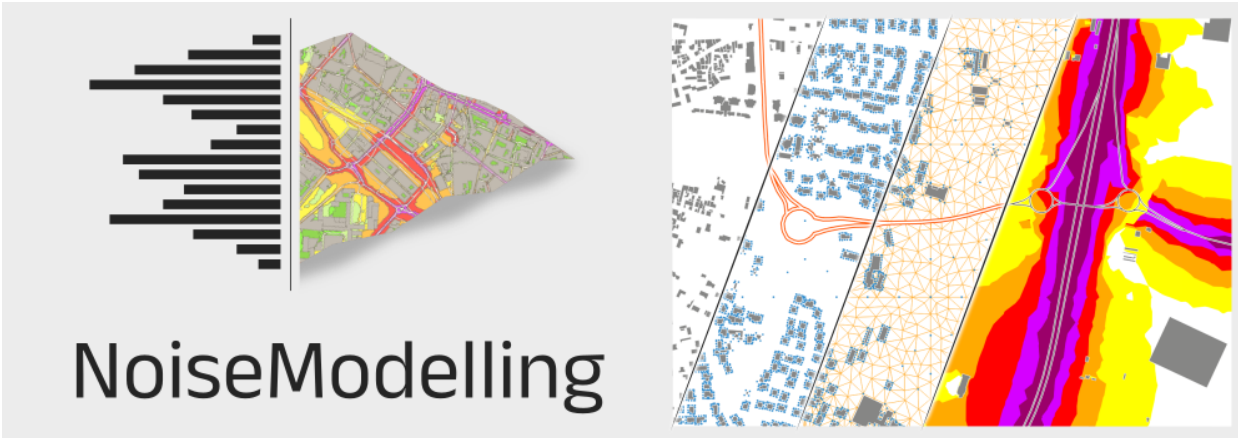
NoiseModelling Documentation

Release 6.0.2-SNAPSHOT

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

Jul 22, 2026

NOISEMODELLING PRESENTATION



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

NoiseModelling is a free and open-source Java library for producing environmental noise maps from local to national scales, implementing CNOSSOS-EU road and rail noise emission and propagation methods while linking to H2GIS and PostGIS for spatial analysis. It responds to the need for robust noise assessment in public health and environmental planning by enabling simulation and prediction of noise propagation for mitigation design and compliance with EU regulations. The software can be used independently or through a graphical interface and is openly available to the research, education, and professional communities. It has been widely used for strategic noise mapping, dynamic maps driven by traffic models or sensors, sensitivity studies, and investigations of particular sources such as emergency sirens and drones.

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
- for **more information** for the final results with the reference results in ISO/TR 17534-4: 2020 follow the "*Conformity to ISO 17534-1:2015*" 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

ONE

PACKAGING

The latest release page offers three NoiseModelling packages:

- `NoiseModelling_X.X.X.zip`: A cross-platform version for Windows, Linux, and macOS. It includes the web GUI and a command-line interface.
- `NoiseModelling_X.X.X_install.exe`: A standalone Windows installer that includes the web GUI and a bundled Java Virtual Machine.
- `NoiseModelling-X.X.X.dmg`: A standalone Mac OS installer that includes the web GUI and a bundled Java Virtual Machine.

Please check the *Installation guide* before installing, and refer to *NoiseModelling client line interface (CLI)* for CLI usage.

> **Warning:** The Windows and Mac OS installer have not been signed yet. So you may have a security warning when installing and you are invited to follow additional steps to bypass this issue.

In addition, a Docker image is provided in the packages page.

CHAPTER TWO

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

THREE

LICENCE

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

CHAPTER

FOUR

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

FIVE

FUNDINGS

Research projects:

- OPTImisation technique et environnementale des moyens de lutte contre le gel en viticulture par Tours-Anti-Gel en Centre-Val de Loire (OptiTAG), funded by Région Centre Val de Loire & co-funded by European Union 2024-2027
- AMELIA, funded by the call "DIAT (Démonstrateurs d'IA frugale au service de la transition écologique) de la Banque des Territoires" 2024-2027
- ANR SYMEXPO (ANR-21-CE22-0022-01) 2021-2026
- Sampols 2.0: a disruptive solution for noise pollution monitoring (SIREN), funded by Eurostars 3 - BPI 2024-2025
- Nature4cities (N4C) project, funded by European Union's Horizon 2020 research and innovation programme under grant agreement N°730468
- ANR CENSE (ANR-16-CE22-0012) 2017-2021
- ANR VegDUD (ANR-09-VILL-0007) 2009-2014
- ANR Eval-PDU (ANR-08-VILL-0005) 2008-2011

Institutional (public) fundings:

- DGPR 2020-2022 & 2025-27
- Université Gustave Eiffel (formerly Ifsttar, formerly LCPC), CNRS, Cerema, Université Bretagne Sud, Ecole Centrale de Nantes

Private fundings:

- Airbus Urban Mobility

Warning:

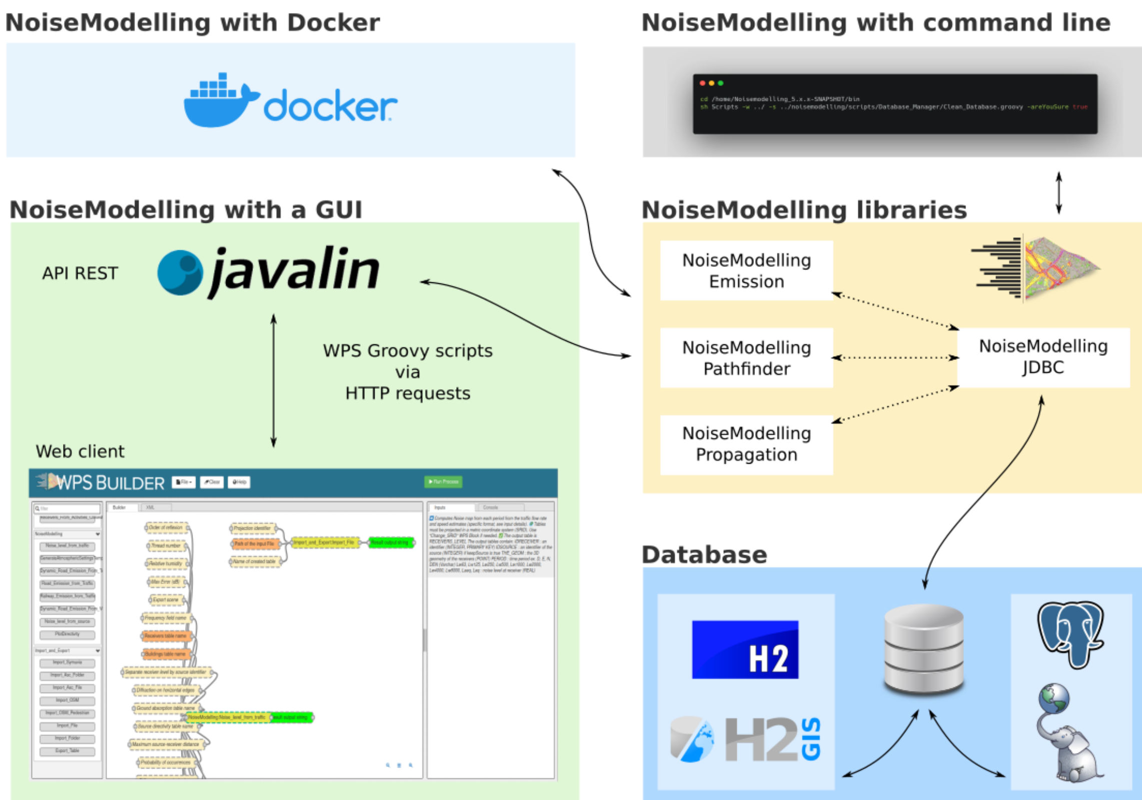
- The official documentation is available in English only
- Some illustrations may refer to previous versions of NoiseModelling
- If you observe some mistakes or errors, please open an issue here or contact us at contact@noise-planet.org
- You are also welcome to contribute to the documentation (click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



5.1.1 1. NoiseModelling libraries

NoiseModelling is made of 5 main libraries:

- `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
- `noisemodelling-scripts` : Groovy scripts (Blocks) and web server

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 two database (free, open-source and powerful) couples:

- H2 / H2GIS, which is configured and embedded by default. In this case, the user has nothing to do.
- PostGreSQL / PostGIS. In this case, the user has to configure the connection (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 and the web page is named "*Builder*".

