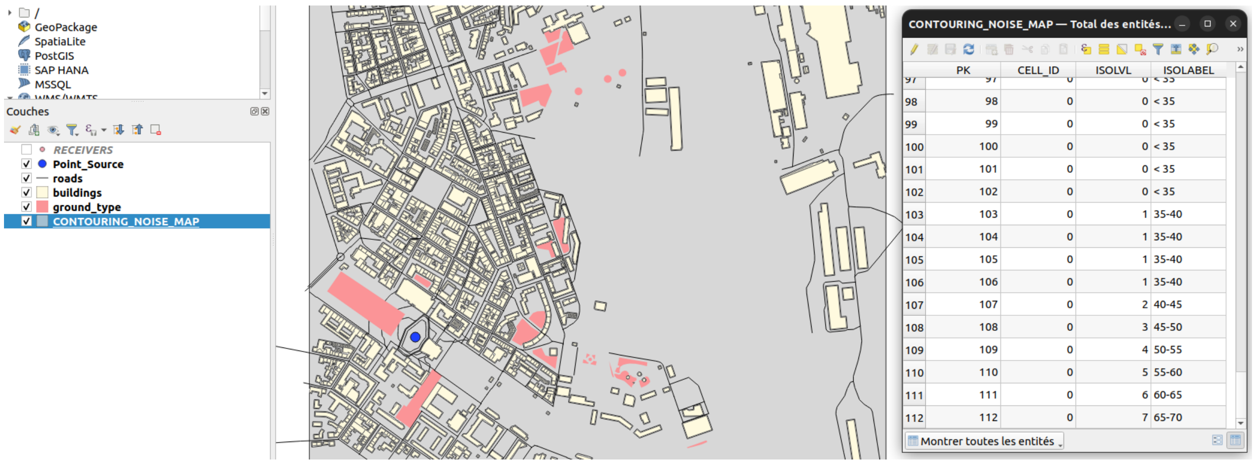
Now, you can export this table into a .shapefile, using the `Import_and_Export:Export_Table` WPS script.

You can then visualize this file into QGIS *(just load the file as seen before)*. The resulting table *(in grey)* is illustred below



Apply a color palette adapted to acoustics

In QGIS, since the isosurface table is not easy to read *(everything is grey in our example)*, we will change the color palette to have colors depending on the noise levels. This information is present in the field `ISOLVL` in the attributes table. To open it, just select the layer `CONTOURING_NOISE_MAP` and press `F6`.



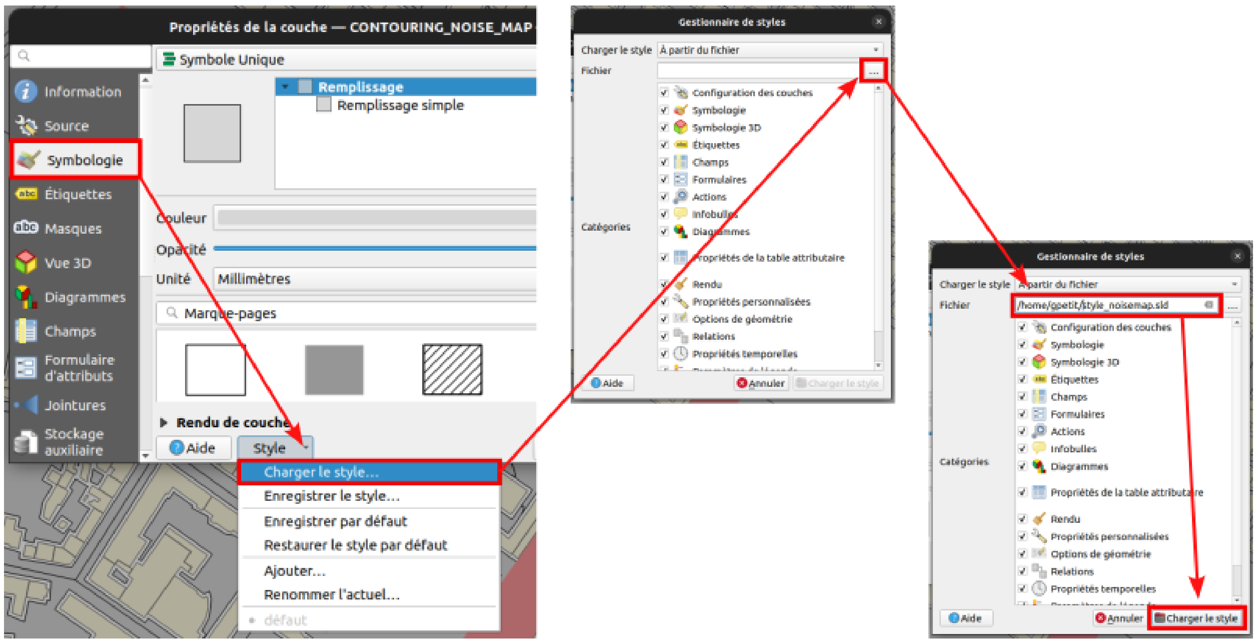
	PK	CELL_ID	ISOLVL	ISOLABEL
97	97	0	0	< 35
98	98	0	0	< 35
99	99	0	0	< 35
100	100	0	0	< 35
101	101	0	0	< 35
102	102	0	0	< 35
103	103	0	1	35-40
104	104	0	1	35-40
105	105	0	1	35-40
106	106	0	1	35-40
107	107	0	2	40-45
108	108	0	3	45-50
109	109	0	4	50-55
110	110	0	5	55-60
111	111	0	6	60-65
112	112	0	7	65-70

To adapt the colors, we will apply a cartographic style. This style:

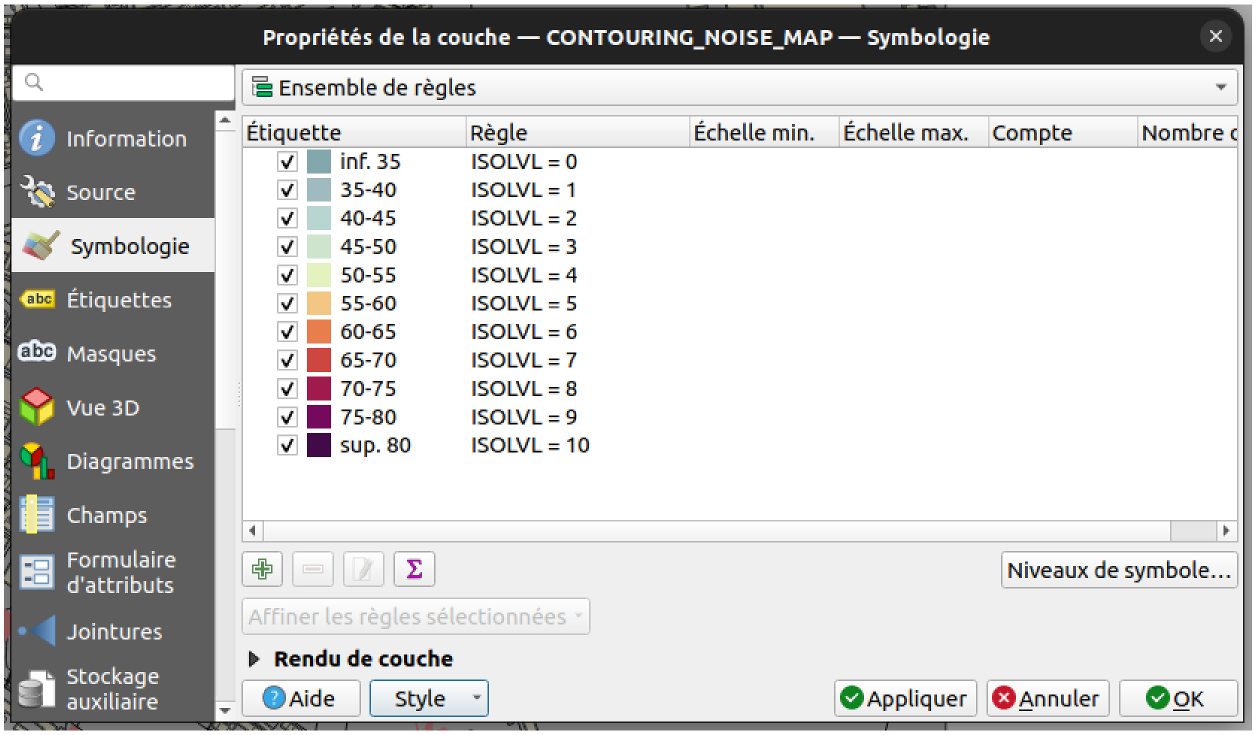
- has been proposed by B. Tomio (Weninger) in *"A Color Scheme for the Presentation of Sound Immission in Maps : Requirements and Principles for Design"* (see publication and website)
- is provided *(by NoiseModelling team)* as a `.sld` *(Style Layer Descriptor)* file and can be downloaded here

Note: If you want to know more about noise map styles, you should read the "*Noise Map Color Scheme*" page.

Once downloaded, make a double click on the layer `CONTOURING_NOISE_MAP`. It will opens the property panel. Here, click on the `Symbology` tab. In the `Style` menu *(at the bottom)*, choose `Load style`. Then in the opened dialog, click on the `...` icon to search the `style_beate_tomio.sld` file. Once selected, click on `Load style`.



The style with its different colors is now displayed.

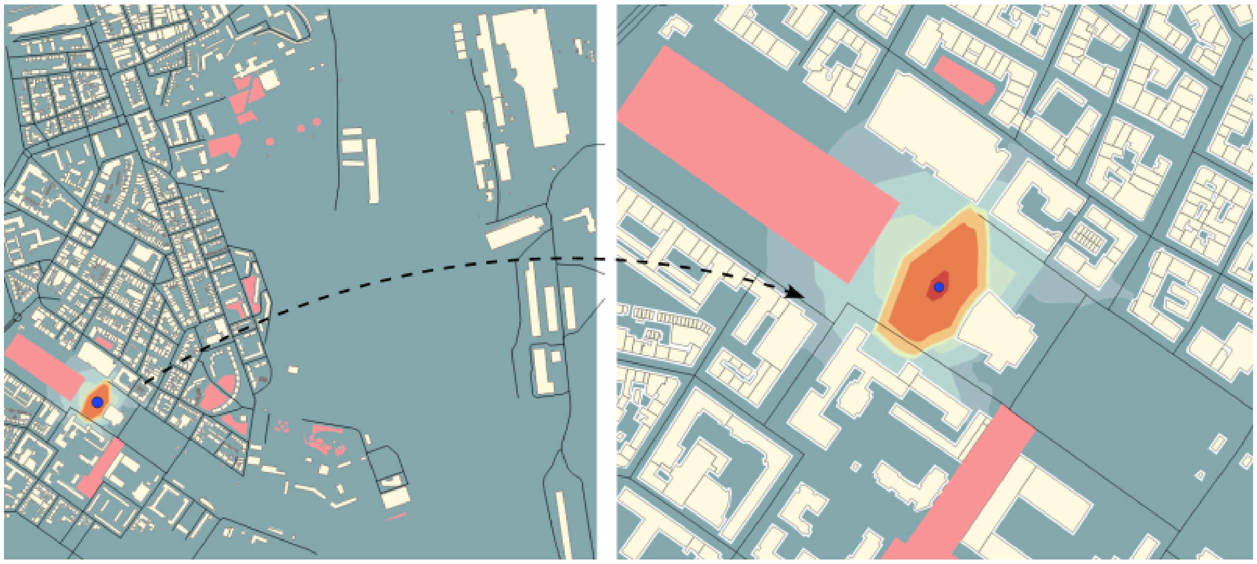


Propriétés de la couche — CONTOURING_NOISE_MAP — Symbologie

Ensemble de règles

Étiquette	Règle	Échelle min.	Échelle max.	Compte	Nombre c
✓ inf. 35	ISOLVL = 0				
✓ 35-40	ISOLVL = 1				
✓ 40-45	ISOLVL = 2				
✓ 45-50	ISOLVL = 3				
✓ 50-55	ISOLVL = 4				
✓ 55-60	ISOLVL = 5				
✓ 60-65	ISOLVL = 6				
✓ 65-70	ISOLVL = 7				
✓ 70-75	ISOLVL = 8				
✓ 75-80	ISOLVL = 9				
✓ sup. 80	ISOLVL = 10				

Press `OK` to apply and close the dialog. Your noise map is now well colorized and you can navigate into it to see the influence of buildings on noise levels.



5.16.4 Step 4: Change the default parameters

To produce this noise map, we used, in most of WPS scripts, default parameters (*e.g* the height of the source, the number of reflections, the air temperature, . . .). You are prompted to redo some of the previous steps by changing some of the settings. You will then be able to visually see what impact they have on the final noise map.

Note: To change optionnal parameters *(the yellow boxes)* just select them and fill the needed informations in the right-side menu.

5.16.5 Step 5 (bonus): Change the directivity

In this bonus step, we will manage with the directivity. To do so, we will apply the following method:

1. Get directivity
2. Update the `Source_Point` table
3. Import needed data into NoiseModelling
4. Produce the noise map, taking into acount directivity parameters

Directivity

The directivity table aims at modeling a realistic directional noise source. To do so, we associate to each "Theta-Phi" pair an attenuation in dB.

- `DIR_ID` : identifier of the directivity sphere
- `THETA` : vertical angle in degrees, 0 (front), -90 (bottom), 90 (top), from -90 to 90
- `PHI`: horizontal angle in degrees, 0 (front) / 90 (right), from 0 to 360
- `LW500` : attenuation levels in dB for 500 Hz

Each of the sound sources has its own directivity. For the exercise we will use the directivity of a train, which is provided in the file Directivity.csv and which you are invited to download.

Table 2: Extract from the directivity table (Directivity.csv)

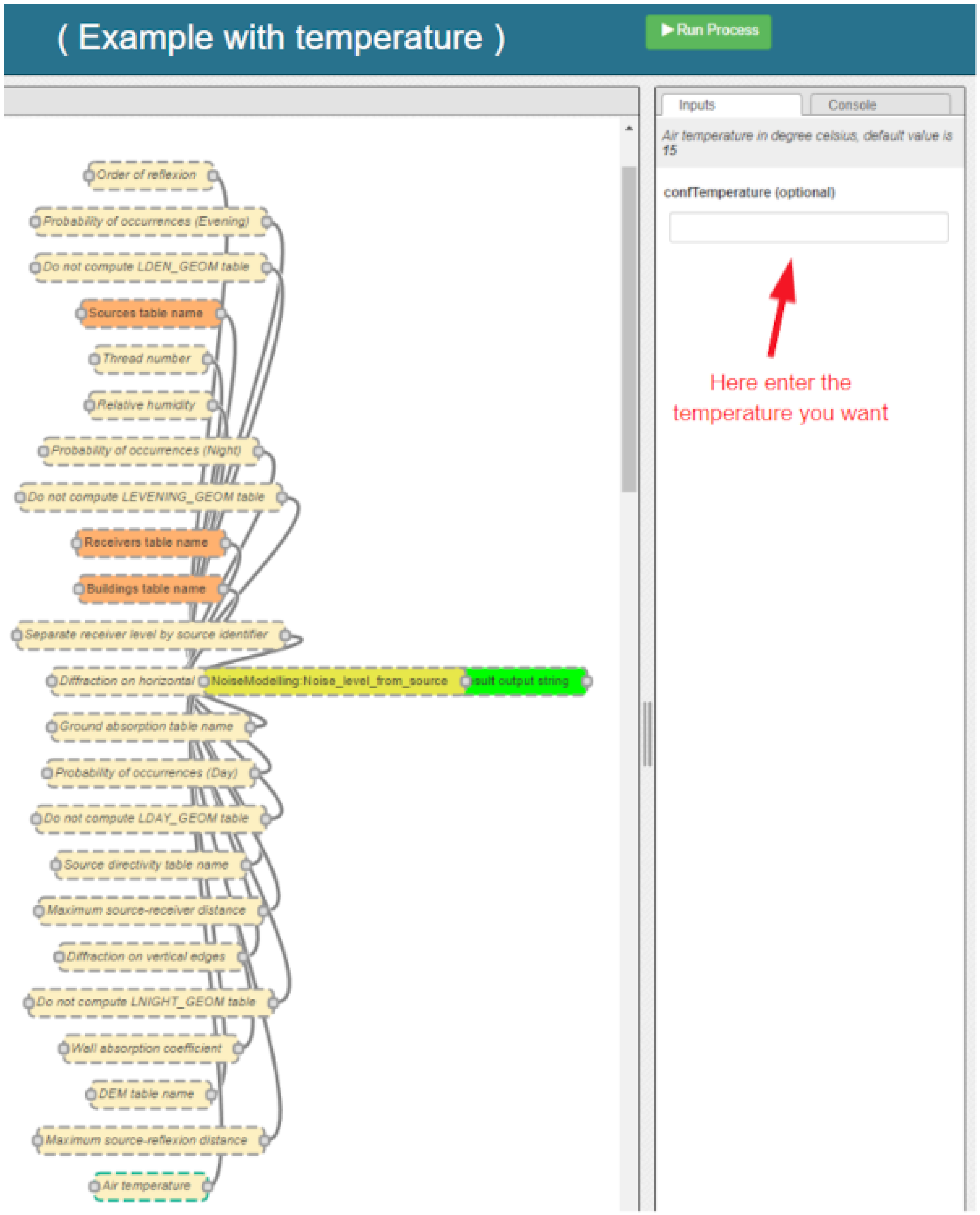
DIR_ID	THETA	PHI	LW500
1	-90	0	-20
1	-85	0	-20
1	-80	0	-20
1	-75	0	-20
1	-70	0	-20
1	-65	0	-20
…	…	…	…

Below is an illustration generated from train directivity formula.

Update source point table

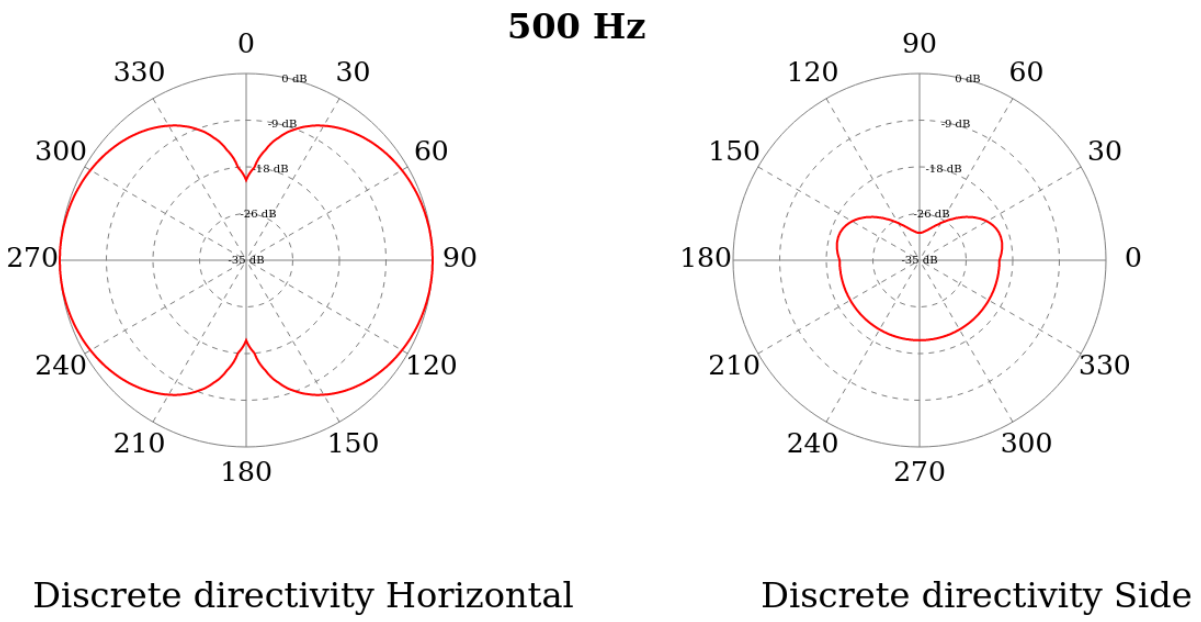
To play with directivity, we need to add 4 fields in the source point table:

- **Yaw**
  - `Name` : YAW
  - `Description` : Source horizontal orientation in degrees. For points 0° North, 90° East. For lines 0° line direction, 90° right of the line direction.
  - `Type` : Decimal number


(Example with temperature)
▶ Run Process
Order of reflexion
Probability of occurrences (Evening)
Do not compute LDEN_GEOM table
Sources table name
Thread number
Relative humidity
Probability of occurrences (Night)
Do not compute LEVENING_GEOM table
Receivers table name
Buildings table name
Separate receiver level by source identifier
Diffraction on horizontal
NoiseModelling:Noise_level_from_source
sult output string
Ground absorption table name
Probability of occurrences (Day)
Do not compute LDAY_GEOM table
Source directivity table name
Maximum source-receiver distance
Diffraction on vertical edges
Do not compute LNIGHT_GEOM table
Wall absorption coefficient
DEM table name
Maximum source-reflexion distance
Air temperature
Inputs
Console
Here enter the temperature you want

Air temperature in degree celsius, default value is 15

confTemperature (optional)

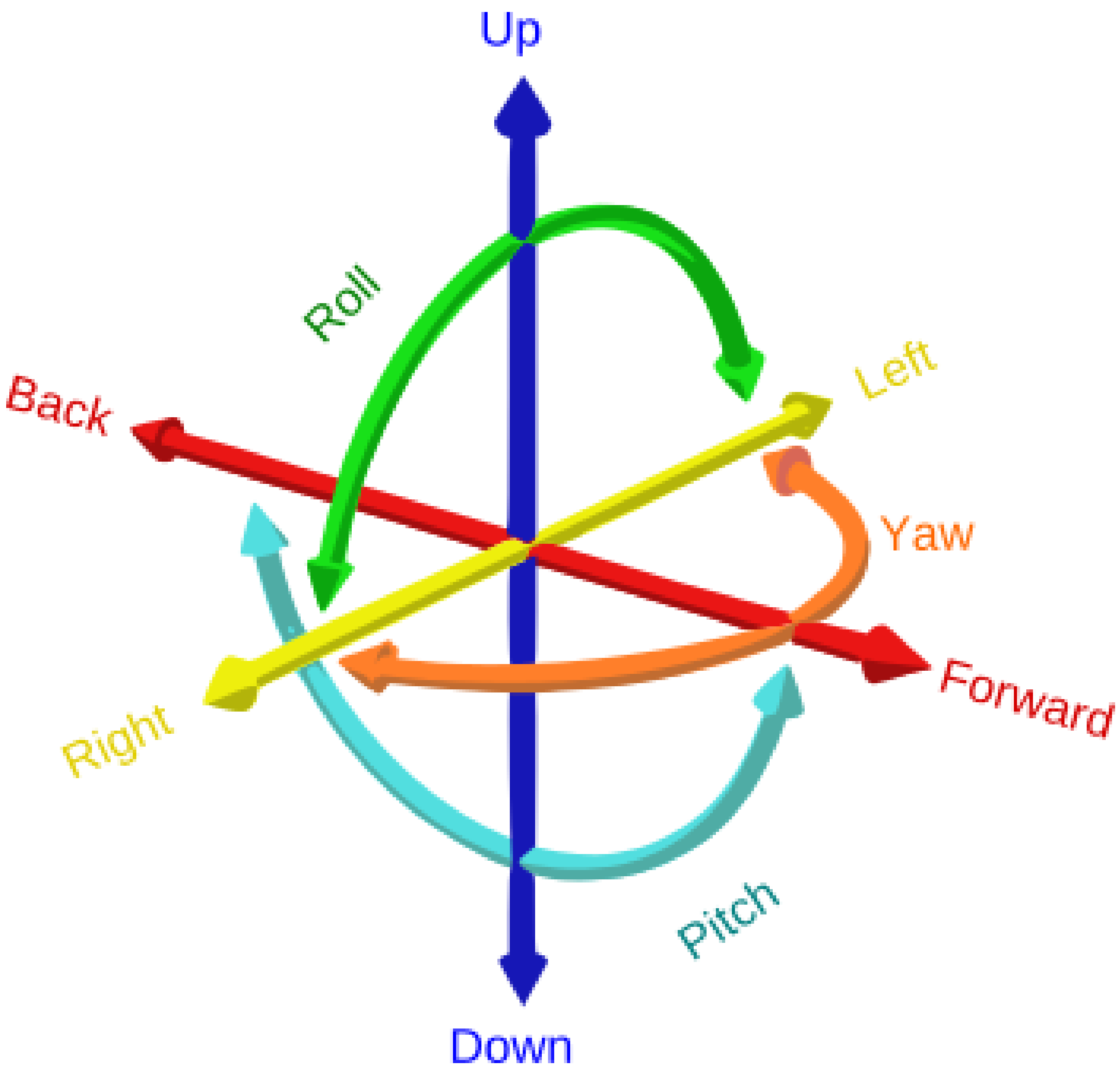


- `Length` : 4

- **Pitch**
  - `Name` : PITCH
  - `Description` : Source vertical orientation in degrees. 0° front, 90° top, -90° bottom. (FLOAT).
  - `Type` : Decimal number
  - `Length` : 4
- **Roll**
  - `Name` : ROLL
  - `Description` : Source roll in degrees
  - `Type` : Decimal number
  - `Length` : 4
- **Direction identifier**
  - `Name` : DIR_ID
  - `Description` : Identifier of the directivity sphere from tableSourceDirectivity parameter or train directivity if not provided -> OMNIDIRECTIONAL(0), ROLLING(1), TRACTIONA(2), TRACTIONB(3), AERODYNAMICA(4), AERODYNAMICB(5), BRIDGE(6)
  - `Type` : Integer
  - `Length` : 2

Note: Source image: GregorDS, CC BY-SA 4.0, via Wikimedia Commons

In our example, we will update the `Point_Source.geojson` file to add these columns and to fill them with new information. To do so, just edit the file into a text editor and replace the following lines. Save it once done.


Up
Roll
Back
Left
Yaw
Right
Forward
Pitch
Down

```
{ "PK": 1, "LWD500": 100.0}
```

by

```
{ "PK": 1, "LWD500": 100.0, "YAW": 45, "PITCH": 0, "ROLL": 0, "DIR_ID" : 1 }
```

Here we can see that the Yaw is setted to 45°. Pitch and Roll are equal to 0, and the directivity is defined as `1` and will refer to the directivy table (see below).

So your final .geojson file should look like this

```
{
"type": "FeatureCollection",
"name": "Point_Source",
"crs": {"type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::2154" } },
"features": [{"type": "Feature", "properties": { "PK": 1, "LWD500": 100.0, "YAW": 45,
↪"PITCH": 0, "ROLL": 0, "DIR_ID" : 1 },
               "geometry": {"type": "Point", "coordinates": [223771.0727, 6757583.2983,
↪ 0.0]}
              }]
}
```

Import data

Now, in NoiseModelling we have to:

- Import the `Directivy.csv` file
- Reimport the `Point_Source.geojson` file in order to take into account the changes
- Import the `dem.geojson` file, which is placed here `./NoiseModelling_4.0.0/data_dir/data/wpsdata/dem.geojson`. By taking into account the ground elevation, this file will help us to get better results.

To do so, just use the `Import_and_Export:Import_Table` WPS script.

Generate the Delaunay triangulation

Use the `Receivers:Delaunay_Grid` WPS script. Fill the following parameters and click on `Run Process` button:

- `Sources table name`: `POINT_SOURCE`
- `Maximum Area`: `60`
- `Buildings table name`: `BUILDINGS`
- `Height`: `1.6`

Compute noise level from source

Use the `NoiseModelling:Noise_level_from_source` WPS script. Fill the following parameters and click on `Run Process` button:

- `Sources table name`: `POINT_SOURCE`
- `Buildings table name`: `BUILDINGS`

- `Receivers table name`: `RECEIVERS`
- `Ground absorption table name`: `GROUND_TYPE`
- `Source directivity table name`: `DIRECTIVITY`
- `Maximum source-receiver distance`: `800`
- `Do not compute LDEN_GEOM table`: `true`
- `Do not compute LNIGHT_GEOM table`: `true`
- `Do not compute LEVENING_GEOM table`: `true`
- `DEM table name`: `DEM`

Create isosurface

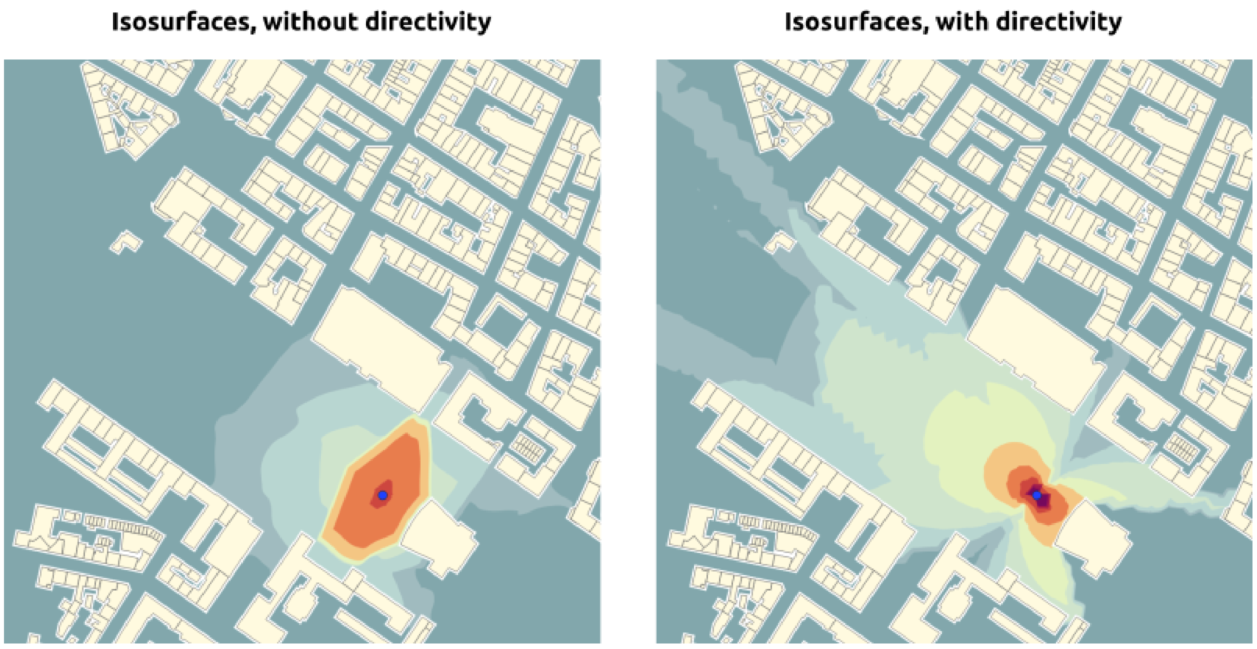
Use the `Acoustic_Tools:Create_Isosurface` WPS script. Fill the following parameters and click on `Run Process` button:

- `Sound levels table`: `LDAY_GEOM`
- `Polygon smoothing coefficient`: 0.4

Export and visualize resulting tables

Use the `Import_and_Export:Export_Table` WPS script to export the `CONTOURING_NOISE_MAP` table into a shapefile called `CONTOURING_NOISE_MAP_DIRECTIVITY`.

Then, load `CONTOURING_NOISE_MAP_DIRECTIVITY.shp` into QGIS. Apply the `noisemap_style.sld` style, and compare with `CONTOURING_NOISE_MAP.shp` produced in Step 3.


Isosurfaces, without directivity
Isosurfaces, with directivity

5.17 MATSim - GUI

5.17.1 Introduction

MATSim (https://matsim.org/) is an open-source framework for implementing large-scale agent-based transport simulations. In this tutorial we will learn how to import the output of a successful MATSim simulation into NoiseModelling. The idea is to use the traffic data from MATSim for NoiseModelling road noise emission. Then we will leverage the fact that MATSim is a multi-agent simulator. We will import MATSim agent's positions to calculate their noise exposition throughout the simulated day.

For this tutorial, we'll look into a simulation of the island in the center of Nantes, the 6th most populated city in France.

5.17.2 Prerequisites

- You need to have a working installation of the latest NoiseModelling version
- A basic knowledge of what the MATSim traffic simulator does and how it works is preferable
- (optional) A working installation of DBeaver (https://dbeaver.io/) can be useful to visualize the NoiseModelling database tables
- (optional) A working installation of Simunto Via (https://www.simunto.com/via/) can be useful for visualizing the MATSim scenario
- (optional) A working installation of QGis (https://www.qgis.org/) can be useful to visualize resulting GIS data

5.17.3 The data

You can download and unzip the data in any folder from here : https://github.com/Universite-Gustave-Eiffel/NoiseModelling/releases/download/v3.3.1/scenario_matsim.zip

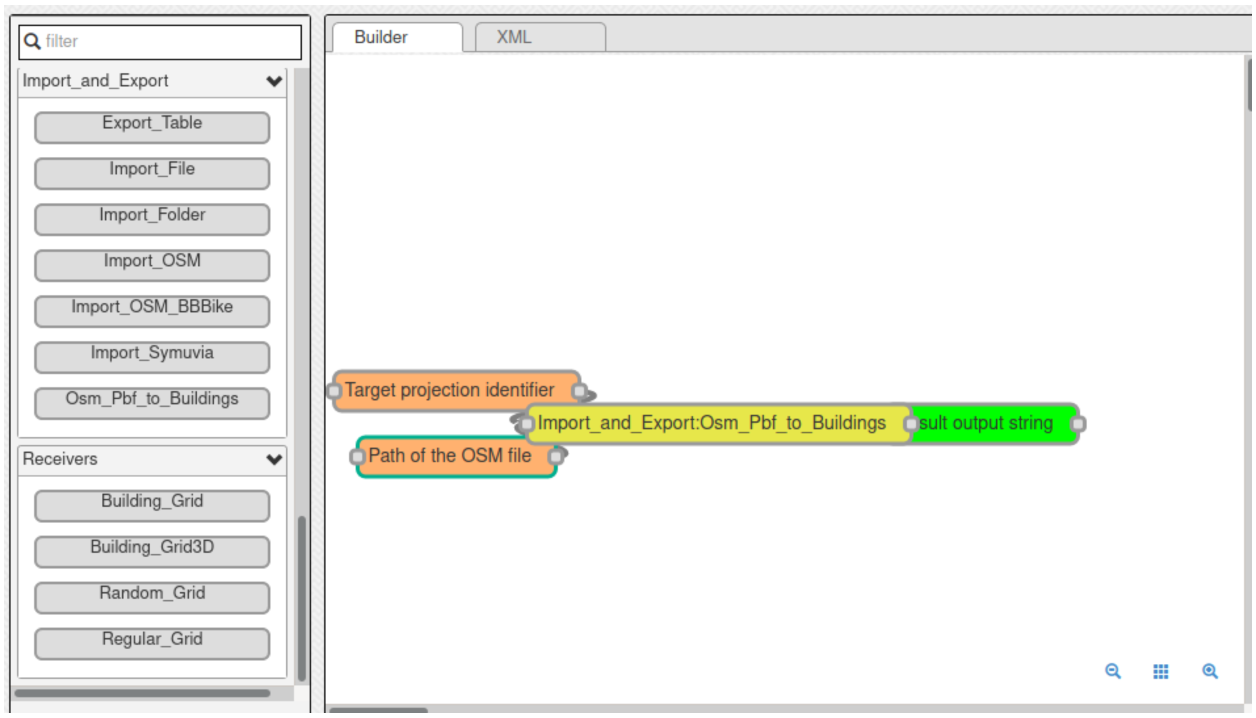
The data folder should contain the following files :

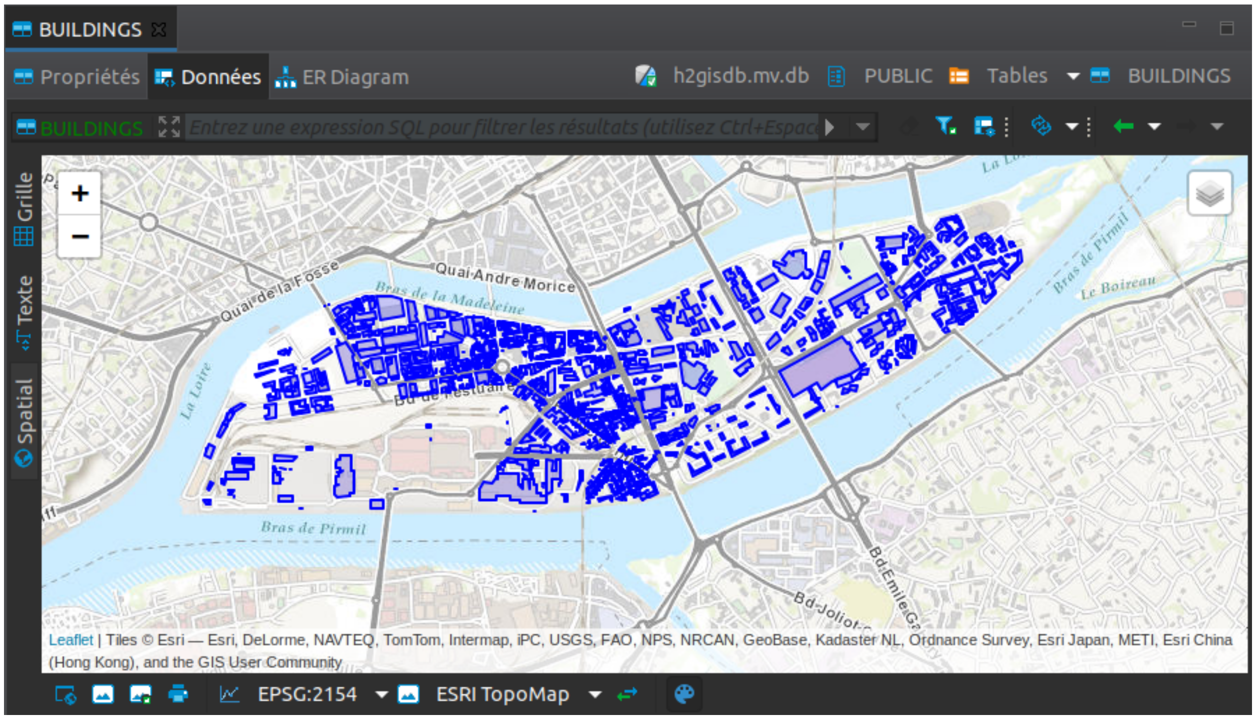
- `nantes_ile.osm.pbf` : the Openstreetmap data of the area. We'll use it to import buildings into NoiseModelling.
- `network.csv` : A file containing the 'true' geometries of the road segments (called "links" in MATSim)
- `output_events.xml.gz` : A file containing the list of MATSim events from the simulation.
- `output_facilities.xml.gz` : A file containing the list of facilities, the agent's activity locations.
- `output_network.xml.gz` : A file containing the MATSim road network, a list of nodes and links.
- `output_plans.xml.gz` : A file containing the list of agents and their final schedule, or plan.

5.17.4 Step 1 : Import Buildings

The first thing we're going to do is to import buildings. We use the `Import_OSM` WPS block to do that. Simply put the `nantes_ile.osm.pbf` path in the 'pathFile' input and set the 'SRID' input to 2154 *(which is the EPSG code for the french regulatory system)*.

You should end up with a `BUILDINGS` table containing the island buildings.


filter
Import_and_Export
Export_Table
Import_File
Import_Folder
Import_OSM
Import_OSM_BBBike
Import_Symuvia
Osm_Pbf_to_Buildings
Receivers
Building_Grid
Building_Grid3D
Random_Grid
Regular_Grid
Builder
XML
Target projection identifier
Import_and_Export:Osm_Pbf_to_Buildings
sult output string
Path of the OSM file


BUILDINGS
Propriétés
Données
ER Diagram
h2gisdb.mv.db
PUBLIC
Tables
BUILDINGS
Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+Espac
Grille
Texte
Spatial
Quai de la Fosse
Quai André Morice
Bras de la Madeleine
La Loire
Bras de Pirmil
Le Boireau
Bd Joliot
Bd Emile
Leaflet | Tiles © Esri — Esri, DeLorme, NAVTEQ, TomTom, Intermap, iPC, USGS, FAO, NPS, NRCAN, GeoBase, Kadaster NL, Ordnance Survey, Esri Japan, METI, Esri China (Hong Kong), and the GIS User Community
EPSG:2154
ESRI TopoMap

5.17.5 Step 2 : Import MATSim Traffic Data

Now we can import the traffic data from the MATSim simulation. To do that, we use the `Traffic_From_Events` WPS block.