In order for the Builder to communicate with the NoiseModelling libraries, we use a standard named Web Processing Service (defined by the Open Geospatial Consortium OGC) API to execute scripts. These scripts are written in the Groovy language and are called "Blocks".

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 .groovy script (Block) you want, with the needed arguments

Note: The `.groovy` script (Block) 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) (Blocks)).

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

Examples can be found in the page "*NoiseModelling client line interface (CLI)*".

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 are published on our Github repository. It is the best way to safely host NoiseModelling on a public server. Be aware that a registered user may be able to run a privilege escalation attack through the usage of scripts/SQL, so you should add only trusted users.

On the root of this repository you can find an example docker compose.

You can edit the following environment variables:

- PROXY_BASE_URL : If you have a domain name you can use the your domain name instead of localhost
- ROOT_URL : By default the service is accessible from the path /noisemodelling but you can change it by using the environment variable ROOT_URL (empty to use the base url)
- UNSECURE_MODE : By default the registration is enabled (with TOTP). If you use this docker image locally, you can disable the registration by setting the environment variable UNSECURE_MODE in the docker-compose.yml file to true.

Dependencies

Install Docker or Podman on your system

Running

Download the file docker-compose.yml and run this command in the same folder:

```
docker compose up -d
```

or

```
podman compose up -d
```

Follow the instructions of the logs in order to register the administrator account (if not in unsecure mode).

```
docker compose logs noisemodelling
```

or

```
podman compose logs noisemodelling
```

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

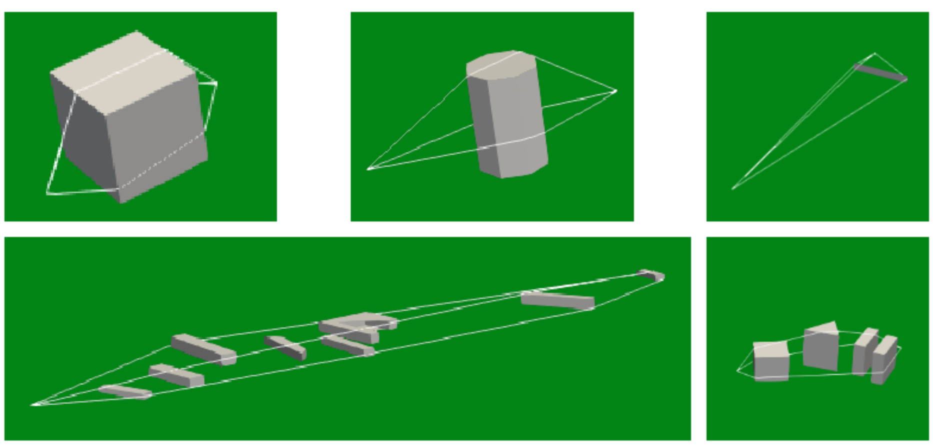
User can also add directly its own emission sound power level (LW).

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 Installation guide

5.3.1 Step 1 : Requirements

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

NoiseModelling requires Java >= 25. Any version of Java 25 or later is supported.

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 ``start_windows.bat` file (in the `NoiseModelling_xxx.zip` release file). In this case, Java >= 25 has to be installed before.

Download and install Java: choose between OpenJDK or Oracle versions.

Linux or Mac

If not already done, you have to install the Java version >= 25.

1. Check whether Java is already installed:

   ```
   java -version
   ```

   The command should print a version starting with `25`. Otherwise, install Java first.

2. Download and install Java: choose between OpenJDK or Oracle versions.

3. Find the installation path to use for `JAVA_HOME`.

   On Linux:

   ```
   readlink -f "$(command -v java)"
   ```

   This prints a path ending with `/bin/java` (for example `/usr/lib/jvm/java-25-openjdk-amd64/bin/java`); `JAVA_HOME` is the parent directory of `bin` (here `/usr/lib/jvm/java-25-openjdk-amd64`).

   On macOS:

   ```
   /usr/libexec/java_home -v latest
   ```

   This prints the directory that must be used as `JAVA_HOME` for Java.

4. Set the `JAVA_HOME` environment variable and update your `PATH` (adapt the path with the one found above):

   ```
   export JAVA_HOME=/usr/lib/jvm/java-25-openjdk-amd64
   export PATH="$JAVA_HOME/bin:$PATH"
   ```

   On macOS you can also use:

```
export JAVA_HOME=$(/usr/libexec/java_home -v latest)
export PATH="$JAVA_HOME/bin:$PATH"
```

5. Verify that `JAVA_HOME` is correctly set:

```
echo $JAVA_HOME
```

You should get the Java directory (for example `/usr/lib/jvm/java-25-openjdk-amd64`). If this is not the case, you are invited to follow the steps proposed here.

5.3.2 Step 2: Download NoiseModelling

Download the latest release of NoiseModelling on Github.

- Windows: you can directly download and run the `NoiseModelling_*.exe` installer file *(or you can also follow the Linux instructions below if you have a working JRE, see previous step)*
- MacOS: you can directly download and run the `NoiseModelling_*.dmg` installer file *(or you can also follow the Linux instructions below if you have a working JRE, see previous step)*
- Linux : download the `NoiseModelling_6.x.x.zip` file and unzip it into a chosen directory

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

Warning: For **Linux** and **Mac** users, please make sure your Java environment is correctly set up (see previous step). **Windows** users who are using the `.exe` file are not concerned, since the Java Runtime Environment is **already embedded**.

5.3.3 Step 3: Start NoiseModelling GUI

As described on the page "*Architecture*", NoiseModelling can be used through a Graphical User Interface (GUI) in a web browser.

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

These tools (WPS Builder and H2GIS) are already included in the archive, so you don't have to install them separately.

To launch NoiseModelling with the GUI, start it from a command prompt (terminal). This will start a local server on your computer, which provides the GUI as a web application.

Please execute:

- Windows: `NoiseModelling.exe` or `NoiseModelling_xxx\start_windows.bat`
- Linux or Mac: `NoiseModelling_xxx/start_linux_macos.sh` *(make sure the file is allowed to be executed before running it)*

Tip: NoiseModelling will stay open as long as the command window is open. If you close it, NoiseModelling will automatically stop and the GUI will no longer be available.

5.3.4 Step 4: Open NoiseModelling GUI

The NoiseModelling GUI is built using the WPS_Builder component and runs as a web application provided by the local server started in Step 3.

By running NoiseModelling your default web browser should have been opened to the http://localhost:8000 address. If not please go to this URL, if something went wrong you should have more information on your terminal.