The mandatory inputs are :

- `folder` : the path of the MATSim folder, here it is where you put the content of the `scenario_matsim.zip` file
- `timeSlice` : wich represents the time period you want to aggregate the traffic data over. Here let's use the "quarter" option.

One optional but very important input is the `Network CSV` file path. The idea is that when the MATSim scenario was run, the link geometries were simplified to save computation time. This simplification of roads geometry is a bad thing for NoiseModelling since we take buidlings into account (simplified links can pass through buildings) and since source-receiver distance has a big impact on noise levels. That's why the `network.csv` file is given with the other data files. It contains the "real" geometry of links before MATSim simplification process (FYI, This is obtained by setting the 'outputDetailedLinkGeometryFile' option to a file name in the `pt2matsim` config file).

An other important parameter is the `populationFactor`. This corresponds to the downscaling factor that was used to generate the list of agents. Typically, this list of agents is generated based on the available census and survey data for an administrative area. Here, for our use case, the MAtsim scenario and it's agents were generated by using only 1% of the area total population (that is a population factor of 0.01).

You can explore the other options by reading their descriptions. Here we are going to set them as follows:

- Network CSV file: `/path/to/your/scenario_matsim/network.csv`
- Export additionnal traffic data ? : `false`
- Calculate All vehicles noise source ?: `true`
- Path of the matsim output folder: `/path/to/your/scenario_matsim`
- populationFactor: `0.01`
- Time Quantification: `quarter`
- outTableName: "" (not set, use default)
- Skip unused links ?: `true`
- Projection identifier: `2154`
- ignoreAgents: "" (not set)

You should end up with a `MATSIM_ROADS` table containing the links ids and their geometry and a `MATSIM_ROADS_STATS` table containing the noise power level of each link per 15 min time slice.

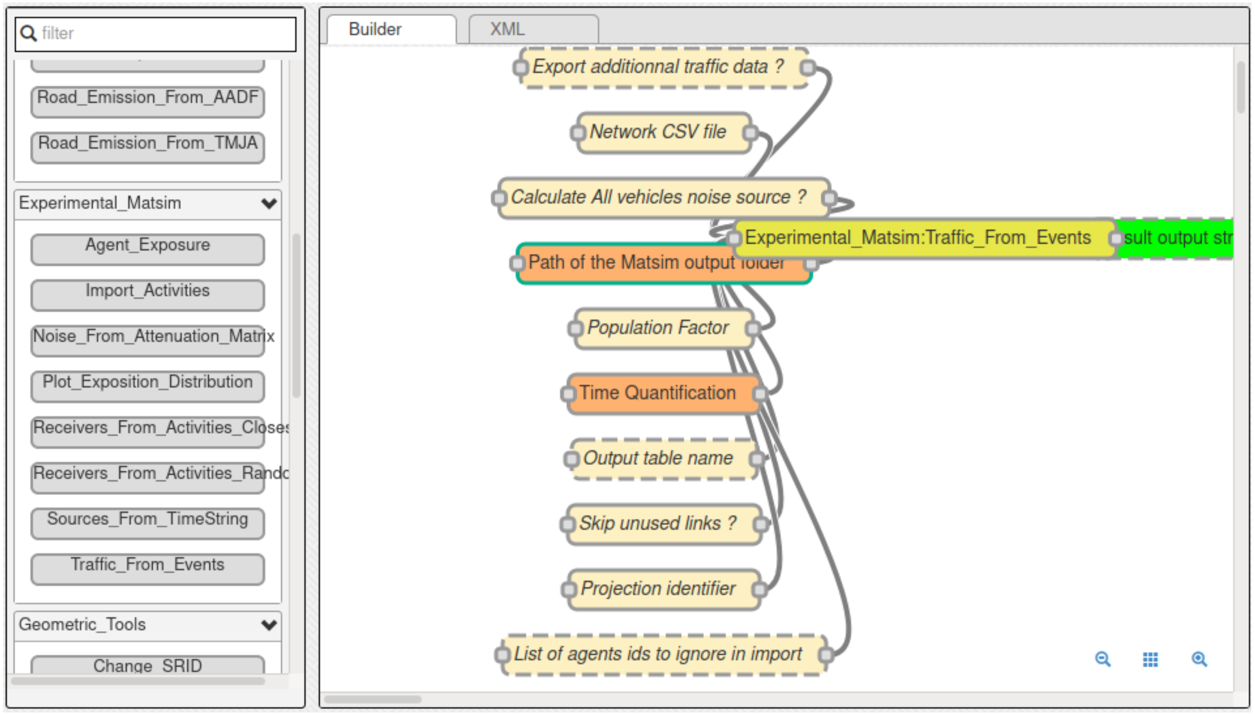
5.17.6 Step 3 : Import MATSim Activities

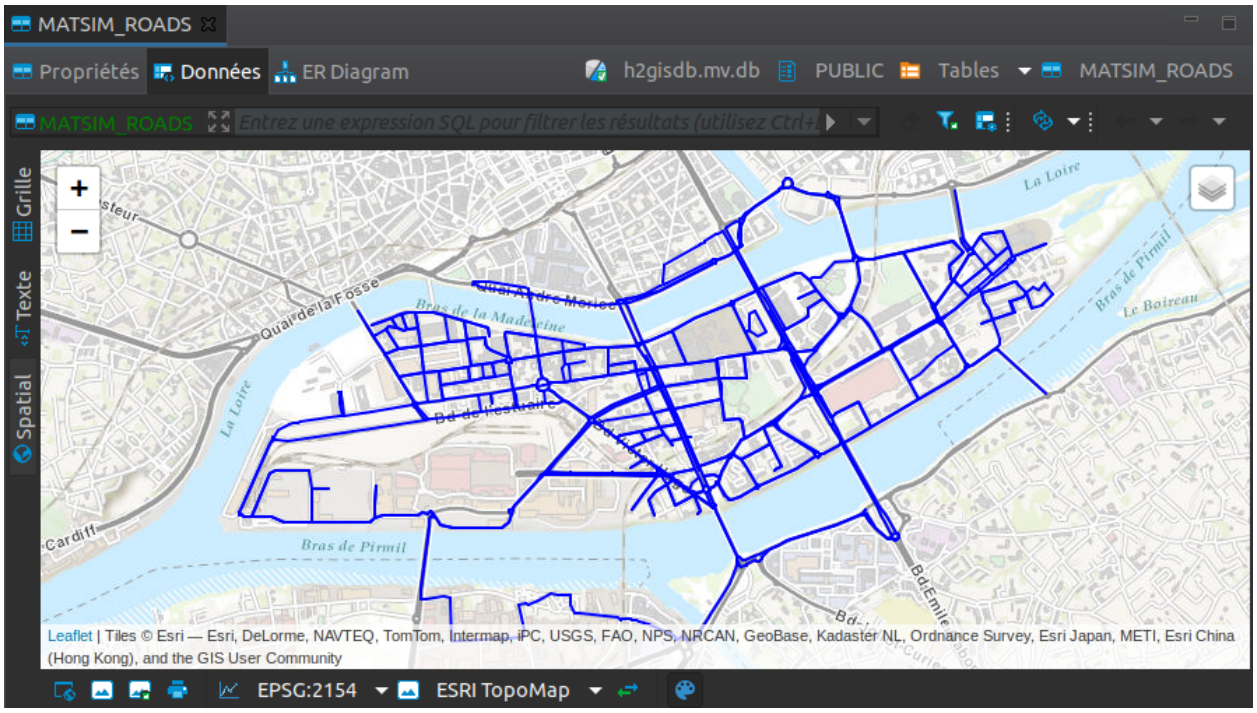
The next step consists in importing the activities locations from the MATSim simulation.In MATSim, activities are also called facilities.

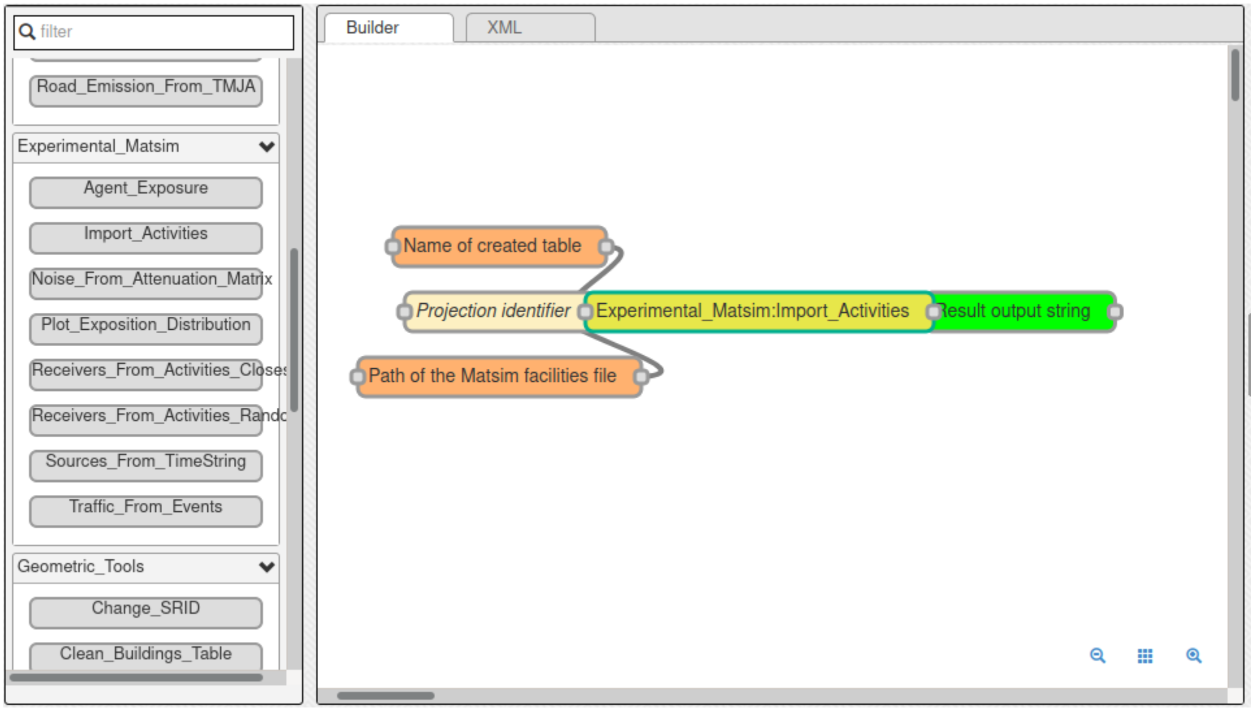
Let's use the `Import_Activities` WPS bloc. The inputs descriptions are quite straightforward :

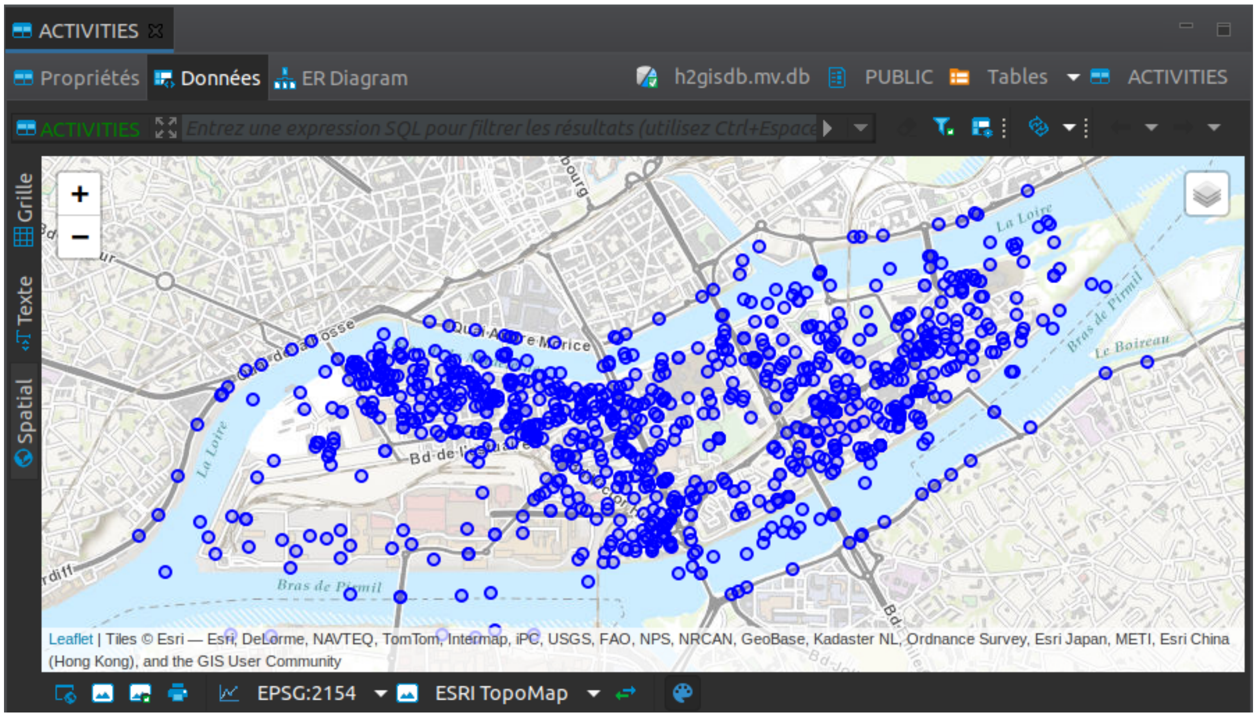
- Name of created table: `ACTIVITIES`
- Projection identifier: `2154`
- Path of MatSim facilities file: `/path/to/your/scenario_mastim/output_facilities.xml.gz`

You should end up with a `ACTIVITIES` table containing the activities location, and few other properties.


filter
Road_Emission_From_AADF
Road_Emission_From_TMJA
Experimental_Matsim
Agent_Exposure
Import_Activities
Noise_From_Attenuation_Matrix
Plot_Exposition_Distribution
Receivers_From_Activities_Closest
Receivers_From_Activities_Random
Sources_From_TimeString
Traffic_From_Events
Geometric_Tools
Change_SRID
Builder
XML
Export additionnal traffic data ?
Network CSV file
Calculate All vehicles noise source ?
Experimental_Matsim:Traffic_From_Events
result output str
Path of the Matsim output folder
Population Factor
Time Quantification
Output table name
Skip unused links ?
Projection identifier
List of agents ids to ignore in import


MATSIM_ROADS
Propriétés
Données
ER Diagram
h2gisdb.mv.db
PUBLIC
Tables
MATSIM_ROADS
Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+
Grille
Texte
Spatial
Leaflet | Tiles © Esri — Esri, DeLorme, NAVTEQ, TomTom, Intermap, iPC, USGS, FAO, NPS, NRCAN, GeoBase, Kadaster NL, Ordnance Survey, Esri Japan, METI, Esri China (Hong Kong), and the GIS User Community
EPSG:2154
ESRI TopoMap


filter
Road_Emission_From_TMJA
Experimental_Matsim
Agent_Exposure
Import_Activities
Noise_From_Attenuation_Matrix
Plot_Exposition_Distribution
Receivers_From_Activities_Closest
Receivers_From_Activities_Rando
Sources_From_TimeString
Traffic_From_Events
Geometric_Tools
Change_SRID
Clean_Buildings_Table
Builder
XML
Name of created table
Projection identifier
Experimental_Matsim:Import_Activities
Result output string
Path of the Matsim facilities file


ACTIVITIES
Propriétés
Données
ER Diagram
h2gisdb.mv.db
PUBLIC
Tables
ACTIVITIES
Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+Espace
Grille
Texte
Spatial
La Loire
Bras de Pirmil
Le Boireau
Leaflet | Tiles © Esri — Esri, DeLorme, NAVTEQ, TomTom, Intermap, iPC, USGS, FAO, NPS, NRCAN, GeoBase, Kadaster NL, Ordnance Survey, Esri Japan, METI, Esri China (Hong Kong), and the GIS User Community
EPSG:2154
ESRI TopoMap

5.17.7 Step 4 : Assign a Receiver to each Activity

Now, if you look closely, activities are placed in unorthodox locations, sometimes in the river, sometimes in buildings, etc. This is irrelevant for a MATSim simulation but here we want to calculate noise levels, so we need properly placed receivers.

So we want to assign a properly placed receiver for every activity we imported. We do that in 2 steps :

1. we calculate all the "valid" receiver positions using the `Building_Grid` WPS bloc
2. we choose, for each activity the right receiver.

There are 2 ways to execute step 4.2. We can simply choose the closest receiver for every activity, using the `Receivers_From_Activity_Closest` WPS bloc. Or we can randomly choose a receiver on the closest building of each activity using the `Receivers_From_Activity_Random` WPS bloc.

Here we are going to use the latter way, the random one.

Let's calculate all the receivers around our buildings using the `Building_Grid` WPS bloc with the following inputs :

- Buildings table table : `BUILDINGS`
- Distance between receivers : `5.0`
- height : `4.0`

That will place receviers around all the buildings, at 4 meter high and 5 meters apart.

Now, we must use the `Receivers_From_Activity_Random` WPS bloc. The inputs are simple, you just have to specify the names of the previously created tables

- Name of created table: `ACTIVITY_RECEIVERS`
- Name of the table containing the activities: `ACTIVITIES`
- Name of the table containing the buildings: `BUILDINGS`
- Name of the table containing the receivers: `RECEIVERS`

You should end up with a `ACTIVITY_RECEIVERS` table containing the new location (`THE_GEOM`, in blue below) as well as the orignal matsim position (`ORIGIN_GEOM`, in red below). You can inspect the results to see where each activity is placed now.

5.17.8 Step 5 : Calculate Noise Attenuation Matrix

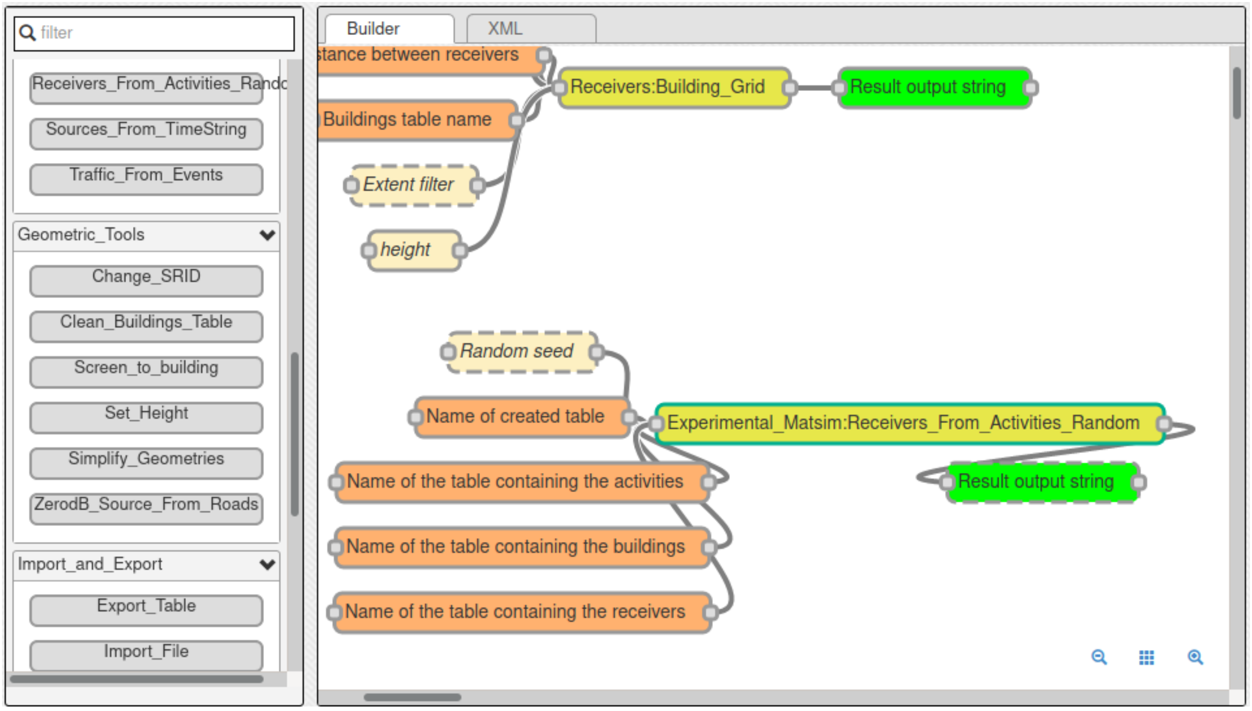
In this step, we want to calculate and store the noise propagation part of NoiseModelling. We need this because we actually have several power spectrum for every road segment, one for every timestep of 15min. In the end we want to have a noise map every 15 minutes (96 maps in total). If we do that directly, by calling something like `Noise_level_from_source` WPS bloc 96 times, we would be calculating the exact same noise propagation 96 times.