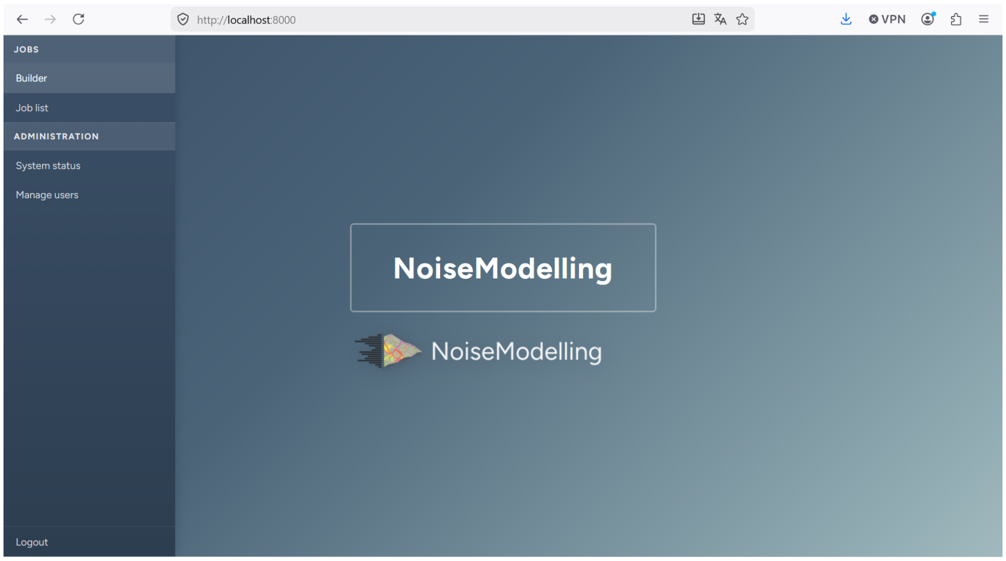
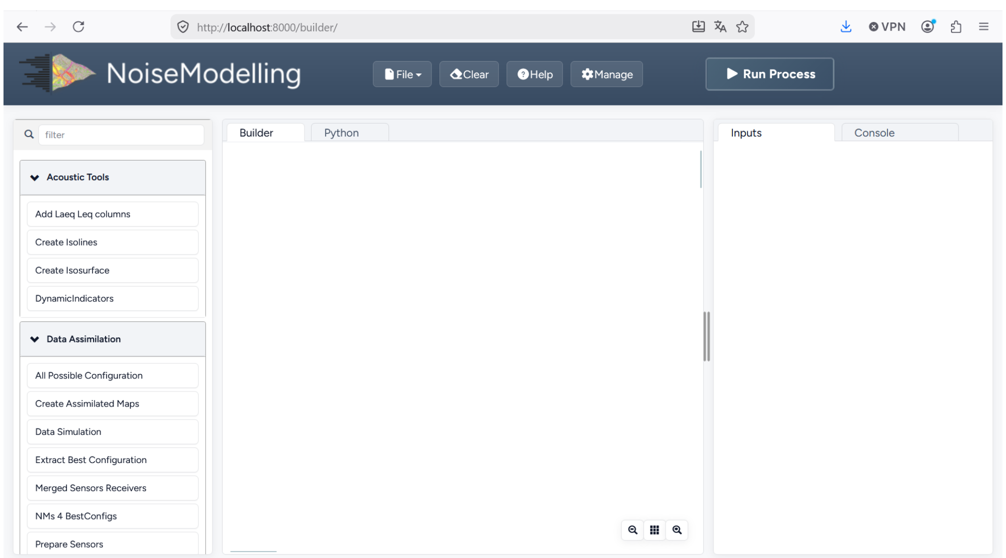


Fig. 1: Noise Modelling GUI landing page

Click `builder` to open the builder.



You are now ready to discover the power of NoiseModelling!

5.4 Validation

5.4.1 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.

5.4.2 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.

5.5 Scientific production

Below is a **non-exhaustive** list of scientific publications in which NoiseModelling has been used.

- Articles (co-)written by members of the NoiseModelling team, belonging to the Joint Research Unit in Environmental Acoustics, are in the `UMRAE` tab.
- Articles written outside the UMRAE are in the `Not UMRAE` tab.

Tip: If you have done work with NoiseModelling and your article is not referenced in this list, please let us know (contact@noise-planet.org).

All

UMRAE

Not UMRAE

Noise Mapping (END)

Traffic dynamics

Other than Traffic noise

Fauna impacts

AI & Surrogates

Data Assimilation

Teaching

Software

1. Cristian-Gabriel Alionte and Daniel-Constantin Comeaga. Noise assessment of the small-scale wind farm. *E3S Web Conf.*, 112:02011, 2019. Publisher: EDP Sciences. URL: https://www.e3s-conferences.org/articles/e3sconf/abs/2019/38/e3sconf_te-re-rd18_02011/e3sconf_te-re-rd18_02011.html (visited on 2025-06-06), doi:10.1051/e3sconf/201911202011.

2. Pierre Aumond, Erwan Bocher, David Ecotiere, Nicolas Fortin, Benoit Gauvreau, Gwenael Guillaume, and Gwendall Petit. Improvement of city noise map production processes and sensitivity analysis to noise models inputs. In *EuroNoise 2021 : 12th European Congress and Exposition on Noise Control Engineering*, 10 p. MADERE, Portugal, October 2021. URL: https://hal.science/hal-03616731 (visited on 2025-06-10).

3. Pierre Aumond, Arna Can, Vivien Mallet, Benoit Gauvreau, and Gwenaël Guillaume. Global sensitivity analysis for urban noise modelling. In *Proceedings of the 23rd International Congress on Acoustics*, –. Aachen, Germany, 2019. URL: https://publications.rwth-aachen.de/record/769725, doi:10.18154/RWTH-CONV-239314.

4. Pierre Aumond, Sophie Cariou, Olivier Chiello, David Ecotière, Adrien Le Bellec, Damien Maltete, Claire Marconot, Nicolas Fortin, Sylvain Palominos, Gwendall Petit, and Judicaël Picaut. Strategic Noise Mapping in France to 2023: Coupling a national database with the open-source model NoiseModelling. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(5):2617–2624, February 2023. doi:10.3397/IN_2022_0367.

5. Pierre Aumond, Sophie Cariou, Olivier Chiello, and others. Couplage entre la base de données nationale PlaMADE et l'outil open-source NoiseModelling pour la réalisation de cartes de bruit stratégiques. In *16ème Congrès Français d'Acoustique (CFA2022)*. Marseille, France, 2022. URL: https://hal.science/hal-03848495/.

6. Pierre Aumond, Nicolas Fortin, and Arnaud Can. Overview of the NoiseModelling open-source software version 3 and its applications. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 261, 2005–2011. 2020. Issue: 4. URL: https://www.youtube.com/watch?v=V1-niMT9cYE.

7. Pierre Aumond, L. Jacquesson, and Arnaud Can. Probabilistic modeling framework for multisource sound mapping. In *Proceedings of Euronoise*, –. 2018. URL: https://www.euronoise2018.eu/docs/papers/102_Euronoise2018.pdf, doi:10.1016/j.apacoust.2018.04.017.

8. Pierre Aumond, Léo Jacquesson, and Arnaud Can. Probabilistic modeling framework for multisource sound mapping. *Applied Acoustics*, 139:34–43, October 2018. URL: https://www.sciencedirect.com/science/article/pii/S0003682X17311283 (visited on 2025-06-06), doi:10.1016/j.apacoust.2018.04.017.

9. Sacha BACLET, Johan NYGREN, and Romain RUMPLER. Impact of car electrification on urban noise pollution: a microscopic traffic simulation study. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(4):7846–7855, October 2024. doi:10.3397/IN_2024_4014.

10. Sacha Baclet, Siddharth Venkataraman, Erik Gomez, and Hamza Bouchouireb. A machine learning-and compressed sensing-based approach for surrogate modelling in environmental acoustics: towards fast evaluation of building façade road traffic noise levels. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 6040–6051. Glasgow, Scotland, 2023. URL: https://www.ingentaconnect.com/content/ince/incecp/2023/00000265/00000001/art00006, doi:10.3397/IN_2022_0898.

11. Sacha Baclet, Siddharth Venkataraman, and Romain Rumpler. A methodology to assess the impact of driving noise from individual vehicles in an urban environment. In *27th International Congress on Sound and Vibration*. 2021. URL: https://kth.diva-portal.org/smash/get/diva2:1641387/FULLTEXT01.pdf.

12. Sacha Baclet, Siddharth Venkataraman, Romain Rumpler, Robin Billsjö, Johannes Horvath, and Per Erik Österlund. From strategic noise maps to receiver-centric noise exposure sensitivity mapping. *Transportation Research Part D: Transport and Environment*, 102:103114, January 2022. URL: https://www.sciencedirect.com/science/article/pii/S1361920921004089 (visited on 2025-06-10), doi:10.1016/j.trd.2021.103114.

13. Jesús López Baeza, Julia L. Sievert, André Landwehr, Jonas Luft, Philipp Preuner, Jürgen Bruns-Berentelg, Ariel Noyman, and Joerg Rainer Noennig. CityScope Platform for Real-Time Analysis and Decision-Support in Urban Design Competitions. *IJEPR*, 10(4):121–137, October 2021. Publisher: IGI Global Scientific Publishing. URL: https://www.igi-global.com/article/cityscope-platform-for-real-time-analysis-and-decision-support-in-urban-design-competitions/www.igi-global.com/article/cityscope-platform-for-real-time-analysis-and-decision-support-in-urban-design-competitions/278826 (visited on 2025-06-10), doi:10.4018/IJEPR.20211001.oa8.

14. Jérôme Bisson, Sabrina Côté, Henri Curry, William Fauteux, Jérémie Hatier, Othmane Labsir, Léa Rivard, Audrey Guy, Gaëlle Belleau-Magnat, Philippe Apparicio, Johanne Roby, and Olivier ROBIN. Setting up light and noise maps to define the contours of a reserve of darkness and silence in Sherbrooke, Québec, Canada. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(6):5334–5342, October 2024. doi:10.3397/IN_2024_3575.

15. Erwan Bocher, Gwenaël Guillaume, Judicaël Picaut, Gwendall Petit, and Nicolas Fortin. NoiseModelling: An Open Source GIS Based Tool to Produce Environmental Noise Maps. *ISPRS International Journal of Geo-Information*, 8(3):130, March 2019. Publisher: MDPI. URL: https://hal.science/hal-02057736 (visited on 2025-06-06), doi:10.3390/ijgi8030130.

16. Arnaud Can, Pierre Aumond, Cécile Bécarie, and Ludovic Leclercq. Approche dynamique pour l'étude de l'emprise spatiale du bruit de trafic routier aux heures de pointe. *Recherche en Transport Sécurité*, 2018(1):1–11, 2018. URL: https://hal.science/hal-02482315/, doi:10.1016/j.retrec.2018.07.001.

17. Arnaud Can, Pierre Aumond, Cécile Bécarie, and Ludovic Leclercq. Dynamic approach for the study of the spatial impact of road traffic noise at peak hours. In *Proceedings of the 23rd International Congress on Acoustics*, –. Aachen, Germany, 2019. URL: https://publications.rwth-aachen.de/record/769733, doi:10.18154/RWTH-CONV-239322.

18. Kerem Ege, Alain Berry, Olivier Robin, and Etienne Parizet. The Sherbrooke-Lyon Research Training Program in Acoustics - Projet Samuel de Champlain 2022-2024. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(10):1746–1752, October 2024. doi:10.3397/IN_2024_3068.

19. Jasso Espadaler-Clapés, Emmanouil Barmpounakis, and Nikolas Geroliminis. Traffic congestion and noise emissions with detailed vehicle trajectories from UAVs. *Transportation Research Part D: Transport and Environment*, 121:103822, August 2023. URL: https://www.sciencedirect.com/science/article/pii/S1361920923002195 (visited on 2025-06-06), doi:10.1016/j.trd.2023.103822.

20. Leonardo Galassi Luquezi, Valentin Le Bescond, Pierre Aumond, Pascal Gastineau, and Arnaud Can. Current limitations and opportunities for improvements of agent-based transport models for noise exposure assessment. *Journal of Environmental Management*, 368:122129, September 2024. URL: https://www.sciencedirect.com/science/article/pii/S0301479724021157 (visited on 2025-06-10), doi:10.1016/j.jenvman.2024.122129.

21. G Graziuso, A B Francavilla, S Mancini, and C Guarnaccia. Open-source software tools for strategic noise mapping: a case study. *J. Phys.: Conf. Ser.*, 2162(1):012014, January 2022. Publisher: IOP Publishing. URL: https://dx.doi.org/10.1088/1742-6596/2162/1/012014 (visited on 2025-06-06), doi:10.1088/1742-6596/2162/1/012014.

22. Gwenaël Guillaume, Erwan Bocher, Pierre Aumond, and David Ecotière. Cartographie du bruit routier à partir de données OpenStreetMap. In *16ème Congrès Français d'Acoustique (CFA2022)*. Marseille, France, 2022. URL: https://hal.science/hal-03848394/.

23. Irune Indacoechea-Vega, Helena Miera-Dominguez, Pedro Lastra-González, and Daniel Castro-Fresno. Life cycle approach for evaluating the environmental and economic viability of low-noise asphalt pavements. *Journal of Cleaner Production*, 466:142785, August 2024. URL: https://www.sciencedirect.com/science/article/pii/S0959652624022339 (visited on 2025-06-10), doi:10.1016/j.jclepro.2024.142785.

24. V. Le Bescond, P. Gastineau, P. Aumond, L. Galassi Luquezi, J. Nygren, L. Soulhac, P. Charvolin-Volta, D. Lejri, and A. Can. Population noise exposure modelling using large scale multi-agent simulation. In *Proceedings of the 10th Convention of the European Acoustics Association Forum Acusticum 2023*, 2659–2662. Turin, Italy, January 2024. European Acoustics Association. URL: https://dael.euracoustics.org/confs/fa2023/data/articles/000680.pdf (visited on 2025-06-06), doi:10.61782/fa.2023.0680.

25. Valentin Le Bescond, Arnaud Can, Pierre Aumond, and Pascal Gastineau. Open-source modeling chain for the dynamic assessment of road traffic noise exposure. *Transportation Research Part D: Transport and Environment*, 94:102793, May 2021. URL: https://www.sciencedirect.com/science/article/pii/S1361920921000973 (visited on 2025-06-06), doi:10.1016/j.trd.2021.102793.

26. Ingrid Legriffon and Elise Ruaud. Drone fleet noise impact calculation - a methodology. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 1675–1680. Nantes, France, 2024. URL: https://hal.science/hal-04711864v1/, doi:10.3397/IN_2024_3049.

27. Antoine Lesieur, Pierre Aumond, Arnaud Can, and Vivien Mallet. Une cartographie du bruit plus rapide et plus précise combinant méta-modélisation et assimilation de données. In *16ème Congrès Français d'Acoustique, CFA2022*. Marseille, France, April 2022. Société Française d'Acoustique and Laboratoire de Mécanique et d'Acoustique. URL: https://hal.science/hal-03848396 (visited on 2025-06-06).

28. Antoine Lesieur, Pierre Aumond, Vivien Mallet, and Arnaud Can. Meta-modeling for urban noise mapping. *J Acoust Soc Am*, 148(6):3671, December 2020. URL: https://pubmed.ncbi.nlm.nih.gov/33379895/, doi:10.1121/10.0002866.

29. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Data assimilation for urban noise mapping with a meta-model. *Applied Acoustics*, 178:107938, 2021. URL: https://www.sciencedirect.com/science/article/abs/pii/S0003682X21000311, doi:10.1016/j.apacoust.2021.107938.

30. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Faster and more accurate noise mapping combining meta-modeling and data assimilation. In *Proceedings of Euronoise 2021*. Madeira, Portugal, 2021. URL: https://cense.ifsttar.fr/fileadmin/contributeurs/CENSE/Documents/Euronoise_2021_WP4.pdf.

31. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Inverse modeling and joint state-parameter estimation with a noise mapping meta-model. *Journal of the Acoustical Society of America*, 149(6):3961–3974, 2021. URL: https://pubs.aip.org/asa/jasa/article/149/6/3961/1076275, doi:10.1121/10.0004984.

32. Arturo Maristany, L.Abadía, D.Moyano, A.Pacharoni, M.Durán, S.Coca, and B.Yelicich. Mapa de ruido de la ciudad de Córdoba. Promeros resultados mediante el uso de software de codigo abierto. In *XVIII Congreso Argentino de Acústica 2023 (AdAA2023)*. Argentina, January 2023.

33. Alexandra L. Montenegro, Dulia Melluso, Gianmarco Stasi, Andrea Panci, Matteo Bolognese, Diego Palazzuoli, Mauro Cerchiai, and Gaetano Licitra. An open-source pipeline in noise modelling and noise exposure reduction in a port city. In *Proceedings of the 30th International Congress on Sound and Vibration (ICSV30)*. Amsterdam, Netherlands, 2024. URL: https://iiav.org/content/archives_icsv_last/2024_icsv30/content/papers/papers/full_paper_183_20240415175901591.pdf.

34. Zahra Nourmohammadi, Tanapon Lilasathapornkit, Mudabber Ashfaq, Ziyuan Gu, and Meead Saberi. Mapping Urban Environmental Performance with Emerging Data Sources: A Case of Urban Greenery and Traffic Noise in Sydney, Australia. *Sustainability*, 13(2):605, January 2021. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/2071-1050/13/2/605 (visited on 2025-06-10), doi:10.3390/su13020605.

35. Johan Nygren, Valentin Le Bescond, Arnaud Can, Pierre Aumond, Pascal Gastineau, Susann Boij, Romain Rumpler, and Ciarán J. O'Reilly. Agent-specific, activity-based noise impact assessment using noise exposure cost. *Sustainable Cities and Society*, 103:105278, April 2024. URL: https://www.sciencedirect.com/science/article/pii/S2210670724001069 (visited on 2025-06-10), doi:10.1016/j.scs.2024.105278.

36. Antonio PASCALE, Domenico ROSSI, Aurora MASCOLO, Margarida COELHO, and Claudio GUARNACCIA. Road Traffic Noise Levels Estimations by means of Fully Dynamic and Microscopic Approach Compared to CNOSSOS-EU Model. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(7):4767–4778, October 2024. doi:10.3397/IN_2024_3506.