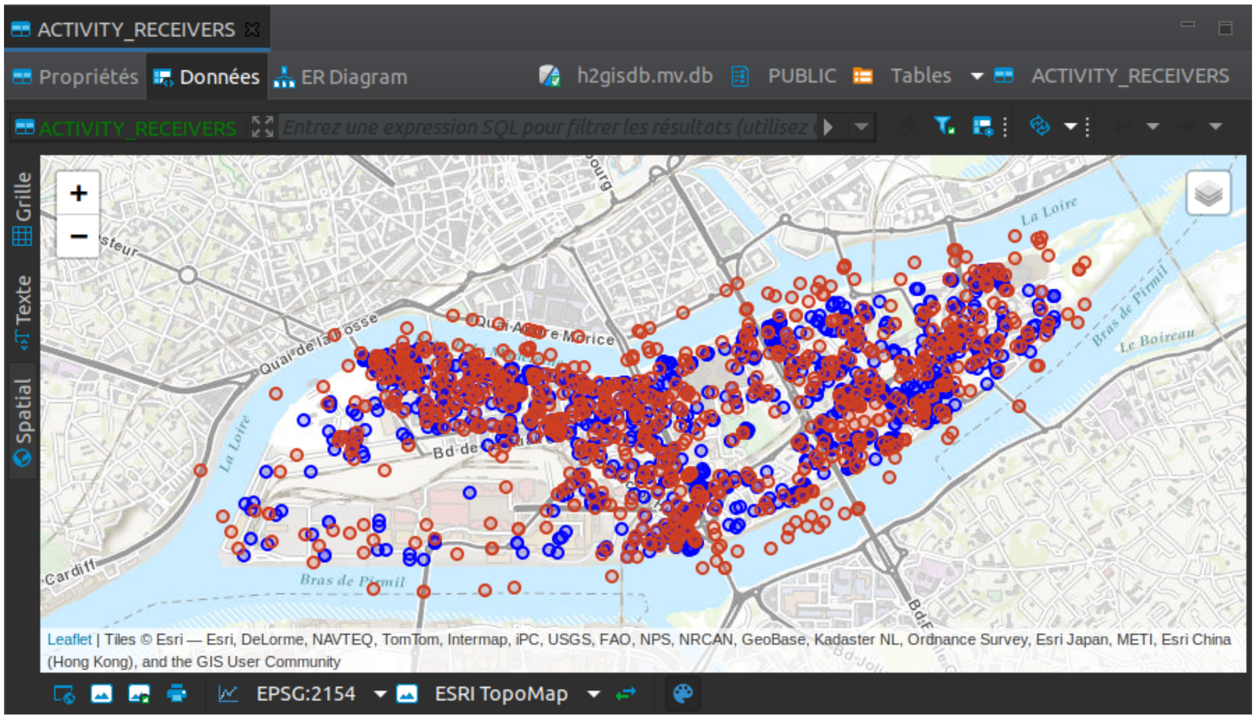
So the process is as follows :

1. we generate a SOURCE table, using the `MATSIM_ROADS` table, where all levels are set to 0 dB.
2. We use that table as input of the `Noise_level_from_source` WPS bloc and setting the `confExportSourceId` input paramter.

The `confExportSourceId` parameter will actually ouput, for every recevier, the list of sources that contribute to the resulting levels, with the source-receiver noise attenuation.

We'll then use this attenuation matrix in the next steps to get the 96 noise maps.


filter
Receivers_From_Activities_Rando
Sources_From_TimeString
Traffic_From_Events
Geometric_Tools
Change_SRID
Clean_Buildings_Table
Screen_to_building
Set_Height
Simplify_Geometries
ZerodB_Source_From_Roads
Import_and_Export
Export_Table
Import_File
Builder
XML
stance between receivers
Buildings table name
Extent filter
height
Receivers:Building_Grid
Result output string
Random seed
Name of created table
Experimental_Matsim:Receivers_From_Activities_Random
Name of the table containing the activities
Result output string
Name of the table containing the buildings
Name of the table containing the receivers


ACTIVITY_RECEIVERS
Propriétés
Données
ER Diagram
h2gisdb.mv.db
PUBLIC
Tables
ACTIVITY_RECEIVERS
ACTIVITY_RECEIVERS
Entrez une expression SQL pour filtrer les résultats (utilisez
Grille
Texte
Spatial
Leaflet | Tiles © Esri — Esri, DeLorme, NAVTEQ, TomTom, Intermap, iPC, USGS, FAO, NPS, NRCAN, GeoBase, Kadaster NL, Ordnance Survey, Esri Japan, METI, Esri China (Hong Kong), and the GIS User Community
EPSG:2154
ESRI TopoMap

Create the 0dB Source table

Here we'll use the `ZerodB_Source_From_Roads` WPS bloc. It's 2 inputs parameters are quite simple and should be set as follows :

- Input table name: `MATSIM_ROADS`
- Output table name: `SOURCES_0DB`

Calculate the attenuation matrix

Let's use the previously generated table to launch our propagation calculation.

As explained before, we'll use the Noise_level_from_source WPS bloc with the 'confExportSourceId' parameter enabled. For more details about the different parameters, browse the NoiseModelling general documentation.

The parameters we will use are the following :

- Buildings table name: `BUILDINGS`
- Receivers table name: `ACTIVITY_RECEIVERS`
- Sources table name: `SOURCES_0DB`
- Maximum source-receiver distance: `250`
- Maximum source reflexion distance: `50`
- Order of reflexion: `1`
- Do not compute LEVENING_GEOM table: `true`
- Do not compute LNIGHT_GEOM table: `true`
- Do not compute LDEN_GEOM table: `true`
- Separate receiver level by source identifier: `true`
- Diffraction on vertical edges: `false`
- Diffraction on horizontal edges: `true`
- Thread number: `4` (your number of available cpu core)

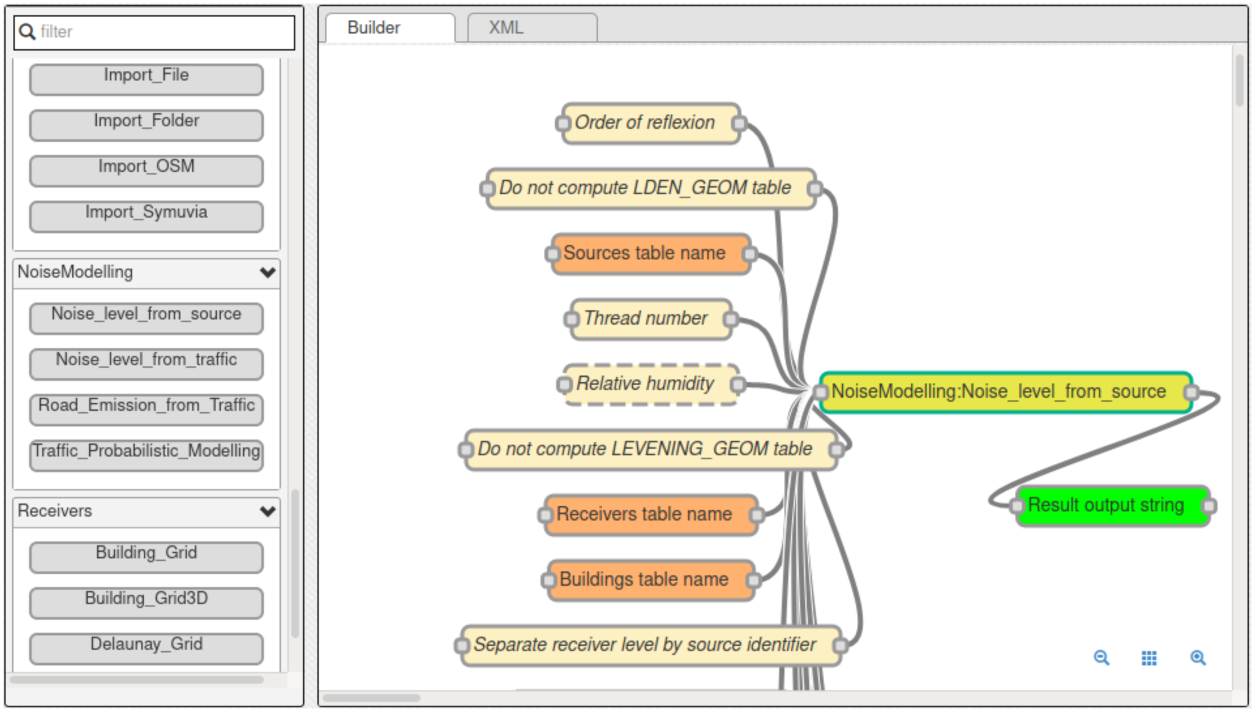
We should end up with a table called `LDAY_GEOM` that contains a list of contributing source attenuation for every receiver. We can see such a list for the receiver n°1 in the figure below:

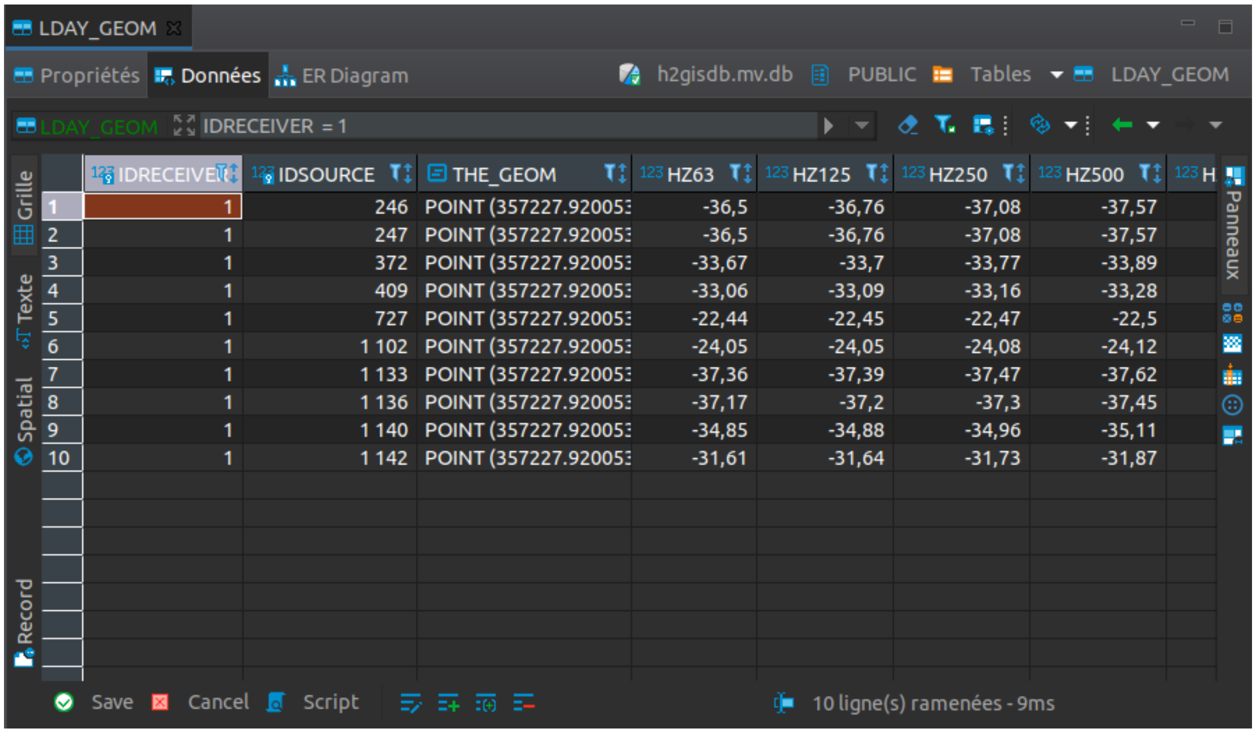
5.17.9 Step 6 : Calculate Noise Maps

We have noise power levels every 15 minutes in the `MATSIM_ROADS_STATS` table, and a source-receiver noise attenuation matrix in the `LDAY_GEOM` table. We just need to combine the two to get receivers noise levels, noise maps, every 15 minutes.

This is the purpose of the `Noise_From_Attenuation_Matrix` WPS bloc. We just have set the right tables as input as follows :

- Attenuation matrix table name: `LDAY_GEOM`
- Output table name: `RESULT_GEOM`
- Table name of the MATSIM table containing the roads LW stats per timeString: `MATSIM_ROADS_STATS`
- Table name of the MATSIM table containing the roads geometries: `MATSIM_ROADS`

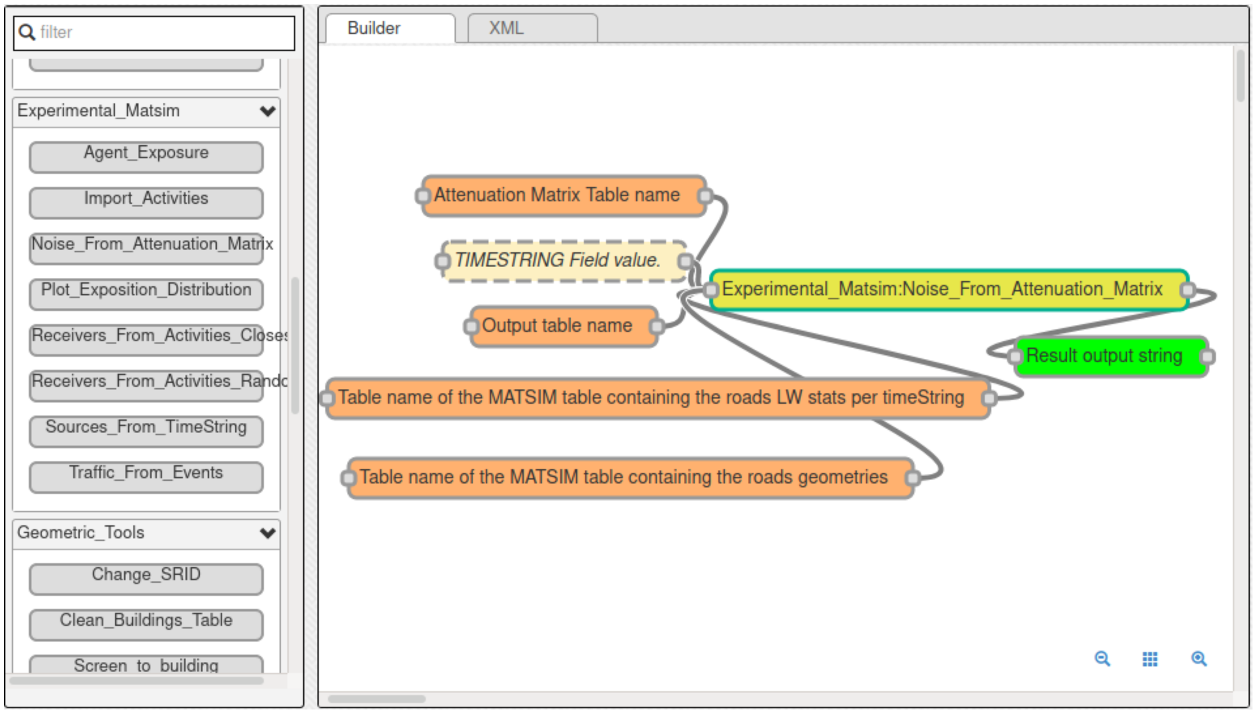




LDAY_GEOM IDRECEIVER = 1

	IDRECEIVER	IDSOURCE	THE_GEOM	HZ63	HZ125	HZ250	HZ500	H
1	1	246	POINT (357227.92005	-36,5	-36,76	-37,08	-37,57	
2	1	247	POINT (357227.92005	-36,5	-36,76	-37,08	-37,57	
3	1	372	POINT (357227.92005	-33,67	-33,7	-33,77	-33,89	
4	1	409	POINT (357227.92005	-33,06	-33,09	-33,16	-33,28	
5	1	727	POINT (357227.92005	-22,44	-22,45	-22,47	-22,5	
6	1	1 102	POINT (357227.92005	-24,05	-24,05	-24,08	-24,12	
7	1	1 133	POINT (357227.92005	-37,36	-37,39	-37,47	-37,62	
8	1	1 136	POINT (357227.92005	-37,17	-37,2	-37,3	-37,45	
9	1	1 140	POINT (357227.92005	-34,85	-34,88	-34,96	-35,11	
10	1	1 142	POINT (357227.92005	-31,61	-31,64	-31,73	-31,87	

10 ligne(s) ramenées - 9ms


filter
Experimental_Matsim
Agent_Exposure
Import_Activities
Noise_From_Attenuation_Matrix
Plot_Exposition_Distribution
Receivers_From_Activities_Closes
Receivers_From_Activities_Rando
Sources_From_TimeString
Traffic_From_Events
Geometric_Tools
Change_SRID
Clean_Buildings_Table
Screen_to_building
Builder
XML
Attenuation Matrix Table name
TIMESTRING Field value.
Experimental_Matsim:Noise_From_Attenuation_Matrix
Output table name
Result output string
Table name of the MATSIM table containing the roads LW stats per timeString
Table name of the MATSIM table containing the roads geometries

It takes some time but in the end you should get a noise spectrum for every receiver every 15 minutes in the table `RESULT_GEOM`.

We have our noise maps !

5.17.10 Visualization

Export the data

Here we'll look at a nice way to look at the results with QGIS.

First we need to export the `RESULT_GEOM` table data into a Shapefile. We'll simply use the `Export_Table WPS` bloc with the following parameters :

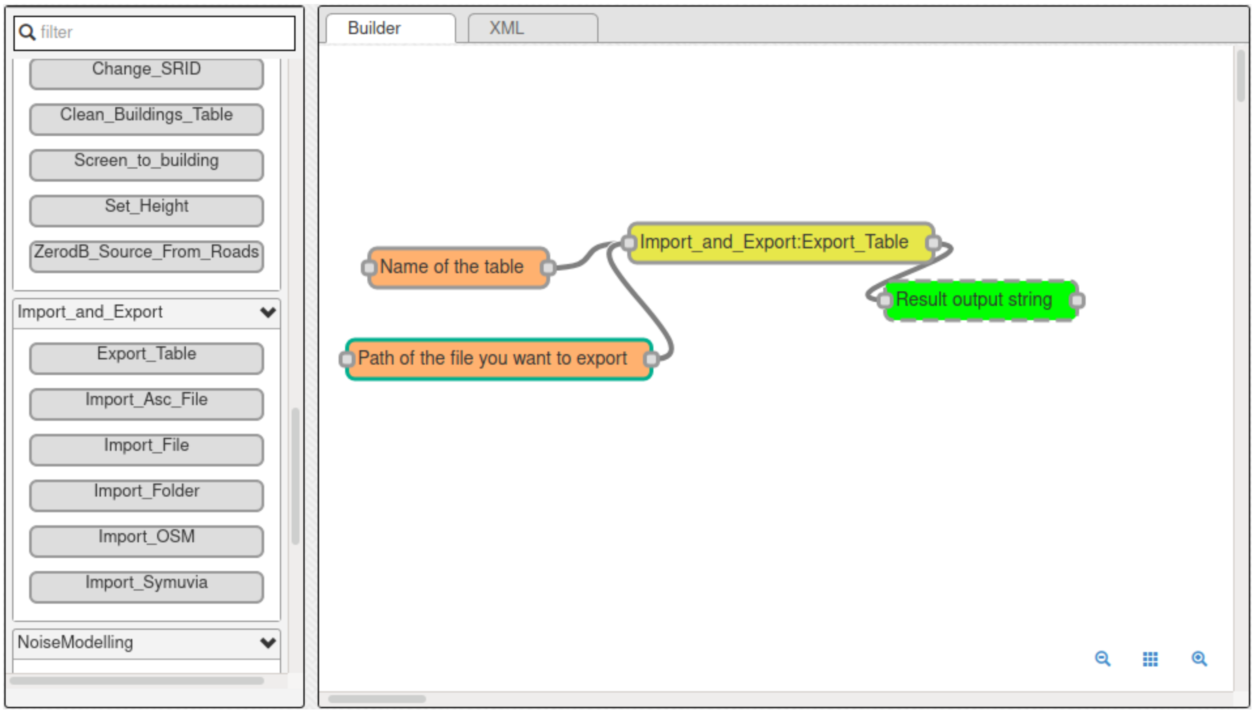
- Name of the table: `RESULT_GEOM`
- Path of the file you want to export: `/path/to/wherever/results.shp`

View it in QGIS

Let's go into QGIS. We are going to import 2 layers : an osm background and our results.