37. G. Quintero, P. Aumond, A. Can, A. Balastegui, and J. Romeu. Statistical requirements for noise mapping based on mobile measurements using bikes. *Applied Acoustics*, 156:271–278, December 2019. URL: https://www.sciencedirect.com/science/article/pii/S0003682X19302087 (visited on 2025-06-06), doi:10.1016/j.apacoust.2019.07.020.

38. J Siliézar, P Aumond, P Chapron, M Péroche, and A Can. Méthode d'évaluation de l'audibilité d'un système d'alerte SAIP. In *16ème Congrès Français d'Acoustique*, 6 p. MARSEILLE, France, April 2022. URL: https://hal.science/hal-03777367 (visited on 2025-06-06).

39. Ignacio Soto-Molina, Miguel Ausejo Prieto, and Rosa Maria Arce Ruiz. Enhancing noise assessment accuracy: FEM as an advanced complement to CNOSSOS-EU. *Noise & Vibration Worldwide*, 2025. URL: https://journals.sagepub.com/doi/10.1177/09574565251319271, doi:10.1177/09574565251319271.

40. Martin Spitznagel and Janis Keuper. Urban Sound Propagation: a Benchmark for 1-Step Generative Modeling of Complex Physical Systems. *arXiv.org*, March 2024. URL: https://arxiv.org/abs/2403.10904v2 (visited on 2025-06-10).

41. Martin Spitznagel, Jan Vaillant, and Janis Keuper. PhysicsGen: Can Generative Models Learn from Images to Predict Complex Physical Relations? In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 11125–11134. 2025. URL: https://openaccess.thecvf.com/content/CVPR2025/html/Spitznagel_PhysicsGen_Can_Generative_Models_Learn_from_Images_to_Predict_Complex_CVPR_2025_paper.html (visited on 2025-06-10).

42. Junta Tagusari. Dynamic Generation of Noise Maps: Accurate and Efficient Assessment of Noise Exposure. 2023. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4612924, doi:10.2139/ssrn.4612924.
43. Junta Tagusari. H-RISK with NoiseModelling: a QGIS plugin to predict environmental noise and estimate health risks. *Journal of Open Source Software*, 9(93):6023, 2024. URL: https://joss.theoj.org/papers/10.21105/joss.06023, doi:10.21105/joss.06023.
44. Junta Tagusari. Prediction of road traffic noise in the vicinity of trunk roads in Japan using digital road map platform. *Acoustical Science and Technology*, June 2024. URL: https://www.jstage.jst.go.jp/article/ast/advpub/0/advpub_e24.50/_pdf, doi:10.1250/ast.e24.50.
45. Junta Tagusari. Small-scale noise mapping for arbitrary regions using open source noise prediction module "NoiseModelling" and global databases. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(9):2621–2627, October 2024. doi:10.3397/IN_2024_3212.
46. Chris Tampère, Paul Ortmann, Karel Jedlička, Walter Lohman, and Stijn Janssen. Future Ready Local Digital Twins and the Use of Predictive Simulations: The Case of Traffic and Traffic Impact Modelling. In Lieven Raes, Susie Ruston McAleer, Ingrid Croket, Pavel Kogut, Martin Brynskov, and Stefan Lefever, editors, *Decide Better: Open and Interoperable Local Digital Twins*, pages 203–230. Springer Nature Switzerland, Cham, 2025. URL: https://doi.org/10.1007/978-3-031-81451-8_8 (visited on 2025-06-10), doi:10.1007/978-3-031-81451-8_8.
47. Zhiyong Wang, Tessio Novack, Yingwei Yan, and Alexander Zipf. Quiet Route Planning for Pedestrians in Traffic Noise Polluted Environments. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7573–7584, December 2021. URL: https://ieeexplore.ieee.org/abstract/document/9139350 (visited on 2025-06-06), doi:10.1109/TITS.2020.3004660.

1. Pierre Aumond, Erwan Bocher, David Ecotiere, Nicolas Fortin, Benoit Gauvreau, Gwenael Guillaume, and Gwendall Petit. Improvement of city noise map production processes and sensitivity analysis to noise models inputs. In *EuroNoise 2021 : 12th European Congress and Exposition on Noise Control Engineering*, 10 p. MADERE, Portugal, October 2021. URL: https://hal.science/hal-03616731 (visited on 2025-06-10).
2. Pierre Aumond, Arna Can, Vivien Mallet, Benoit Gauvreau, and Gwenaël Guillaume. Global sensitivity analysis for urban noise modelling. In *Proceedings of the 23rd International Congress on Acoustics*, –. Aachen, Germany, 2019. URL: https://publications.rwth-aachen.de/record/769725, doi:10.18154/RWTH-CONV-239314.
3. Pierre Aumond, Sophie Cariou, Olivier Chiello, David Ecotière, Adrien Le Bellec, Damien Maltete, Claire Marconot, Nicolas Fortin, Sylvain Palominos, Gwendall Petit, and Judicaël Picaut. Strategic Noise Mapping in France to 2023: Coupling a national database with the open-source model NoiseModelling. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(5):2617–2624, February 2023. doi:10.3397/IN_2022_0367.
4. Pierre Aumond, Sophie Cariou, Olivier Chiello, and others. Couplage entre la base de données nationale PlaMADE et l'outil open-source NoiseModelling pour la réalisation de cartes de bruit stratégiques. In *16ème Congrès Français d'Acoustique (CFA2022)*. Marseille, France, 2022. URL: https://hal.science/hal-03848495/.
5. Pierre Aumond, Nicolas Fortin, and Arnaud Can. Overview of the NoiseModelling open-source software version 3 and its applications. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 261, 2005–2011. 2020. Issue: 4. URL: https://www.youtube.com/watch?v=V1-niMT9cYE.
6. Pierre Aumond, L. Jacquesson, and Arnaud Can. Probabilistic modeling framework for multisource sound mapping. In *Proceedings of Euronoise*, –. 2018. URL: https://www.euronoise2018.eu/docs/papers/102_Euronoise2018.pdf, doi:10.1016/j.apacoust.2018.04.017.
7. Pierre Aumond, Léo Jacquesson, and Arnaud Can. Probabilistic modeling framework for multisource sound mapping. *Applied Acoustics*, 139:34–43, October 2018. URL: https://www.sciencedirect.com/science/article/pii/S0003682X17311283 (visited on 2025-06-06), doi:10.1016/j.apacoust.2018.04.017.
8. Erwan Bocher, Gwenaël Guillaume, Judicaël Picaut, Gwendall Petit, and Nicolas Fortin. NoiseModelling: An Open Source GIS Based Tool to Produce Environmental Noise Maps. *ISPRS International Journal of Geo-*

Information, 8(3):130, March 2019. Publisher: MDPI. URL: https://hal.science/hal-02057736 (visited on 2025-06-06), doi:10.3390/ijgi8030130.

9. Arnaud Can, Pierre Aumond, Cécile Bécarie, and Ludovic Leclercq. Approche dynamique pour l'étude de l'emprise spatiale du bruit de trafic routier aux heures de pointe. *Recherche en Transport Sécurité*, 2018(1):1–11, 2018. URL: https://hal.science/hal-02482315/, doi:10.1016/j.retrec.2018.07.001.

10. Arnaud Can, Pierre Aumond, Cécile Bécarie, and Ludovic Leclercq. Dynamic approach for the study of the spatial impact of road traffic noise at peak hours. In *Proceedings of the 23rd International Congress on Acoustics*, –. Aachen, Germany, 2019. URL: https://publications.rwth-aachen.de/record/769733, doi:10.18154/RWTH-CONV-239322.

11. Leonardo Galassi Luquezi, Valentin Le Bescond, Pierre Aumond, Pascal Gastineau, and Arnaud Can. Current limitations and opportunities for improvements of agent-based transport models for noise exposure assessment. *Journal of Environmental Management*, 368:122129, September 2024. URL: https://www.sciencedirect.com/science/article/pii/S0301479724021157 (visited on 2025-06-10), doi:10.1016/j.jenvman.2024.122129.

12. Gwenaël Guillaume, Erwan Bocher, Pierre Aumond, and David Ecotière. Cartographie du bruit routier à partir de données OpenStreetMap. In *16ème Congrès Français d'Acoustique (CFA2022)*. Marseille, France, 2022. URL: https://hal.science/hal-03848394/.

13. V. Le Bescond, P. Gastineau, P. Aumond, L. Galassi Luquezi, J. Nygren, L. Soulhac, P. Charvolin-Volta, D. Lejri, and A. Can. Population noise exposure modelling using large scale multi-agent simulation. In *Proceedings of the 10th Convention of the European Acoustics Association Forum Acusticum 2023*, 2659–2662. Turin, Italy, January 2024. European Acoustics Association. URL: https://dael.euracoustics.org/confs/fa2023/data/articles/000680.pdf (visited on 2025-06-06), doi:10.61782/fa.2023.0680.

14. Valentin Le Bescond, Arnaud Can, Pierre Aumond, and Pascal Gastineau. Open-source modeling chain for the dynamic assessment of road traffic noise exposure. *Transportation Research Part D: Transport and Environment*, 94:102793, May 2021. URL: https://www.sciencedirect.com/science/article/pii/S1361920921000973 (visited on 2025-06-06), doi:10.1016/j.trd.2021.102793.

15. Antoine Lesieur, Pierre Aumond, Arnaud Can, and Vivien Mallet. Une cartographie du bruit plus rapide et plus précise combinant méta-modélisation et assimilation de données. In *16ème Congrès Français d'Acoustique, CFA2022*. Marseille, France, April 2022. Société Française d'Acoustique and Laboratoire de Mécanique et d'Acoustique. URL: https://hal.science/hal-03848396 (visited on 2025-06-06).

16. Antoine Lesieur, Pierre Aumond, Vivien Mallet, and Arnaud Can. Meta-modeling for urban noise mapping. *J Acoust Soc Am*, 148(6):3671, December 2020. URL: https://pubmed.ncbi.nlm.nih.gov/33379895/, doi:10.1121/10.0002866.

17. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Data assimilation for urban noise mapping with a meta-model. *Applied Acoustics*, 178:107938, 2021. URL: https://www.sciencedirect.com/science/article/abs/pii/S0003682X21000311, doi:10.1016/j.apacoust.2021.107938.

18. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Faster and more accurate noise mapping combining meta-modeling and data assimilation. In *Proceedings of Euronoise 2021*. Madeira, Portugal, 2021. URL: https://cense.ifsttar.fr/fileadmin/contributeurs/CENSE/Documents/Euronoise_2021_WP4.pdf.

19. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Inverse modeling and joint state-parameter estimation with a noise mapping meta-model. *Journal of the Acoustical Society of America*, 149(6):3961–3974, 2021. URL: https://pubs.aip.org/asa/jasa/article/149/6/3961/1076275, doi:10.1121/10.0004984.

20. Johan Nygren, Valentin Le Bescond, Arnaud Can, Pierre Aumond, Pascal Gastineau, Susann Boij, Romain Rumpler, and Ciarán J. O'Reilly. Agent-specific, activity-based noise impact assessment using noise exposure cost. *Sustainable Cities and Society*, 103:105278, April 2024. URL: https://www.sciencedirect.com/science/article/pii/S2210670724001069 (visited on 2025-06-10), doi:10.1016/j.scs.2024.105278.

21. G. Quintero, P. Aumond, A. Can, A. Balastegui, and J. Romeu. Statistical requirements for noise mapping based on mobile measurements using bikes. *Applied Acoustics*, 156:271–278, December

2019. URL: https://www.sciencedirect.com/science/article/pii/S0003682X19302087 (visited on 2025-06-06), doi:10.1016/j.apacoust.2019.07.020.

22. J Siliézar, P Aumond, P Chapron, M Péroche, and A Can. Méthode d'évaluation de l'audibilité d'un système d'alerte SAIP. In *16ème Congrès Français d'Acoustique*, 6 p. MARSEILLE, France, April 2022. URL: https://hal.science/hal-03777367 (visited on 2025-06-06).

1. Cristian-Gabriel Alionte and Daniel-Constantin Comeaga. Noise assessment of the small-scale wind farm. *E3S Web Conf.*, 112:02011, 2019. Publisher: EDP Sciences. URL: https://www.e3s-conferences.org/articles/e3sconf/abs/2019/38/e3sconf_te-re-rd18_02011/e3sconf_te-re-rd18_02011.html (visited on 2025-06-06), doi:10.1051/e3sconf/201911202011.
2. Sacha BACLET, Johan NYGREN, and Romain RUMPLER. Impact of car electrification on urban noise pollution: a microscopic traffic simulation study. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(4):7846–7855, October 2024. doi:10.3397/IN_2024_4014.
3. Sacha Baclet, Siddharth Venkataraman, Erik Gomez, and Hamza Bouchouireb. A machine learning-and compressed sensing-based approach for surrogate modelling in environmental acoustics: towards fast evaluation of building façade road traffic noise levels. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 6040–6051. Glasgow, Scotland, 2023. URL: https://www.ingentaconnect.com/content/ince/incecp/2023/00000265/00000001/art00006, doi:10.3397/IN_2022_0898.
4. Sacha Baclet, Siddharth Venkataraman, and Romain Rumpler. A methodology to assess the impact of driving noise from individual vehicles in an urban environment. In *27th International Congress on Sound and Vibration*. 2021. URL: https://kth.diva-portal.org/smash/get/diva2:1641387/FULLTEXT01.pdf.
5. Sacha Baclet, Siddharth Venkataraman, Romain Rumpler, Robin Billsjö, Johannes Horvath, and Per Erik Österlund. From strategic noise maps to receiver-centric noise exposure sensitivity mapping. *Transportation Research Part D: Transport and Environment*, 102:103114, January 2022. URL: https://www.sciencedirect.com/science/article/pii/S1361920921004089 (visited on 2025-06-10), doi:10.1016/j.trd.2021.103114.
6. Jesús López Baeza, Julia L. Sievert, André Landwehr, Jonas Luft, Philipp Preuner, Jürgen Bruns-Berentelg, Ariel Noyman, and Joerg Rainer Noennig. CityScope Platform for Real-Time Analysis and Decision-Support in Urban Design Competitions. *IJEPR*, 10(4):121–137, October 2021. Publisher: IGI Global Scientific Publishing. URL: https://www.igi-global.com/article/cityscope-platform-for-real-time-analysis-and-decision-support-in-urban-design-competitions/www.igi-global.com/article/cityscope-platform-for-real-time-analysis-and-decision-support-in-urban-design-competitions/278826 (visited on 2025-06-10), doi:10.4018/IJEPR.20211001.oa8.
7. Jérôme Bisson, Sabrina Côté, Henri Curry, William Fauteux, Jérémie Hatier, Othmane Labsir, Léa Rivard, Audrey Guy, Gaëlle Belleau-Magnat, Philippe Apparicio, Johanne Roby, and Olivier ROBIN. Setting up light and noise maps to define the contours of a reserve of darkness and silence in Sherbrooke, Québec, Canada. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(6):5334–5342, October 2024. doi:10.3397/IN_2024_3575.
8. Kerem Ege, Alain Berry, Olivier Robin, and Etienne Parizet. The Sherbrooke-Lyon Research Training Program in Acoustics - Projet Samuel de Champlain 2022-2024. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(10):1746–1752, October 2024. doi:10.3397/IN_2024_3068.
9. Jasso Espadaler-Clapés, Emmanouil Barmpounakis, and Nikolas Geroliminis. Traffic congestion and noise emissions with detailed vehicle trajectories from UAVs. *Transportation Research Part D: Transport and Environment*, 121:103822, August 2023. URL: https://www.sciencedirect.com/science/article/pii/S1361920923002195 (visited on 2025-06-06), doi:10.1016/j.trd.2023.103822.
10. G Graziuso, A B Francavilla, S Mancini, and C Guarnaccia. Open-source software tools for strategic noise mapping: a case study. *J. Phys.: Conf. Ser.*, 2162(1):012014, January 2022. Publisher: IOP Publishing. URL: https://dx.doi.org/10.1088/1742-6596/2162/1/012014 (visited on 2025-06-06), doi:10.1088/1742-6596/2162/1/012014.

11. Irune Indacoechea-Vega, Helena Miera-Dominguez, Pedro Lastra-González, and Daniel Castro-Fresno. Life cycle approach for evaluating the environmental and economic viability of low-noise asphalt pavements. *Journal of Cleaner Production*, 466:142785, August 2024. URL: https://www.sciencedirect.com/science/article/pii/S0959652624022339 (visited on 2025-06-10), doi:10.1016/j.jclepro.2024.142785.