Note: For those who are new to GIS and want to get started with QGIS, we advise you to follow this tutorial as a start.

- In `Layer Add Layer Add vector layer`, you can enter the path of your `results.shp` file. Then click on `Add`.
- In `Layer Add Layer Add XYZ Layer`, you can add the OpenStreetMap background.



You should see a lot of points all of the same color.

We now need to choose a timeslice we want to visualize, let's pick `10h00_10h15`. If you right click on the receivers layer and click on `Filter...` you should see the filter dialog.

To filter results for the 10h00_10h15 time period you can enter the following filter query :

```
TIMESTRING = '10h00_10h15'
```

The last step is to color the dots based on the LEQA field. Here is my configuration :

And the final result, between 10h00 and 10h15 :

5.18 Pilot NoiseModelling with scripts

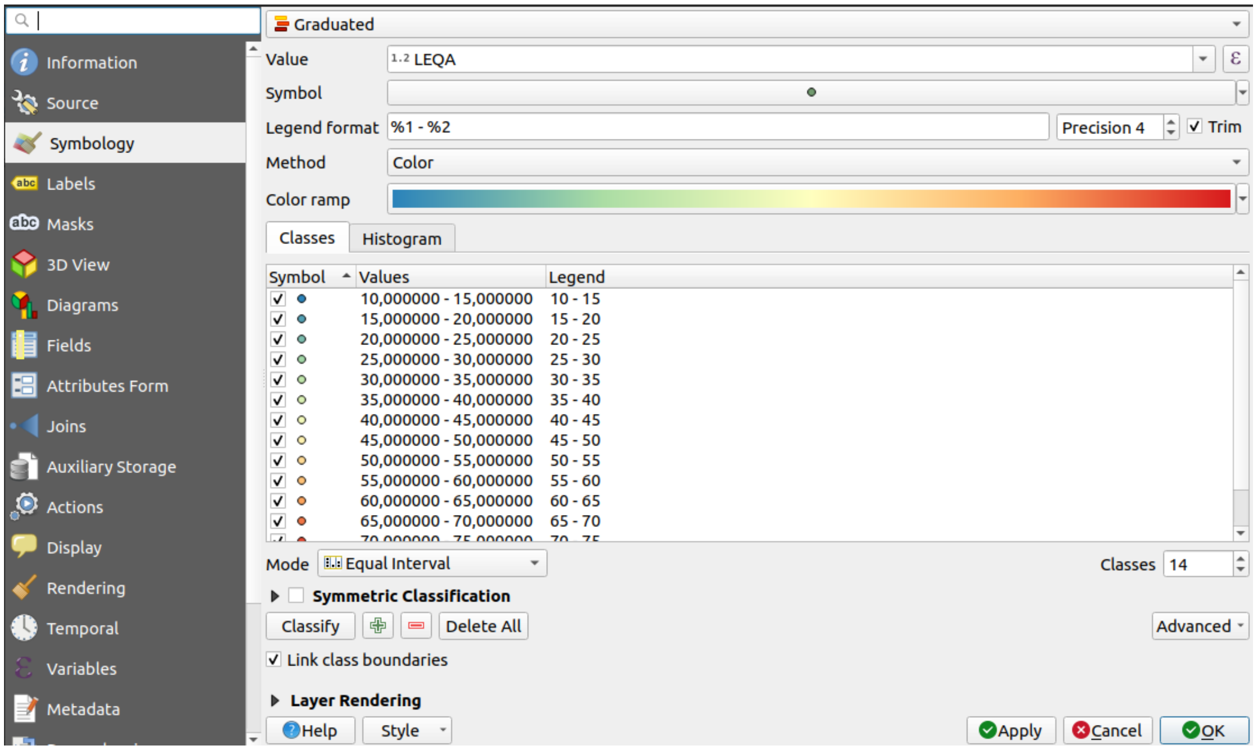
In this tutorial, we describe the different ways to pilot NoiseModelling thanks to scripts. To do so, we will use a dedicated packaging of NoiseModelling, called `NoiseModelling_4.0.0_without_gui`, in which the GUI has been removed (no more Geoserver and *WPS Builder*).

1. Go to the NoiseModelling latest release page
2. Download and unzip the NoiseModelling_4.0.0_without_gui file

From that point, NoiseModelling can be executed in 3 different maners:

1. with simple command lines
2. with Bash script
3. with Groovy script

To illustrate, users are invited to reproduce the tutorial "*Get Started - GUI*" in command lines.



Graduated

Value: 1.2 LEQA

Symbol

Legend format: %1 - %2 Precision 4 ✓ Trim

Method: Color

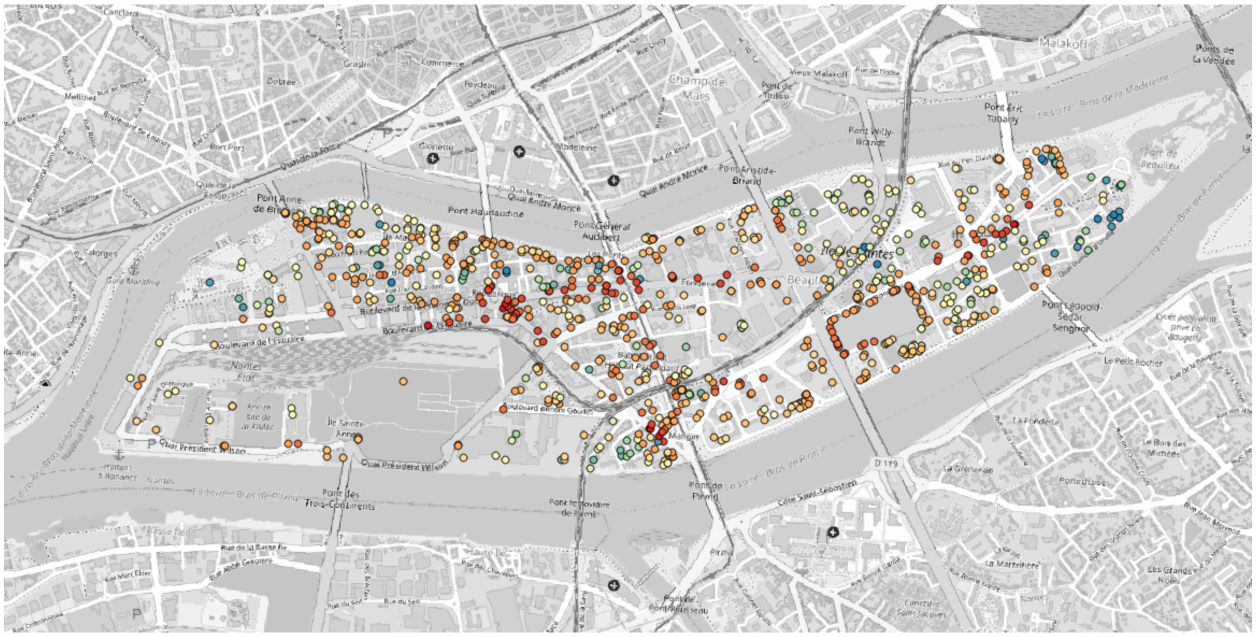
Color ramp

Symbol	Values	Legend
✓ ●	10,000000 - 15,000000	10 - 15
✓ ●	15,000000 - 20,000000	15 - 20
✓ ●	20,000000 - 25,000000	20 - 25
✓ ●	25,000000 - 30,000000	25 - 30
✓ ●	30,000000 - 35,000000	30 - 35
✓ ●	35,000000 - 40,000000	35 - 40
✓ ●	40,000000 - 45,000000	40 - 45
✓ ●	45,000000 - 50,000000	45 - 50
✓ ●	50,000000 - 55,000000	50 - 55
✓ ●	55,000000 - 60,000000	55 - 60
✓ ●	60,000000 - 65,000000	60 - 65
✓ ●	65,000000 - 70,000000	65 - 70

Mode: Equal Interval Classes: 14

Symmetric Classification

✓ Link class boundaries



Note: This tutorial is mainly dedicated to advanced users.

Warning: The URL is here adapted to Linux or Mac users. Windows user may adapt the address by replacing / by \ and the drive name.

5.18.1 Requirements

Warning: For all users (**Linux** , **Mac** and **Windows**), please make sure your Java environment is well setted. For more information, please read the page *Requirements*.

5.18.2 1. Simple command line

Below is an example of a bash instruction, executing the `Noise_level_from_traffic.groovy` WPS Script (located in the directory `/noisemodelling/wps/`). This block has 5 arguments corresponding to the input table names (for buildings, roads, receivers, dem and ground type).

```
cd /home/user/NoiseModelling_4.0.0_without_gui/

./bin/wps_scripts -w ./ -s noisemodelling/wps/Import_and_Export/Import_File.groovy -
↪pathFile resources/org/noise_planet/noisemodelling/wps/ground_type.shp
```

`./bin/wps_scripts` instruction allows to launch the `wps_scripts.sh` or `wps_scripts.bat` *(depending on if you are on Linux / Mac or Windows)* file, which is located in the `bin/` directory.

Warning: Adapt `/home/user/` address with your own situation

5.18.3 2. Bash script

Below is an example of a sequence of simple .groovy scripts, using bash instructions and launching the differents steps described in the "*Get Started - GUI*".

```
#! /bin/bash

# Run the get started turorial
# https://noisemodelling.readthedocs.io/en/latest/Get_Started_Tutorial.html

# Step 4: Upload files to database
# create (or load existing) database and load a shape file into the database
./bin/wps_scripts -w ./ -s noisemodelling/wps/Import_and_Export/Import_File.groovy -
↪pathFile resources/org/noise_planet/noisemodelling/wps/ground_type.shp
./bin/wps_scripts -w ./ -s noisemodelling/wps/Import_and_Export/Import_File.groovy -
↪pathFile resources/org/noise_planet/noisemodelling/wps/buildings.shp
./bin/wps_scripts -w ./ -s noisemodelling/wps/Import_and_Export/Import_File.groovy -
↪pathFile resources/org/noise_planet/noisemodelling/wps/receivers.shp
./bin/wps_scripts -w ./ -s noisemodelling/wps/Import_and_Export/Import_File.groovy -
↪pathFile resources/org/noise_planet/noisemodelling/wps/ROADS2.shp
./bin/wps_scripts -w ./ -s noisemodelling/wps/Import_and_Export/Import_File.groovy -
↪pathFile resources/org/noise_planet/noisemodelling/wps/dem.geojson
```

(continues on next page)

(continued from previous page)

```


# Step 5: Run Calculation
./bin/wps_scripts -w ./ -s noisemodelling/wps/NoiseModelling/Noise_level_from_traffic.
↪groovy -tableBuilding BUILDINGS -tableRoads ROADS2 -tableReceivers RECEIVERS -
↪tableDEM DEM -tableGroundAbs GROUND_TYPE

# Step 6: Export (& see) the results
./bin/wps_scripts -w ./ -s noisemodelling/wps/Import_and_Export/Export_Table.groovy -
↪exportPath LDAY_GEOM.shp -tableToExport LDAY_GEOM
```

5.18.4 3. Groovy script

Below is an example of a complex .groovy script, launching the differents steps described in the "*Get Started - GUI*".

```
/**
 * NoiseModelling is an open-source tool designed to produce environmental noise maps
 * on very large urban areas. It can be used as a Java library or be controlled␣
↪through
 * a user friendly web interface.
 *
 * This version is developed by the DECIDE team from the Lab-STICC (CNRS) and by the
 * Mixt Research Unit in Environmental Acoustics (Université Gustave Eiffel).
 * <http://noise-planet.org/noisemodelling.html>
 *
 * NoiseModelling is distributed under GPL 3 license. You can read a copy of this
 * License in the file LICENCE provided with this software.
 *
 * Contact: contact@noise-planet.org
 */

/**
 * @Author Pierre Aumond, Université Gustave Eiffel
 * @Author Nicolas Fortin, Université Gustave Eiffel
 */

import org.h2gis.api.ProgressVisitor
import org.slf4j.Logger
import org.slf4j.LoggerFactory
import java.sql.Connection

title = 'Tutorial script'
description = 'Long description of tutorial script'

inputs = []

outputs = [result: [name: 'Result output string', title: 'Result output string',␣
↪description: 'This type of result does not allow the blocks to be linked together.',
↪ type: String.class]]


def runScript(connection, scriptFile, arguments) {
    Logger logger = LoggerFactory.getLogger("script")
    GroovyShell shell = new GroovyShell()
    Script scriptInstance = shell.parse(new File(scriptFile))
```

(continues on next page)

(continued from previous page)

```
    Object result = scriptInstance.invokeMethod("exec", [connection, arguments])
    if(result != null) {
        logger.info(result.toString())
    }
}

def exec(Connection connection, input) {

  // Step 4: Upload files to database
  runScript(connection, "noisemodelling/wps/Import_and_Export/Import_File.groovy",
        ["pathFile":"resources/org/noise_planet/noisemodelling/wps/ground_type.shp"])

  runScript(connection, "noisemodelling/wps/Import_and_Export/Import_File.groovy",
        ["pathFile":"resources/org/noise_planet/noisemodelling/wps/buildings.shp"])

  runScript(connection, "noisemodelling/wps/Import_and_Export/Import_File.groovy",
        ["pathFile":"resources/org/noise_planet/noisemodelling/wps/receivers.shp"])

  runScript(connection, "noisemodelling/wps/Import_and_Export/Import_File.groovy",
        ["pathFile":"resources/org/noise_planet/noisemodelling/wps/ROADS2.shp"])

  runScript(connection, "noisemodelling/wps/Import_and_Export/Import_File.groovy",
        ["pathFile":"resources/org/noise_planet/noisemodelling/wps/dem.geojson"])

  // Step 5: Run Calculation
  runScript(connection, "noisemodelling/wps/NoiseModelling/Noise_level_from_traffic.
↪groovy",
        ["tableBuilding":"BUILDINGS", "tableRoads":"ROADS2", "tableReceivers":
↪"RECEIVERS",
         "tableDEM":"DEM", "tableGroundAbs":"GROUND_TYPE"])

  // Step 6: Export (& see) the results
  runScript(connection, "noisemodelling/wps/Import_and_Export/Export_Table.groovy",
        ["exportPath":"LDAY_GEOM.shp", "tableToExport":"LDAY_GEOM"])
}
```

You can find this script online here

5.19 Tutorials - FAQ

5.19.1 Shapefiles or GeoJSON?

Shapefile is a file format for geographic information systems (GIS).

Its extension is classically `.shp`, and it is always accompanied by (at least) two other files with the same name:

- `.dbf`, a file that contains attribute data relating to the objects contained in the shapefile,
- `.shx`, file that stores the index of the geometry.

Other files can also be provided :

- `.prj` - coordinate system information, using the WKT (Well Known Text) format;
- `.sbn` and `.sbx` - spatial shape index;
- `.fbn` and `.fbx` - spatial shape index for read-only shapefiles;

- `.ain` and `.aih` - attribute index for active fields in a table or in a theme attribute table;
- etc.

GeoJSON (Geographic JSON) is an open format for encoding simple geospatial data sets using the JSON (JavaScript Object Notation) standard. It is an alternative to the Shapefile format. It has the advantage of being readable directly in a text editor.

5.19.2 PostGreSQL or H2?

PostgreSQL & H2 Database are two DataBase Management Systems (DBMS). They are used to store, manipulate or manage, and share information in a database, ensuring the quality, permanence and confidentiality of the information, while hiding the complexity of the operations. NoiseModelling can connect to DBMS in H2 - H2GIS or PostGreSQL - PostGIS format.

5.19.3 OSM

OpenStreetMap (OSM) is a collaborative project to create a free (and open-access) editable map of the world. The geodata underlying the map is considered the primary output of the project. The creation and growth of OSM has been motivated by restrictions on use or availability of map data across much of the world, and the advent of inexpensive portable satellite navigation devices. OSM is considered a prominent example of Volunteered Geographic Information (VGI).

5.19.4 Metric SRID

Spatial reference systems can be referred to using a **SRID integer**, including EPSG codes.

In several input files, you need to specify coordinates, *e.g* road network. It is strongly suggested not to use WGS84 coordinates (i.e. GPS coordinates - `EPSG:4326`). Acoustic propagation formulas make the assumption that coordinates are metric. Many countries and regions have custom coordinate system defined, optimized for usages in their appropriate areas. It might be best to ask some GIS specialists in your region of interest what the most commonly used local coordinate system is and use that as well for your data. If you don't have any clue about what coordinate system is used in your region, it might be best to use the Universal Transverse Mercator coordinate system. This coordinate system divides the world into multiple bands, each six degrees width and separated into a northern and southern part, which is called UTM zones (see http://en.wikipedia.org/wiki/UTM_zones#UTM_zone for more details). For each zone, an optimized coordinate system is defined. Choose the UTM zone which covers your region (Wikipedia has a nice map showing the zones) and use its coordinate system.

Here is the map : https://upload.wikimedia.org/wikipedia/commons/e/ed/Utm-zones.jpg

Note: We recommand using the website https://epsg.io/ to find the appropriate **SRID** code for your location.

5.19.5 Primary Key

"In the design of a database table, the Primary Key (abbreviated PK) is selected among the non-empty set of candidate keys. The PK is a column (or an irreducible group of columns) able to identify every row of the table." (Source)

5.20 WPS Blocks

5.20.1 WPS general presentation

The OGC Web Processing Service (WPS) Interface Standard provides rules for standardizing inputs and outputs (requests and responses) for invoking geospatial processing services, such as polygon overlay, as a web service.

The WPS standard defines how a client can request the execution of a process, and how the output from the process is handled. It defines an interface that facilitates the publishing of geospatial processes and clients' discovery of and binding to those processes.

5.20.2 NoiseModelling and WPS

Since release v.3.0.0, NoiseModelling comes with various WPS scripts, encapsulated in so-called blocks. These blocks, written in Groovy language, are executed thanks to the GeoServer WPS engine.

Physically stored as `.groovy` files *(openable in any text editor)*, they are located in the `NoiseModelling\\data_dir\\scripts\\wps\\` directory.

Tip: To know the functionality of each WPS block, wait a few moments with your mouse on the block, a tooltip text will appear.

Note: With each new version, new blocks are added. Be curious and check out the latest version!

5.20.3 Create your own WPS block

Please see Advanced Users Section, because now you want to be one!

5.21 WPS Builder

5.21.1 What is WPS Builder ?

WPS Builder allows to creates graphical process workflows that can be easily executed and reproduced. It allows Web Processing Services to operate through a user interface.

We have developed a version of WPS Builder adapted to the needs of NoiseModelling. This version being very close to the original version initially developped by former Boundlessgeo company.

5.21.2 Frequently Asked Question

What do the colors correspond to?

- Orange block are mandatory
- Beige blocks are optional
- Green block are unfortunately useless *(not due to NoiseModelling)*

- Blocks get solid border when they are ready

Can I save my WPS Builder project?

Yes. To save your WPS Builder project you two possibilities:

1. Save in the local browser storage
2. Export into a JSON file

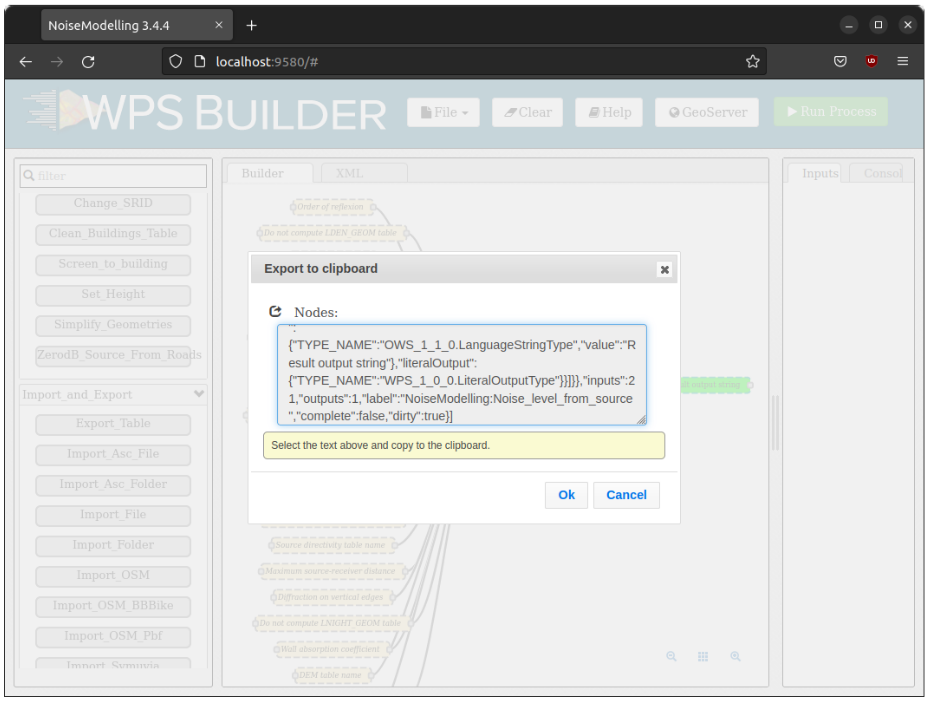
1. Local browser

Click on the `File` icon and then choose `Save to local browser storage`. This way, your working environment will saved in the memory of your web browser.

Once you restart NoiseModelling, you can reload this environment by clicking on `File / Open from local browser storage`

2. Export into JSON

Click on the `File` icon and then choose `Export to clipboard`. In the opening panel, you have a JSON text that you can copy / paste and save into a `` ` ``.txt or `json` file.



Export to clipboard

Nodes:

```
":
{"TYPE_NAME":"OWS_1_1_0.LanguageStringType","value":"R
esult output string"},"literalOutput":
{"TYPE_NAME":"WPS_1_0_0.LiteralOutputType"}}]}},"inputs":2
1,"outputs":1,"label":"NoiseModelling:Noise_level_from_source
","complete":false,"dirty":true}]
```

Select the text above and copy to the clipboard.

Ok Cancel

Once you restart NoiseModelling, you can reload this environment by clicking on `File / Import clipboard`. In the opening panel, paste your JSON text and click `Ok`.

Why everything is wrong when I use "Enter"?

Don't click on your `Enter` keyboard key, it refreshes web page.

I can't link process block between them?

It is normal. . . this feature has not yet been implemented!

5.22 Create your own WPS block

5.22.1 Presentation

The OGC Web Processing Service (WPS) Standard provides rules for standardizing inputs and outputs (requests and responses) for invoking geospatial processing services as a web service.

WPS scripts for NoiseModelling are written in Groovy language. They are located in the `NoiseModelling/data_dir/scripts/wps` directory.

To help you build your WPS script, you will find a template in the `NoiseModelling/data_dir/scripts/template` directory

Note: Don't be shy, if you think your script can be useful to the community, you can redistribute it using github or by sending it directly to us.

Tip: The best way to make your own WPS is to be inspired by those that are already made. See how the tutorial is built or contact us for many more examples.

5.22.2 General Structure

1. Import used libraries

```
import geoserver.GeoServer
import geoserver.catalog.Store
```

2. WPS Script meta data

```
title = '....'
description = '.....'
```

3. WPS Script input & output

```
inputs = [
    inputparameter1: [name: '...', description : '...', title: '...', type: String.
↪class],
    inputparameter2: [name: '...', description : '...', title: '...', type: String.
↪class]
]

outputs = [
    ouputparameter: [name: '...', title: '...', type: String.class]
]
```

4. Set connection method

```
def static Connection openPostgreSQLDataStoreConnection() {
    Store store = new GeoServer().catalog.getStore("h2gisdb")
    JDBCDataStore jdbcDataStore = (JDBCDataStore)store.getDataStoreInfo().
↪getDataStore(null)
    return jdbcDataStore.getDataSource().getConnection()
}
```

5. Set main method to execute

```
def run(input) {

    // Open connection and close it at the end
    openPostgreSQLDataStoreConnection(dbName).withCloseable { Connection connection ->
        // Execute code here
        // for example, run SQL command lines
        Sql sql = new Sql(connection)
        sql.execute("drop table if exists TABLETODROP")
    }

    // print to Console windows
    return [result : 'Ok ! ']
}
```

5.23 Access NoiseModelling database

5.23.1 Introduction

NoiseModelling has been preconfigured to use H2 / H2GIS as the default database (to store and manage all the needed data).

This database does not need to be configured or installed on the system. It's transparent to users.

Tip: Many spatial processing are possible with H2GIS. Please have a look to the numerous functions on the H2GIS website.

To visualize and manage NoiseModelling data (*e.g* roads, buildings or landcover layers) you have the choice between the three following approaches *(listed from simple to advanced)*:

1. Use WPS blocks (inside)
2. Use H2/H2GIS web client
3. Use DBeaver client

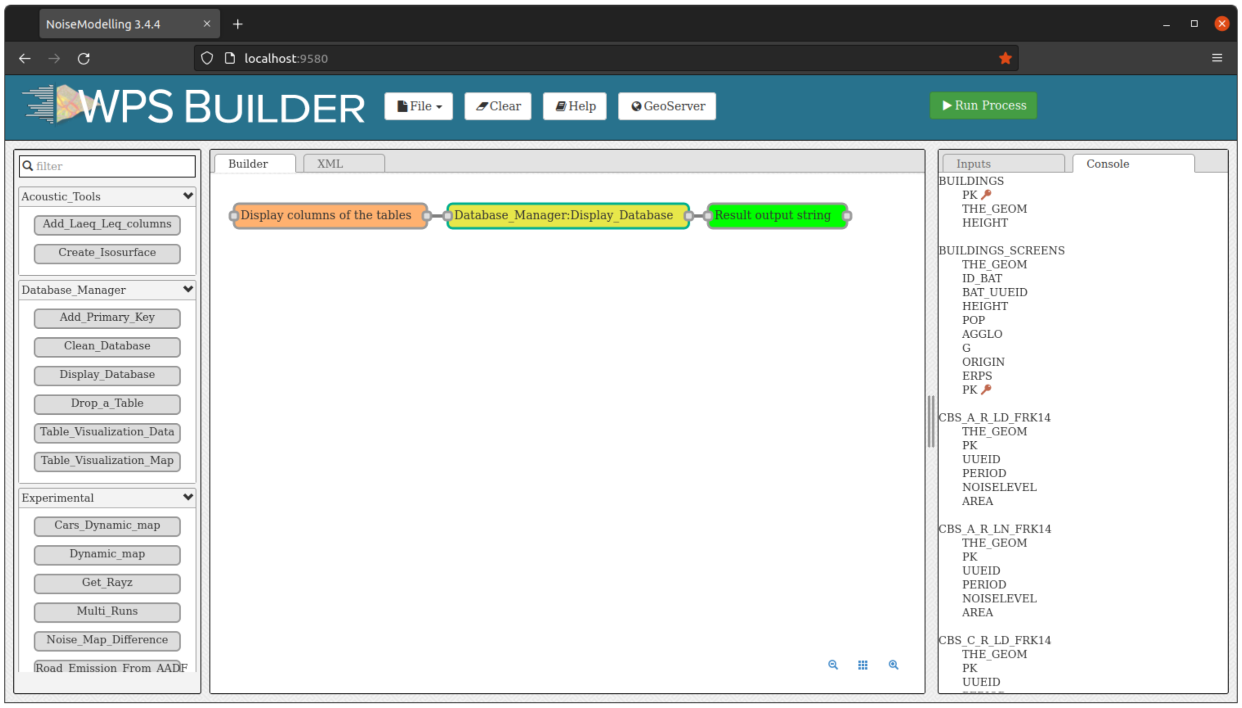
5.23.2 1. Use WPS blocks

Once NoiseModelling UI is launched (open http://localhost:9580/ in your web browser), you can manage your data thanks to the `Database_Manager` WPS blocks folder *(on the left side)*. In particular, you can do these actions:

- `Add_Primary_Key`: allows to add a primary key on a column of a specific layer (table)

- `Clean_Database`: remove all the layers (tables) from NoiseModelling *(can be useful when starting a new project)*
- `Display_Database`: list all the layers (tables) and the columns inside
- `Drop_a_Table`: remove the selected layer (table) from NoiseModelling
- `Table_Visualization_Data`: display the layer (table) as an array of values
- `Table_Visualization_Map`: if the layer (table) is geographic (contains geometry(ies)), display the data in a map

Below is an illustration with the `Display_Database` WPS block



```
BUILDINGS
    PK
    THE_GEOM
    HEIGHT

BUILDINGS_SCREENS
    THE_GEOM
    ID_BAT
    BAT_UUEID
    HEIGHT
    POP
    AGGLO
    G
    ORIGIN
    ERPS
    PK

CBS_A_R_LD_FRK14
    THE_GEOM
    PK
    UUEID
    PERIOD
    NOISELEVEL
    AREA

CBS_A_R_LN_FRK14
    THE_GEOM
    PK
    UUEID
    PERIOD
    NOISELEVEL
    AREA

CBS_C_R_LD_FRK14
    THE_GEOM
    PK
    UUEID
```

5.23.3 2. Use H2/H2GIS web client

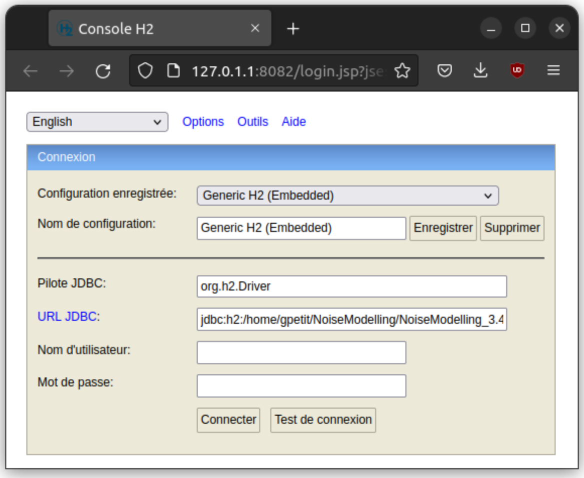
If you want to have full capabilities on visualization, edition and processing on data, you may need to connect to the database thanks to the H2/H2GIS web interface.

To do so, follow these steps:

1. download the H2/H2GIS v.2.0 database client (which is used with NoiseModelling 4.0)
2. unzip the `h2gis-dist-2.0.0-bin.zip` file
3. in the `/h2gis-standalone/` folder, double-click on the `h2gis-dist-2.0.0-SNAPSHOT`.jar` file to launch the web client *(depending on your Operating System, you may need to allow the execution of this file)*
4. the H2/H2GIS web client should be opened in your default web browser (the URL looks like this `http://127.0.0.1.1:8082/login.jsp?jsessionid=08ef3ad5d6838b614cf91b42e10bca8f`)

In the connexion panel, you have to specify the following informations:

- `Driver Class`: the driver that allows to connect to a specific database. Here we want to connect to a H2 db, so let the default value `org.h2.Driver`



English

Configuration enregistrée: Generic H2 (Embedded)

Nom de configuration: Generic H2 (Embedded)

Pilote JDBC: org.h2.Driver

URL JDBC: jdbc:h2:/home/gpetit/NoiseModelling/NoiseModelling_3.4

Nom d'utilisateur:

Mot de passe:

- `JDBC URL`: the JDBC address of the NoiseModelling database. By default, this database is placed in here `/.../data_dir/h2gisdb.mv.db`. So, fill this text area with `jdbc:h2:/.../data_dir/h2gisdb.mv.db`.
- `User name`: the db user name. By default, keep the empty value
- `Password`: the db password. By default, keep the empty value

Warning: If you want to open the database while NoiseModelling is running, you have to add `;AUTO_SERVER=TRUE` after the `JDBC URL`. If not, you will only be able to open the database once NoiseModelling is closed.

Below is an example, with a database located on the computer here: `/home/nm_user/NoiseModelling/NoiseModelling_4.0/data_dir/h2gisdb.mv.db`. We want to open the db while NoiseModelling is running.

- `JDBC URL`: `jdbc:h2:/home/nm_user/NoiseModelling/NoiseModelling_4.0/data_dir/h2gisdb.mv.db;AUTO_SERVER=TRUE`
- `User name`: *empty*
- `Password`: *empty*

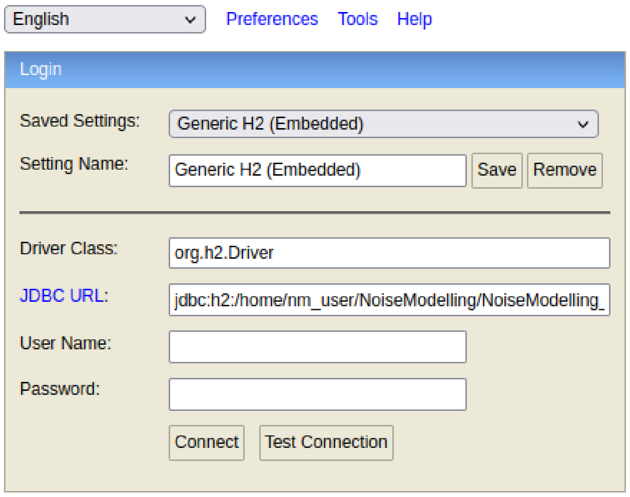
Warning: The URL is here adapted to Linux or Mac users. Windows user may adapt the address by replacing / by \ and the drive name.

Once done, click on `Connect`

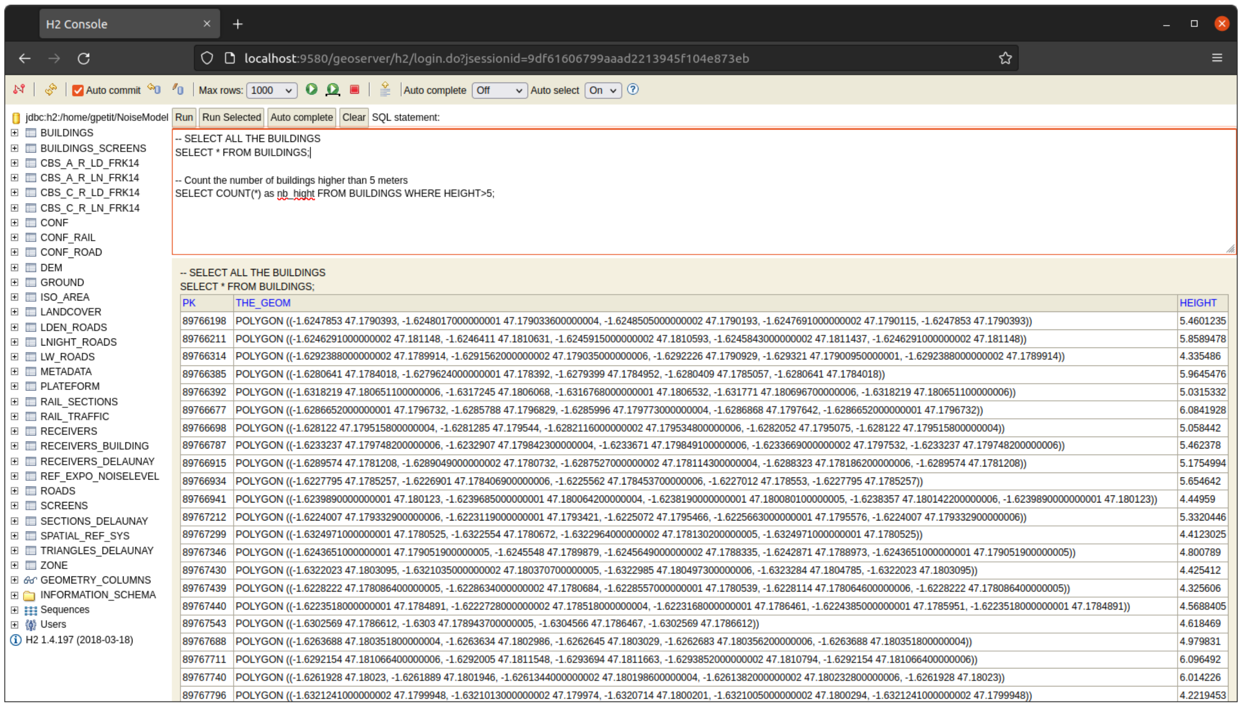
In the new interface, you discover a full database manager, with the list of tables on the left side, a SQL console (where you can execute all the instructions you want, independently of NoiseModelling) and a result panel.

5.23.4 3. Use DBeaver client

DBeaver is a free and open-source universal SQL / database client for developers and database administrators. DBeaver is able (among others) to connect to H2/H2GIS database or to PostgreSQL/PostGIS.



Saved Settings:	Generic H2 (Embedded)
Setting Name:	Generic H2 (Embedded)
Driver Class:	org.h2.Driver
JDBC URL:	jdbc:h2:/home/nm_user/NoiseModelling/NoiseModelling_
User Name:	
Password:	



```
-- SELECT ALL THE BUILDINGS
SELECT * FROM BUILDINGS;

-- Count the number of buildings higher than 5 meters
SELECT COUNT(*) as nb_hight FROM BUILDINGS WHERE HEIGHT>5;
```

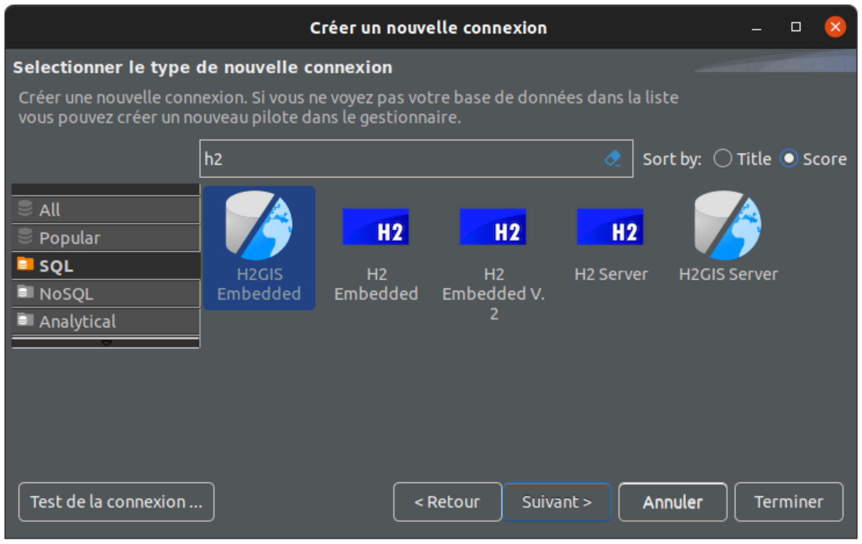
```
-- SELECT ALL THE BUILDINGS
SELECT * FROM BUILDINGS;
```