12. Ingrid Legriffon and Elise Ruaud. Drone fleet noise impact calculation - a methodology. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 1675–1680. Nantes, France, 2024. URL: https://hal.science/hal-04711864v1/, doi:10.3397/IN_2024_3049.

13. Arturo Maristany, L.Abadía, D.Moyano, A.Pacharoni, M.Durán, S.Coca, and B.Yelicich. Mapa de ruido de la ciudad de Córdoba. Promeros resultados mediante el uso de software de codigo abierto. In *XVIII Congreso Argentino de Acústica 2023 (AdAA2023)*. Argentina, January 2023.

14. Alexandra L. Montenegro, Dulia Melluso, Gianmarco Stasi, Andrea Panci, Matteo Bolognese, Diego Palazzuoli, Mauro Cerchiai, and Gaetano Licitra. An open-source pipeline in noise modelling and noise exposure reduction in a port city. In *Proceedings of the 30th International Congress on Sound and Vibration (ICSV30)*. Amsterdam, Netherlands, 2024. URL: https://iiav.org/content/archives_icsv_last/2024_icsv30/content/papers/papers/full_paper_183_20240415175901591.pdf.

15. Zahra Nourmohammadi, Tanapon Lilasathapornkit, Mudabber Ashfaq, Ziyuan Gu, and Meead Saberi. Mapping Urban Environmental Performance with Emerging Data Sources: A Case of Urban Greenery and Traffic Noise in Sydney, Australia. *Sustainability*, 13(2):605, January 2021. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/2071-1050/13/2/605 (visited on 2025-06-10), doi:10.3390/su13020605.

16. Antonio PASCALE, Domenico ROSSI, Aurora MASCOLO, Margarida COELHO, and Claudio GUARNACCIA. Road Traffic Noise Levels Estimations by means of Fully Dynamic and Microscopic Approach Compared to CNOSSOS-EU Model. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(7):4767–4778, October 2024. doi:10.3397/IN_2024_3506.

17. Ignacio Soto-Molina, Miguel Ausejo Prieto, and Rosa Maria Arce Ruiz. Enhancing noise assessment accuracy: FEM as an advanced complement to CNOSSOS-EU. *Noise & Vibration Worldwide*, 2025. URL: https://journals.sagepub.com/doi/10.1177/09574565251319271, doi:10.1177/09574565251319271.

18. Martin Spitznagel and Janis Keuper. Urban Sound Propagation: a Benchmark for 1-Step Generative Modeling of Complex Physical Systems. *arXiv.org*, March 2024. URL: https://arxiv.org/abs/2403.10904v2 (visited on 2025-06-10).

19. Martin Spitznagel, Jan Vaillant, and Janis Keuper. PhysicsGen: Can Generative Models Learn from Images to Predict Complex Physical Relations? In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 11125–11134. 2025. URL: https://openaccess.thecvf.com/content/CVPR2025/html/Spitznagel_PhysicsGen_Can_Generative_Models_Learn_from_Images_to_Predict_Complex_CVPR_2025_paper.html (visited on 2025-06-10).

20. Junta Tagusari. Dynamic Generation of Noise Maps: Accurate and Efficient Assessment of Noise Exposure. 2023. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4612924, doi:10.2139/ssrn.4612924.

21. Junta Tagusari. H-RISK with NoiseModelling: a QGIS plugin to predict environmental noise and estimate health risks. *Journal of Open Source Software*, 9(93):6023, 2024. URL: https://joss.theoj.org/papers/10.21105/joss.06023, doi:10.21105/joss.06023.

22. Junta Tagusari. Prediction of road traffic noise in the vicinity of trunk roads in Japan using digital road map platform. *Acoustical Science and Technology*, June 2024. URL: https://www.jstage.jst.go.jp/article/ast/advpub/0/advpub_e24.50/_pdf, doi:10.1250/ast.e24.50.

23. Junta Tagusari. Small-scale noise mapping for arbitrary regions using open source noise prediction module "NoiseModelling" and global databases. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(9):2621–2627, October 2024. doi:10.3397/IN_2024_3212.

24. Chris Tampère, Paul Ortmann, Karel Jedlička, Walter Lohman, and Stijn Janssen. Future Ready Local Digital Twins and the Use of Predictive Simulations: The Case of Traffic and Traffic Impact Modelling. In Lieven Raes, Susie Ruston McAleer, Ingrid Croket, Pavel Kogut, Martin Brynskov, and Stefan Lefever, editors, *Decide Better: Open and Interoperable Local Digital Twins*, pages 203–230. Springer Nature Switzerland, Cham, 2025. URL: https://doi.org/10.1007/978-3-031-81451-8_8 (visited on 2025-06-10), doi:10.1007/978-3-031-81451-8_8.

25. Zhiyong Wang, Tessio Novack, Yingwei Yan, and Alexander Zipf. Quiet Route Planning for Pedestrians in Traffic Noise Polluted Environments. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7573–7584, December 2021. URL: https://ieeexplore.ieee.org/abstract/document/9139350 (visited on 2025-06-06), doi:10.1109/TITS.2020.3004660.

1. Pierre Aumond, Erwan Bocher, David Ecotiere, Nicolas Fortin, Benoit Gauvreau, Gwenael Guillaume, and Gwendall Petit. Improvement of city noise map production processes and sensitivity analysis to noise models inputs. In *EuroNoise 2021 : 12th European Congress and Exposition on Noise Control Engineering*, 10 p. MADERE, Portugal, October 2021. URL: https://hal.science/hal-03616731 (visited on 2025-06-10).

2. Pierre Aumond, Arna Can, Vivien Mallet, Benoit Gauvreau, and Gwenaël Guillaume. Global sensitivity analysis for urban noise modelling. In *Proceedings of the 23rd International Congress on Acoustics*, –. Aachen, Germany, 2019. URL: https://publications.rwth-aachen.de/record/769725, doi:10.18154/RWTH-CONV-239314.

3. Pierre Aumond, Sophie Cariou, Olivier Chiello, David Ecotière, Adrien Le Bellec, Damien Maltete, Claire Marconot, Nicolas Fortin, Sylvain Palominos, Gwendall Petit, and Judicaël Picaut. Strategic Noise Mapping in France to 2023: Coupling a national database with the open-source model NoiseModelling. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(5):2617–2624, February 2023. doi:10.3397/IN_2022_0367.

4. Pierre Aumond, Sophie Cariou, Olivier Chiello, and others. Couplage entre la base de données nationale PlaMADE et l'outil open-source NoiseModelling pour la réalisation de cartes de bruit stratégiques. In *16ème Congrès Français d'Acoustique (CFA2022)*. Marseille, France, 2022. URL: https://hal.science/hal-03848495/.

5. Sacha Baclet, Siddharth Venkataraman, Romain Rumpler, Robin Billsjö, Johannes Horvath, and Per Erik Österlund. From strategic noise maps to receiver-centric noise exposure sensitivity mapping. *Transportation Research Part D: Transport and Environment*, 102:103114, January 2022. URL: https://www.sciencedirect.com/science/article/pii/S1361920921004089 (visited on 2025-06-10), doi:10.1016/j.trd.2021.103114.

6. Erwan Bocher, Gwenaël Guillaume, Judicaël Picaut, Gwendall Petit, and Nicolas Fortin. NoiseModelling: An Open Source GIS Based Tool to Produce Environmental Noise Maps. *ISPRS International Journal of Geo-Information*, 8(3):130, March 2019. Publisher: MDPI. URL: https://hal.science/hal-02057736 (visited on 2025-06-06), doi:10.3390/ijgi8030130.

7. G Graziuso, A B Francavilla, S Mancini, and C Guarnaccia. Open-source software tools for strategic noise mapping: a case study. *J. Phys.: Conf. Ser.*, 2162(1):012014, January 2022. Publisher: IOP Publishing. URL: https://dx.doi.org/10.1088/1742-6596/2162/1/012014 (visited on 2025-06-06), doi:10.1088/1742-6596/2162/1/012014.

8. Gwenaël Guillaume, Erwan Bocher, Pierre Aumond, and David Ecotière. Cartographie du bruit routier à partir de données OpenStreetMap. In *16ème Congrès Français d'Acoustique (CFA2022)*. Marseille, France, 2022. URL: https://hal.science/hal-03848394/.