PK	THE_GEOM	HEIGHT
89766198	POLYGON ((-1.6247853 47.1790393, -1.6248017000000001 47.179033600000004, -1.6248505000000002 47.1790193, -1.6247691000000002 47.1790115, -1.6247853 47.1790393))	5.4601235
89766211	POLYGON ((-1.6246291000000002 47.181148, -1.6246411 47.1810631, -1.6245915000000002 47.1810593, -1.6245843000000002 47.1811437, -1.6246291000000002 47.181148))	5.8589478
89766314	POLYGON ((-1.6292388000000002 47.1789914, -1.6291562000000002 47.179035000000006, -1.6292226 47.1790929, -1.629321 47.17900950000001, -1.6292388000000002 47.1789914))	4.335486
89766385	POLYGON ((-1.6280641 47.1784018, -1.6279624000000001 47.178392, -1.6279399 47.1784952, -1.6280409 47.1785057, -1.6280641 47.1784018))	5.9645476
89766392	POLYGON ((-1.6318219 47.180651100000006, -1.6317245 47.1806068, -1.6316768000000001 47.1806532, -1.631771 47.180696700000006, -1.6318219 47.180651100000006))	5.0315332
89766677	POLYGON ((-1.6286652000000001 47.1796732, -1.6285788 47.1796829, -1.6285996 47.179773000000004, -1.6286868 47.1797642, -1.6286652000000001 47.1796732))	6.0841928
89766698	POLYGON ((-1.628122 47.179515800000004, -1.6281285 47.179544, -1.6282116000000002 47.179534800000006, -1.6282052 47.1795075, -1.628122 47.179515800000004))	5.058442
89766787	POLYGON ((-1.6233237 47.179748200000006, -1.6232907 47.179842300000004, -1.6233671 47.179849100000006, -1.6233669000000002 47.1797532, -1.6233237 47.179748200000006))	5.462378
89766915	POLYGON ((-1.6289574 47.1781208, -1.6289049000000002 47.1780732, -1.6287527000000002 47.178114300000004, -1.6288323 47.178186200000006, -1.6289574 47.1781208))	5.1754994
89766934	POLYGON ((-1.6227795 47.1785257, -1.6226901 47.178406900000006, -1.6225562 47.178453700000006, -1.6227012 47.178553, -1.6227795 47.1785257))	5.654642
89766941	POLYGON ((-1.6239890000000001 47.180123, -1.6239685000000001 47.180064200000004, -1.6238190000000001 47.180080100000005, -1.6238357 47.180142200000006, -1.6239890000000001 47.180123))	4.44959
89767212	POLYGON ((-1.6224007 47.179332900000006, -1.6223119000000001 47.1793421, -1.6225072 47.1795466, -1.6225663000000001 47.1795576, -1.6224007 47.179332900000006))	5.3320446
89767299	POLYGON ((-1.6324971000000001 47.1780525, -1.6322554 47.1780672, -1.6322964000000002 47.178130200000005, -1.6324971000000001 47.1780525))	4.4123025
89767346	POLYGON ((-1.6243651000000001 47.179051900000005, -1.6245548 47.1789879, -1.6245649000000002 47.1788335, -1.6242871 47.1788973, -1.6243651000000001 47.179051900000005))	4.800789
89767430	POLYGON ((-1.6322023 47.1803095, -1.6321035000000002 47.180370700000005, -1.6322985 47.180497300000006, -1.6323284 47.1804785, -1.6322023 47.1803095))	4.425412
89767439	POLYGON ((-1.6228222 47.178086400000005, -1.6228634000000002 47.1780684, -1.6228557000000001 47.1780539, -1.6228114 47.178064600000006, -1.6228222 47.178086400000005))	4.325606
89767440	POLYGON ((-1.6223518000000001 47.1784891, -1.6222728000000002 47.178518000000004, -1.6223168000000001 47.1786461, -1.6224385000000001 47.1785951, -1.6223518000000001 47.1784891))	4.5688405
89767543	POLYGON ((-1.6302569 47.1786612, -1.6303 47.178943700000005, -1.6304566 47.1786467, -1.6302569 47.1786612))	4.618469
89767688	POLYGON ((-1.6263688 47.180351800000004, -1.6263634 47.1802986, -1.6262645 47.1803029, -1.6262683 47.180356200000006, -1.6263688 47.180351800000004))	4.979831
89767711	POLYGON ((-1.6292154 47.181066400000006, -1.6292005 47.1811548, -1.6293694 47.1811663, -1.6293852000000002 47.1810794, -1.6292154 47.181066400000006))	6.096492
89767740	POLYGON ((-1.6261928 47.18023, -1.6261889 47.1801946, -1.6261344000000002 47.180198600000004, -1.6261382000000002 47.180232800000006, -1.6261928 47.18023))	6.014226
89767796	POLYGON ((-1.6321241000000002 47.1799948, -1.6321013000000002 47.179974, -1.6320714 47.1800201, -1.6321005000000002 47.1800294, -1.6321241000000002 47.1799948))	4.2219453

You can download DBeaver on this webpage.

Connect DBeaver to your database

1. Run DBeaver
2. Add a new connection
3. If you use a H2GIS type databse, please select `H2GIS embedded` *(use the search engine to filter)*
4. Point the database path by clicking on `Browse ....` By default the database is placed in the `NoiseModelling/data_dir` directory and is named `h2gisdb.mv.db`.
5. In the `Path` text area, remove `.mv.db` at the end of the address
6. If you want to open the database while NoiseModelling is running, add `;AUTO_SERVER=TRUE` at the end of the path (you should have something like this `/home/nm_user/NoiseModelling/NoiseModelling_4.0/data_dir/h2gisdb;AUTO_SERVER=TRUE`)
7. Click on `Terminate` to open your dabatase!


Créer un nouvelle connexion

Selectionner le type de nouvelle connexion

Créer une nouvelle connexion. Si vous ne voyez pas votre base de données dans la liste vous pouvez créer un nouveau pilote dans le gestionnaire.

h2

Sort by: Title [✓ Score]

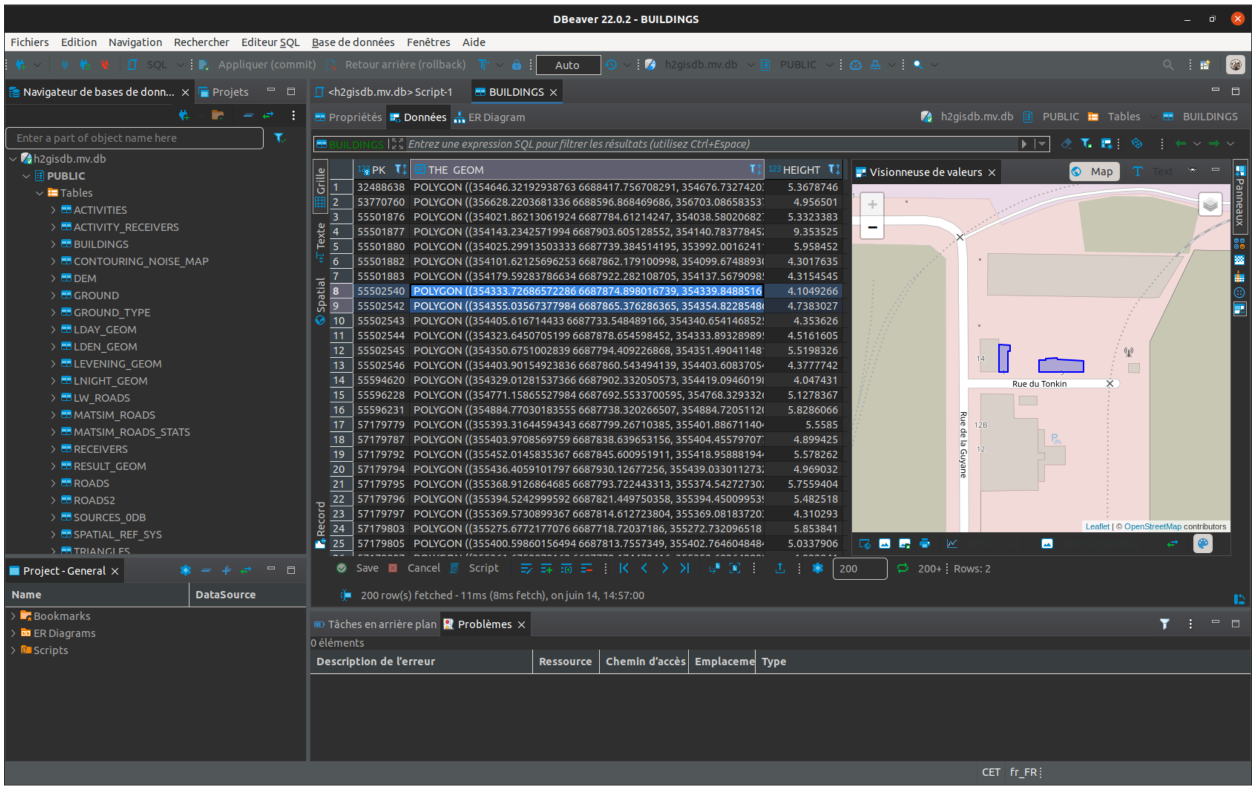
Warning: If you are using a version of DBeaver prior to 22.0.4, the interface may ask you to `Save` instead of `Opening` the existing db. Once you click on `Save`, a panel will warns you that `h2gisdb.mv.db` already exists and will ask you if you want to `Cancel` or `Replace` : click on `Replace`.

Now you can use the full potential of DBeaver and the H2GIS database. You can explore, display and manage your database.

5.24 Use NoiseModelling with a PostGIS database

5.24.1 Introduction

NoiseModelling is distributed with GeoServer. This application has been preconfigured to use H2GIS as the default database.



BUILDINGS | Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+Espace)

	PK	THE_GEOM	HEIGHT
1	32488638	POLYGON ((354646.32192938763 6688417.756708291, 354676.7327420	5.3678746
2	53770760	POLYGON ((356628.2203681336 6688596.868469686, 356703.0865835	4.956501
3	55501876	POLYGON ((354021.86213061924 6687784.61214247, 354038.5802068	5.3323383
4	55501877	POLYGON ((354143.2342571994 6687903.605128552, 354140.7837784	9.353525
5	55501880	POLYGON ((354025.29913503333 6687739.384514195, 353992.0016241	5.958452
6	55501882	POLYGON ((354101.62125696253 6687862.179100998, 354099.6748893	4.3017635
7	55501883	POLYGON ((354179.59283786634 6687922.282108705, 354137.5679098	4.3154545
8	55502540	POLYGON ((354333.72686572286 6687874.898016739, 354339.8488516	4.1049266
9	55502542	POLYGON ((354355.03567377984 6687865.376286365, 354354.8228548	4.7383027
10	55502543	POLYGON ((354405.616714433 6687733.548489166, 354340.654146852	4.353626
11	55502544	POLYGON ((354323.6450705199 6687878.654598452, 354333.8932898	4.5161605
12	55502545	POLYGON ((354350.6751002839 6687794.409226868, 354351.4904114	5.5198326
13	55502546	POLYGON ((354403.90154923836 6687860.543494139, 354403.6083705	4.3777742
14	55594620	POLYGON ((354329.01281537366 6687902.332050573, 354419.0946019	4.047431
15	55596228	POLYGON ((354771.15865527984 6687692.5533700595, 354768.329332	5.1278367
16	55596231	POLYGON ((354884.77030183555 6687738.320266507, 354884.7205112	5.8286066
17	57179779	POLYGON ((355393.31644594343 6687799.26710385, 355401.8867114	5.5585
18	57179787	POLYGON ((355403.9708569759 6687838.639653156, 355404.4557970	4.899425
19	57179792	POLYGON ((355452.0145835367 6687845.600951911, 355418.9588819	5.578262
20	57179794	POLYGON ((355436.4059101797 6687930.12677256, 355439.0330112	4.969032
21	57179795	POLYGON ((355368.9126864685 6687793.722443313, 355374.5427273	5.7559404
22	57179796	POLYGON ((355394.5242999592 6687821.449750358, 355394.4500995	5.482518
23	57179797	POLYGON ((355369.5730899367 6687814.612723804, 355369.0818372	4.310293
24	57179803	POLYGON ((355275.6772177076 6687718.72037186, 355272.732096518	5.853841
25	57179805	POLYGON ((355400.59860156494 6687813.7557349, 355402.764604848	5.0337906

200 200+ Rows: 2

200 row(s) fetched - 11ms (8ms fetch), on juin 14, 14:57:00

0 éléments

Description de l'erreur	Ressource	Chemin d'accès	Emplaceme	Type

CET fr_FR

H2GIS does not need to be configured or installed on the system and is therefore perfectly suitable as a default database.

However, you may want to connect NoiseModelling to a PostgreSQL/PostGIS database (this option may be interesting especially if you are using huge datasets (*e.g* on large area)).

That is why NoiseModelling has been written with the idea of maintaining the H2GIS/PostGIS compatibility.

This tutorial will not cover the steps for installing and configuring a PostGIS database.

5.24.2 Connect with Java

First you have to add some libraries. We will use PostgreSQL/PostGIS wrapper available in the H2GIS library:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
↪XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.
↪org/maven-v4_0_0.xsd">
    <properties>
        <h2gis-version>2.1.0-SNAPSHOT<</h2gis-version>
        <noisemodelling-version>4.0.0</noisemodelling-version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.12</version>
        </dependency>
```

(continues on next page)

(continued from previous page)

```
        <dependency>
            <groupId>org.orbisgis</groupId>
            <artifactId>noisemodelling-emission</artifactId>
            <version>${noisemodelling-version}</version>
        </dependency>
        <dependency>
            <groupId>org.orbisgis</groupId>
            <artifactId>noisemodelling-propagation</artifactId>
            <version>${noisemodelling-version}</version>
        </dependency>
        <dependency>
            <groupId>org.orbisgis</groupId>
            <artifactId>h2gis</artifactId>
            <version>${h2gis-version}</version>
        </dependency>
        <dependency>
            <groupId>org.orbisgis</groupId>
            <artifactId>h2gis-api</artifactId>
            <version>${h2gis-version}</version>
        </dependency>
        <dependency>
            <groupId>org.orbisgis</groupId>
            <artifactId>h2gis-utilities</artifactId>
            <version>${h2gis-version}</version>
        </dependency>
        <dependency>
            <groupId>org.orbisgis</groupId>
            <artifactId>postgis-jts-osgi</artifactId>
            <version>${h2gis-version}</version>
        </dependency>
    </dependencies>
</project>
```

The new dependency here is `postgis-jts-osgi`. It contains some code to convert PostGIS geometries objects into/from JTS objects.

In your code you have to import the PostGIS wrapper class and some utility class:

```
import org.h2gis.functions.io.geojson.GeoJsonRead;
import org.h2gis.postgis_jts_osgi.DataSourceFactoryImpl;

import java.net.ConnectException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.HashSet;
import java.util.Locale;
```

Then use it to connect to you local or remote PostGIS database and obtain a valid JDBC connection object:

```
    public static void main() throws Exception {
        DataSourceFactoryImpl dataSourceFactory = new DataSourceFactoryImpl();
        Properties p = new Properties();
        p.setProperty("serverName", "localhost");
        p.setProperty("portNumber", "5432");
        p.setProperty("databaseName", "postgres");
```

(continues on next page)

(continued from previous page)

```
        p.setProperty("user", "postgres");
        p.setProperty("password", "");
        try(Connection connection = SFSUtilities.wrapConnection(dataSourceFactory.
↪createDataSource(p).getConnection())) {
            Statement sql = connection.createStatement();
```

Finally you can use the NoiseModelling functions as usual:

```
package org.noise_planet.nmtutorial01;

import org.h2gis.api.EmptyProgressVisitor;
import org.h2gis.api.ProgressVisitor;
import org.h2gis.functions.io.csv.CSVDriverFunction;
import org.h2gis.functions.io.geojson.GeoJsonRead;
import org.h2gis.postgis_jts_osgi.DataSourceFactoryImpl;
import org.h2gis.utilities.SFSUtilities;
import org.junit.Test;
import org.noise_planet.noisemodelling.emission.jdbc.LDENConfig;
import org.noise_planet.noisemodelling.emission.jdbc.LDENPointNoiseMapFactory;
import org.noise_planet.noisemodelling.propagation.ComputeRaysOut;
import org.noise_planet.noisemodelling.propagation.IComputeRaysOut;
import org.noise_planet.noisemodelling.propagation.RootProgressVisitor;
import org.noise_planet.noisemodelling.propagation.jdbc.PointNoiseMap;
import org.postgresql.util.PSQLException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.net.ConnectException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.HashSet;
import java.util.Locale;
import java.util.Properties;
import java.util.Set;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

public class Main {
    static Logger LOGGER = LoggerFactory.getLogger(Main.class);

    public static void main() throws Exception {
        DataSourceFactoryImpl dataSourceFactory = new DataSourceFactoryImpl();
        Properties p = new Properties();
        p.setProperty("serverName", "localhost");
        p.setProperty("portNumber", "5432");
        p.setProperty("databaseName", "postgres");
        p.setProperty("user", "postgres");
        p.setProperty("password", "");
        try(Connection connection = SFSUtilities.wrapConnection(dataSourceFactory.
↪createDataSource(p).getConnection())) {
            Statement sql = connection.createStatement();

            // Clean DB

```

(continues on next page)

(continued from previous page)

```
            sql.execute("DROP TABLE IF EXISTS BUILDINGS");
            sql.execute("DROP TABLE IF EXISTS LW_ROADS");
            sql.execute("DROP TABLE IF EXISTS RECEIVERS");
            sql.execute("DROP TABLE IF EXISTS DEM");

            // Import BUILDINGS

            LOGGER.info("Import buildings");

            GeoJsonRead.readGeoJson(connection, Main.class.getResource("buildings.
↪geojson").getFile(), "BUILDINGS");

            // Import noise source

            LOGGER.info("Import noise source");

            GeoJsonRead.readGeoJson(connection, Main.class.getResource("lw_roads.
↪geojson").getFile(), "lw_roads");
            // Set primary key
            sql.execute("ALTER TABLE lw_roads ADD CONSTRAINT lw_roads_pk PRIMARY KEY␣
↪(\"PK\");");

            // Import BUILDINGS

            LOGGER.info("Import evaluation coordinates");

            GeoJsonRead.readGeoJson(connection, Main.class.getResource("receivers.
↪geojson").getFile(), "receivers");
            // Set primary key
            sql.execute("ALTER TABLE receivers ADD CONSTRAINT RECEIVERS_pk PRIMARY␣
↪KEY (\"PK\");");

            // Import MNT

            LOGGER.info("Import digital elevation model");

            GeoJsonRead.readGeoJson(connection, Main.class.getResource("dem_lorient.
↪geojson").getFile(), "dem");

            // Init NoiseModelling
            PointNoiseMap pointNoiseMap = new PointNoiseMap("buildings", "lw_roads",
↪"receivers");

            pointNoiseMap.setMaximumPropagationDistance(160.0d);
            pointNoiseMap.setSoundReflectionOrder(0);
            pointNoiseMap.setComputeHorizontalDiffraction(true);
            pointNoiseMap.setComputeVerticalDiffraction(true);
            // Building height field name
            pointNoiseMap.setHeightField("HEIGHT");
            // Point cloud height above sea level POINT(X Y Z)
            pointNoiseMap.setDemTable("DEM");
            // Do not propagate for low emission or far away sources.
            // error in dB
            pointNoiseMap.setMaximumError(0.1d);

            // Init custom input in order to compute more than just attenuation
            // LW_ROADS contain Day Evening Night emission spectrum
```

(continues on next page)

(continued from previous page)

```
            LDENConfig ldenConfig = new LDENConfig(LDENConfig.INPUT_MODE.INPUT_MODE_
↪LW_DEN);

            ldenConfig.setComputeLDay(true);
            ldenConfig.setComputeLEvening(true);
            ldenConfig.setComputeLNight(true);
            ldenConfig.setComputeLDEN(true);

            LDENPointNoiseMapFactory tableWriter = new␣
↪LDENPointNoiseMapFactory(connection, ldenConfig);

            tableWriter.setKeepRays(true);

            pointNoiseMap.setPropagationProcessDataFactory(tableWriter);
            pointNoiseMap.setComputeRaysOutFactory(tableWriter);

            RootProgressVisitor progressLogger = new RootProgressVisitor(1, true, 1);

            pointNoiseMap.initialize(connection, new EmptyProgressVisitor());

            // force the creation of a 2x2 cells
            pointNoiseMap.setGridDim(2);


            // Set of already processed receivers
            Set<Long> receivers = new HashSet<>();
            ProgressVisitor progressVisitor = progressLogger.subProcess(pointNoiseMap.
↪getGridDim()*pointNoiseMap.getGridDim());
            LOGGER.info("start");
            long start = System.currentTimeMillis();

            // Iterate over computation areas
            try {
                tableWriter.start();
                for (int i = 0; i < pointNoiseMap.getGridDim(); i++) {
                    for (int j = 0; j < pointNoiseMap.getGridDim(); j++) {
                        // Run ray propagation
                        IComputeRaysOut out = pointNoiseMap.evaluateCell(connection,␣
↪i, j, progressVisitor, receivers);
                    }
                }
            } finally {
                tableWriter.stop();
            }
            long computationTime = System.currentTimeMillis() - start;
            logger.info(String.format(Locale.ROOT, "Computed in %d ms, %.2f ms per␣
↪receiver",
                    computationTime,computationTime / (double)receivers.size()));
            // Export result tables as csv files
            CSVDriverFunction csv = new CSVDriverFunction();
            csv.exportTable(connection, ldenConfig.getlDayTable(), new␣
↪File(ldenConfig.getlDayTable()+".csv"), new EmptyProgressVisitor());
            csv.exportTable(connection, ldenConfig.getlEveningTable(), new␣
↪File(ldenConfig.getlEveningTable()+".csv"), new EmptyProgressVisitor());
            csv.exportTable(connection, ldenConfig.getlNightTable(), new␣
↪File(ldenConfig.getlNightTable()+".csv"), new EmptyProgressVisitor());
            csv.exportTable(connection, ldenConfig.getlDenTable(), new␣
↪File(ldenConfig.getlDenTable()+".csv"), new EmptyProgressVisitor());
```

(continues on next page)

(continued from previous page)

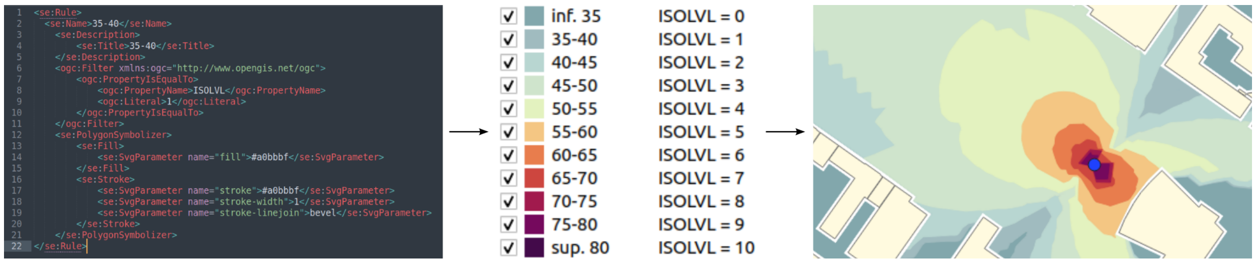
```
        } catch (PSQLException ex) {
            if (ex.getCause() instanceof ConnectException) {
                // Connection issue ignore
                LOGGER.warn("Connection error to local PostGIS, ignored", ex);
            } else {
                throw ex;
            }
        } catch (SQLException ex) {
            LOGGER.error(ex.getLocalizedMessage(), ex.getNextException());
            throw ex;
        }
    }
}
```

5.25 Get Started

1. If not already done, create an account on Github,
2. Go to the official NoiseModelling repository: https://github.com/Universite-Gustave-Eiffel/NoiseModelling
3. There are 4 main librairies:
   - `noisemodelling-emission` : to determine the noise emission
   - `noisemodelling-jdbc` : to connect NoiseModelling to a database
   - `noisemodelling-pathfinder` : to determine the noise path
   - `noisemodelling-propagation` : to calculate the noise propagation
4. Enjoy & feel free to contact us!

5.26 Noise Map Color Scheme

Below are presented some color schemes used to colorize noise isophones, produced in the `CONTOURING_NOISE_MAP` layer.



```
<se:Rule>
  <se:Name>35-40</se:Name>
    <se:Description>
        <se:Title>35-40</se:Title>
    </se:Description>
    <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
        <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>ISOLVL</ogc:PropertyName>
            <ogc:Literal>1</ogc:Literal>
        </ogc:PropertyIsEqualTo>
    </ogc:Filter>
    <se:PolygonSymbolizer>
        <se:Fill>
            <se:SvgParameter name="fill">#a0bbbf</se:SvgParameter>
        </se:Fill>
        <se:Stroke>
            <se:SvgParameter name="stroke">#a0bbbf</se:SvgParameter>
            <se:SvgParameter name="stroke-width">1</se:SvgParameter>
            <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
        </se:Stroke>
    </se:PolygonSymbolizer>
</se:Rule>
```

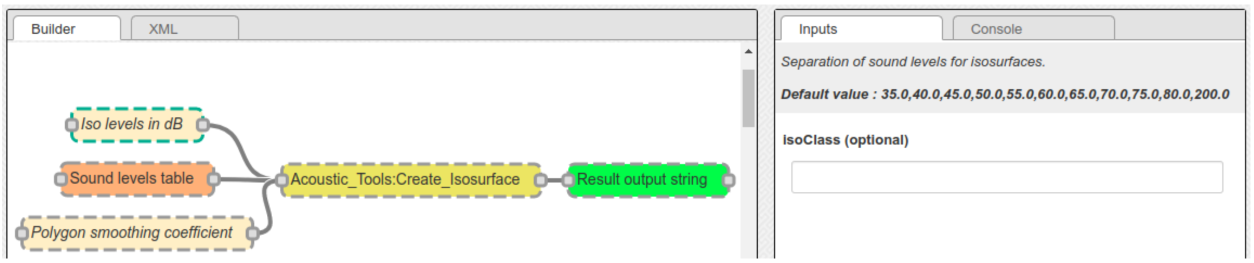
✓	■	inf. 35	ISOLVL = 0
✓	■	35-40	ISOLVL = 1
✓	■	40-45	ISOLVL = 2
✓	■	45-50	ISOLVL = 3
✓	■	50-55	ISOLVL = 4
✓	■	55-60	ISOLVL = 5
✓	■	60-65	ISOLVL = 6
✓	■	65-70	ISOLVL = 7
✓	■	70-75	ISOLVL = 8
✓	■	75-80	ISOLVL = 9
✓	■	sup. 80	ISOLVL = 10

Note: If you want to feed this list with other schemes, please contact us (see *Support* page).

5.26.1 Introduction

Creation of the Isosurfaces

NoiseModelling can produce isophones (also called isosurfaces) thanks to the `Acoustic_Tools:Create_Isosurface` script. In this script, an optionnal parameter called `Iso levels in dB` allows the user to specify the thresholds used to generate the surfaces.



Separation of sound levels for isosurfaces.

Default value : 35.0,40.0,45.0,50.0,55.0,60.0,65.0,70.0,75.0,80.0,200.0

By default, the thresholds are 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 200. In the resulting `CONTOURING_NOISE_MAP` layer, these values are then converted into integer and stored in the `ISOLVL` column. The first threshold is equal to `0`. The second one is equal to `1` *… (See table below).*

Table 3: Correspondence between default thresholds and `ISOLVL` values

Threshold	ISOLVL	ISOLABEL
35	0	<35
40	1	35-40
45	2	40-45
50	3	45-50
55	4	50-55
60	5	55-60
65	6	60-65
70	7	65-70
75	8	70-75
80	9	75-80
200	10	>80

Warning: So the `ISOLVL` values directly depends on the `Iso levels in dB` thresholds. When applying a style (see below), you must check that this parameter feets with the classes defined in the `.sld` file.

SLD file

For each of the color schemes presented below, a cartographic style, following the "Style Layer Descriptor" formalism, is provided as an `.sld` file. This `.sld` file can be loaded in many GIS applications, such as QGIS. The classification is made on the `ISOLVL` column, in the `CONTOURING_NOISE_MAP` table.

Note: For those who are new to GIS and want to get started with QGIS, we advise you to follow this tutorial as a start.

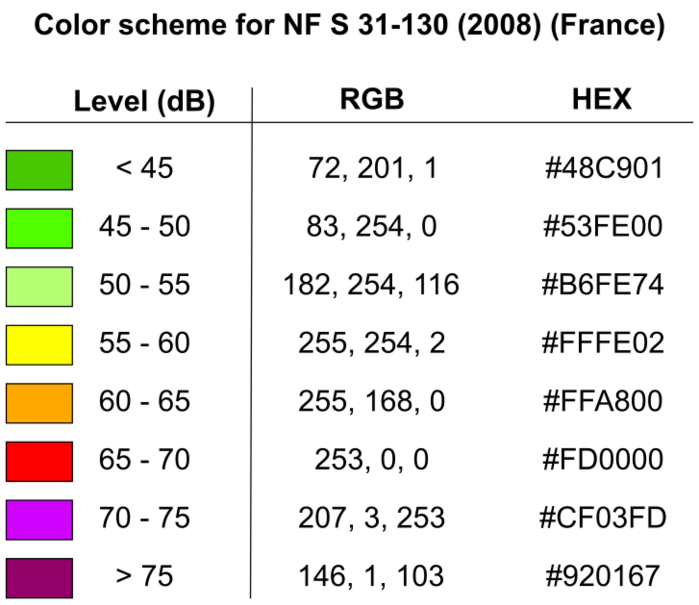
To know how to load an `.sld` file, you can also consult the NoiseModelling tutorial *Noise Map from Point Source - GUI* in the section "`Step 3 - Apply a color palette adapted to acoustics`"

5.26.2 French NF S31-130

The "NF S31-130" is the standard currently in force in France.

- **French title**: "Acoustique Cartographie du bruit en milieu extérieur Élaboration des cartes et représentation graphique"
- **English title**: "Acoustics - Cartography of outside environment noise - Drawing up of maps and graphical representation"
- **Last update**: 2008

Color scheme



Color scheme for NF S 31-130 (2008) (France)

	Level (dB)	RGB	HEX
🟩	< 45	72, 201, 1	#48C901
🟩	45 - 50	83, 254, 0	#53FE00
🟩	50 - 55	182, 254, 116	#B6FE74
🟨	55 - 60	255, 254, 2	#FFFE02
🟧	60 - 65	255, 168, 0	#FFA800
🟥	65 - 70	253, 0, 0	#FD0000
🟪	70 - 75	207, 3, 253	#CF03FD
🟪	> 75	146, 1, 103	#920167

SLD file

The SLD representation of this color scheme is available here : Style NF S31-130

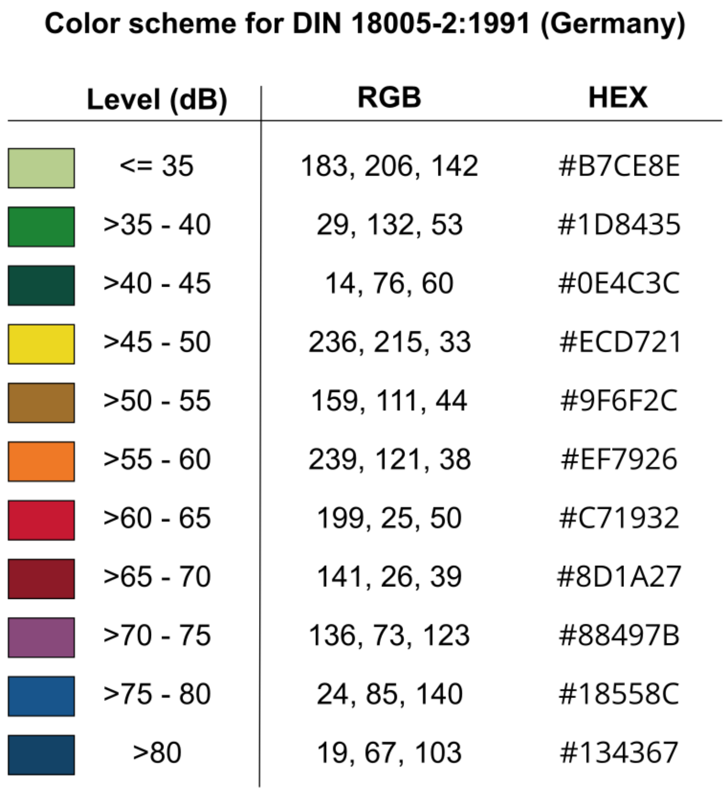
Warning: This style will work only if you specified `Iso levels in dB` = `45, 50, 55, 60, 65, 70, 75, 200` when exectuting the `Acoustic_Tools:Create_Isosurface` script

5.26.3 German DIN 18005-2:1991

The "DIN 18005-2:1991" is the standard currently in force in Germany.

- **German title**: "Schallschutz im Städtebau; Lärmkarten; Kartenmäßige Darstellung von Schallimmissionen"
- **English title**: "Noise abatement in town planning; noise maps; graphical representation of noise pollution"
- **Last update**: 1991

Color scheme



Color scheme for DIN 18005-2:1991 (Germany)

	Level (dB)	RGB	HEX
🟩	<= 35	183, 206, 142	#B7CE8E
🟩	>35 - 40	29, 132, 53	#1D8435
🟩	>40 - 45	14, 76, 60	#0E4C3C
🟨	>45 - 50	236, 215, 33	#ECD721
🟫	>50 - 55	159, 111, 44	#9F6F2C
🟧	>55 - 60	239, 121, 38	#EF7926
🟥	>60 - 65	199, 25, 50	#C71932
🟥	>65 - 70	141, 26, 39	#8D1A27
🟪	>70 - 75	136, 73, 123	#88497B
🟦	>75 - 80	24, 85, 140	#18558C
🟦	>80	19, 67, 103	#134367

SLD file

The SLD representation of this color scheme is available here : Style DIN 18005-2:1991

Warning: This style will work only if you specified `Iso levels in dB` = `35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 200` when exectuting the `Acoustic_Tools:Create_Isosurface` script

5.26.4 Italian Normativa tecnica UNI 9884

The "Normativa tecnica UNI 9884" is a standard currently used in Italy.

- **Italian title**: "Acustica. Caratterizzazione acustica del territorio mediante la descrizione del rumore ambientale"
- **English title**: "Acoustics. Acoustic characterisation of the territory through the description of environmental noise"
- **Last update**: 1991

Color scheme

Table 4: Norma UNI 9884 - Convenzioni per la rappresentazione delle mappe di rumore

Zone di rumore dB(A)	Colore
Sotto 35	Verde chiaro
Da 35 a 40	Verde
Da 40 a 45	Verde scuro
Da 45 a 50	Giallo
Da 50 a 55	Ocra
Da 55 a 60	Arancione
Da 60 a 65	Vermiglio
Da 65 a 70	Carminio
Da 70 a 75	Rosso violetto
Da 75 a 80	Blu
Sopra 80	Blu scuro

We can see that the thresholds and colors defined in the table above are the same values as the ones defined in “German DIN 18005-2:1991”.

SLD file

Since this norm is almost the same as “German DIN 18005-2:1991”, you are invited to use the German SLD file, available here : Style DIN 18005-2:1991

Warning: This style will work only if you specified `Iso levels in dB` = `35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 200` when exectuting the `Acoustic_Tools:Create_Isosurface` script

5.26.5 Coloring Noise

The “Coloring Noise” scheme is a proposition made by Beate Tomio, within her PhD.

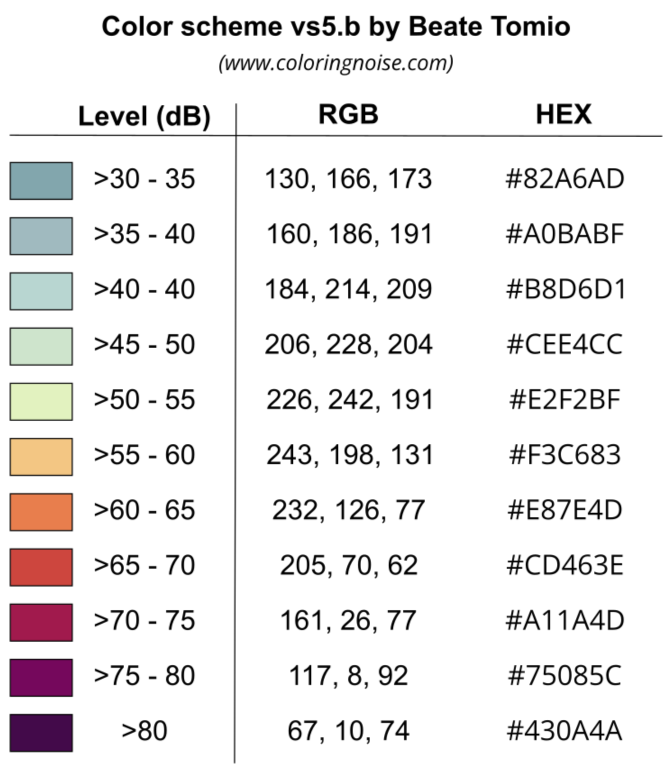
- **English title**: Coloring Noise - A color scheme for visualizing noise immission in maps
- **Description**: The creation process of this color scheme is presented on Beate’s website
- **Last update**: 2016

Color scheme

SLD file

The SLD representation of this color scheme is available here : Style Coloring Noise

Warning: This style will work only if you specified `Iso levels in dB` = `35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 200` when exectuting the `Acoustic_Tools:Create_Isosurface` script



Color scheme vs5.b by Beate Tomio

(www.coloringnoise.com)

	Level (dB)	RGB	HEX
■	>30 - 35	130, 166, 173	#82A6AD
■	>35 - 40	160, 186, 191	#A0BABF
■	>40 - 40	184, 214, 209	#B8D6D1
■	>45 - 50	206, 228, 204	#CEE4CC
■	>50 - 55	226, 242, 191	#E2F2BF
■	>55 - 60	243, 198, 131	#F3C683
■	>60 - 65	232, 126, 77	#E87E4D
■	>65 - 70	205, 70, 62	#CD463E
■	>70 - 75	161, 26, 77	#A11A4D
■	>75 - 80	117, 8, 92	#75085C
■	>80	67, 10, 74	#430A4A

5.26.6 Create your own .SLD file

The `.sld` is an `.xml` file that may be opened and edited in most of the text editor. So you can easily modify existing `.sld` files to feet with your needs.

SLD structure

An `.sld` file is made of `rules` (`<se:Rule>`). A rule has a name (`<se:Name>`), a description (`<se:Title>`) and is applied on some specific values (Filter) and for one symbol.

Filter

The rule is applied:

- thanks to an operator that indicates how to filter the table values. In the example below `PropertyIsEqualTo` indicates that an equality test will be made to select values. If the value in the column match with the one defined in the rule, the object (geometry) will be selected to apply the rule.
- on a specific column : `<ogc:PropertyName>`. In the example below, `ISOLVL`. If the column does not exist in the table or if the name is not written exactly in the same way, your rule will not work.
- for a specific value : `<ogc:Literal>`. In the example below, `1`. So for each objetcs that have `1` in the column `ISOLVL` the rule will be applied

Symbol

For one rule, we can define how the symbol will be displayed. In our case, the symbol is a polygon (the isosufrce). In the SLD langage, a polygon is called a `PolygonSymbolizer`. This object has two main caracteristics:

- **The fill** `[<se:Fill>]`
  - a color, exprimed with an hexadecimal code. In the example below, #a0bbbf
- **The stroke** `[<se:Stroke>]`
  - a color, exprimed with an hexadecimal code. In the example below, #a0bbbf *(we choosed to have the same color for fill and stroke for esthetic purpose, but you can change it)*
  - a width (`stroke-width`). In the example below, 1
  - a `stroke-linejoin` option that defines how two segments may join. `bevel` is the default option

Below is an extraction from an `.sld` file that illustrates all these points seen before.

```
<se:Rule>
  <se:Name>35-40</se:Name>
    <se:Description>
        <se:Title>35-40</se:Title>
    </se:Description>
    <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>ISOLVL</ogc:PropertyName>
          <ogc:Literal>1</ogc:Literal>
        </ogc:PropertyIsEqualTo>
    </ogc:Filter>
    <se:PolygonSymbolizer>
        <se:Fill>
          <se:SvgParameter name="fill">#a0bbbf</se:SvgParameter>
        </se:Fill>
        <se:Stroke>
          <se:SvgParameter name="stroke">#a0bbbf</se:SvgParameter>
          <se:SvgParameter name="stroke-width">1</se:SvgParameter>
          <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
        </se:Stroke>
    </se:PolygonSymbolizer>
</se:Rule>
```

5.27 Support

If you are having trouble with NoiseModelling, you can contact the NoiseModelling team through the following channels:

1. open an issue on Github : https://github.com/Universite-Gustave-Eiffel/NoiseModelling/issues
2. write a message on Github : https://github.com/Universite-Gustave-Eiffel/NoiseModelling/discussions
3. send us an email at contact@noise-planet.org

We warmly encourage you to choose options 1 or 2 because they have the merit of being public and can therefore benefit the community.

If you have more "private" issue, or if you don't have any knowledge with Github, emails are welcome !

5.28 License

NoiseModelling and its documentation are distributed under GPL v3 license and are jointly developed by the *Joint Research Unit in Environmental Acoustics* (UMRAE, Université Gustave Eiffel - Cerema) and the DECIDE team from the Lab-STICC (CNRS).

5.29 Glossary

Below are defined some terms or acronyms used in this documentation:

- LAeq : A-weighted Leq sound level.
- Lday : LAeq during the day (6h-18h)
- Lden : LAeq over a whole day (day-evening-night)
- Leq : equivalent continuous sound level (in *dB*) having the same total sound energy as the fluctuating level measured
- Levening : LAeq during the evening (18h-22h)
- Lnight : LAeq during the night (22h-6h)

CHAPTER 6

Indices and tables

- genindex
- modindex
- search