9. Irune Indacoechea-Vega, Helena Miera-Dominguez, Pedro Lastra-González, and Daniel Castro-Fresno. Life cycle approach for evaluating the environmental and economic viability of low-noise asphalt pavements. *Journal of Cleaner Production*, 466:142785, August 2024. URL: https://www.sciencedirect.com/science/article/pii/S0959652624022339 (visited on 2025-06-10), doi:10.1016/j.jclepro.2024.142785.

10. Arturo Maristany, L.Abadía, D.Moyano, A.Pacharoni, M.Durán, S.Coca, and B.Yelicich. Mapa de ruido de la ciudad de Córdoba. Promeros resultados mediante el uso de software de codigo abierto. In *XVIII Congreso Argentino de Acústica 2023 (AdAA2023)*. Argentina, January 2023.

11. Zahra Nourmohammadi, Tanapon Lilasathapornkit, Mudabber Ashfaq, Ziyuan Gu, and Meead Saberi. Mapping Urban Environmental Performance with Emerging Data Sources: A Case of Urban Greenery and Traf-

fic Noise in Sydney, Australia. *Sustainability*, 13(2):605, January 2021. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/2071-1050/13/2/605 (visited on 2025-06-10), doi:10.3390/su13020605.

12. Ignacio Soto-Molina, Miguel Ausejo Prieto, and Rosa Maria Arce Ruiz. Enhancing noise assessment accuracy: FEM as an advanced complement to CNOSSOS-EU. *Noise & Vibration Worldwide*, 2025. URL: https://journals.sagepub.com/doi/10.1177/09574565251319271, doi:10.1177/09574565251319271.
13. Junta Tagusari. Prediction of road traffic noise in the vicinity of trunk roads in Japan using digital road map platform. *Acoustical Science and Technology*, June 2024. URL: https://www.jstage.jst.go.jp/article/ast/advpub/0/advpub_e24.50/_pdf, doi:10.1250/ast.e24.50.
14. Junta Tagusari. Small-scale noise mapping for arbitrary regions using open source noise prediction module "NoiseModelling" and global databases. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(9):2621–2627, October 2024. doi:10.3397/IN_2024_3212.
15. Zhiyong Wang, Tessio Novack, Yingwei Yan, and Alexander Zipf. Quiet Route Planning for Pedestrians in Traffic Noise Polluted Environments. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7573–7584, December 2021. URL: https://ieeexplore.ieee.org/abstract/document/9139350 (visited on 2025-06-06), doi:10.1109/TITS.2020.3004660.

1. Sacha BACLET, Johan NYGREN, and Romain RUMPLER. Impact of car electrification on urban noise pollution: a microscopic traffic simulation study. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(4):7846–7855, October 2024. doi:10.3397/IN_2024_4014.
2. Sacha Baclet, Siddharth Venkataraman, and Romain Rumpler. A methodology to assess the impact of driving noise from individual vehicles in an urban environment. In *27th International Congress on Sound and Vibration*. 2021. URL: https://kth.diva-portal.org/smash/get/diva2:1641387/FULLTEXT01.pdf.
3. Sacha Baclet, Siddharth Venkataraman, Romain Rumpler, Robin Billsjö, Johannes Horvath, and Per Erik Österlund. From strategic noise maps to receiver-centric noise exposure sensitivity mapping. *Transportation Research Part D: Transport and Environment*, 102:103114, January 2022. URL: https://www.sciencedirect.com/science/article/pii/S1361920921004089 (visited on 2025-06-10), doi:10.1016/j.trd.2021.103114.
4. Arnaud Can, Pierre Aumond, Cécile Bécarie, and Ludovic Leclercq. Approche dynamique pour l'étude de l'emprise spatiale du bruit de trafic routier aux heures de pointe. *Recherche en Transport Sécurité*, 2018(1):1–11, 2018. URL: https://hal.science/hal-02482315/, doi:10.1016/j.retrec.2018.07.001.
5. Arnaud Can, Pierre Aumond, Cécile Bécarie, and Ludovic Leclercq. Dynamic approach for the study of the spatial impact of road traffic noise at peak hours. In *Proceedings of the 23rd International Congress on Acoustics*, –. Aachen, Germany, 2019. URL: https://publications.rwth-aachen.de/record/769733, doi:10.18154/RWTH-CONV-239322.
6. Jasso Espadaler-Clapés, Emmanouil Barmpounakis, and Nikolas Geroliminis. Traffic congestion and noise emissions with detailed vehicle trajectories from UAVs. *Transportation Research Part D: Transport and Environment*, 121:103822, August 2023. URL: https://www.sciencedirect.com/science/article/pii/S1361920923002195 (visited on 2025-06-06), doi:10.1016/j.trd.2023.103822.
7. Leonardo Galassi Luquezi, Valentin Le Bescond, Pierre Aumond, Pascal Gastineau, and Arnaud Can. Current limitations and opportunities for improvements of agent-based transport models for noise exposure assessment. *Journal of Environmental Management*, 368:122129, September 2024. URL: https://www.sciencedirect.com/science/article/pii/S0301479724021157 (visited on 2025-06-10), doi:10.1016/j.jenvman.2024.122129.
8. V. Le Bescond, P. Gastineau, P. Aumond, L. Galassi Luquezi, J. Nygren, L. Soulhac, P. Charvolin-Volta, D. Lejri, and A. Can. Population noise exposure modelling using large scale multi-agent simulation. In *Proceedings of the 10th Convention of the European Acoustics Association Forum Acusticum 2023*, 2659–2662. Turin, Italy, January 2024. European Acoustics Association. URL: https://dael.euracoustics.org/confs/fa2023/data/articles/000680.pdf (visited on 2025-06-06), doi:10.61782/fa.2023.0680.

9. Valentin Le Bescond, Arnaud Can, Pierre Aumond, and Pascal Gastineau. Open-source modeling chain for the dynamic assessment of road traffic noise exposure. *Transportation Research Part D: Transport and Environment*, 94:102793, May 2021. URL: https://www.sciencedirect.com/science/article/pii/S1361920921000973 (visited on 2025-06-06), doi:10.1016/j.trd.2021.102793.

10. Alexandra L. Montenegro, Dulia Melluso, Gianmarco Stasi, Andrea Panci, Matteo Bolognese, Diego Palazzuoli, Mauro Cerchiai, and Gaetano Licitra. An open-source pipeline in noise modelling and noise exposure reduction in a port city. In *Proceedings of the 30th International Congress on Sound and Vibration (ICSV30)*. Amsterdam, Netherlands, 2024. URL: https://iiav.org/content/archives_icsv_last/2024_icsv30/content/papers/papers/full_paper_183_20240415175901591.pdf.

11. Johan Nygren, Valentin Le Bescond, Arnaud Can, Pierre Aumond, Pascal Gastineau, Susann Boij, Romain Rumpler, and Ciarán J. O'Reilly. Agent-specific, activity-based noise impact assessment using noise exposure cost. *Sustainable Cities and Society*, 103:105278, April 2024. URL: https://www.sciencedirect.com/science/article/pii/S2210670724001069 (visited on 2025-06-10), doi:10.1016/j.scs.2024.105278.

12. Antonio PASCALE, Domenico ROSSI, Aurora MASCOLO, Margarida COELHO, and Claudio GUARNACCIA. Road Traffic Noise Levels Estimations by means of Fully Dynamic and Microscopic Approach Compared to CNOSSOS-EU Model. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(7):4767–4778, October 2024. doi:10.3397/IN_2024_3506.

13. G. Quintero, P. Aumond, A. Can, A. Balastegui, and J. Romeu. Statistical requirements for noise mapping based on mobile measurements using bikes. *Applied Acoustics*, 156:271–278, December 2019. URL: https://www.sciencedirect.com/science/article/pii/S0003682X19302087 (visited on 2025-06-06), doi:10.1016/j.apacoust.2019.07.020.

1. Cristian-Gabriel Alionte and Daniel-Constantin Comeaga. Noise assessment of the small-scale wind farm. *E3S Web Conf.*, 112:02011, 2019. Publisher: EDP Sciences. URL: https://www.e3s-conferences.org/articles/e3sconf/abs/2019/38/e3sconf_te-re-rd18_02011/e3sconf_te-re-rd18_02011.html (visited on 2025-06-06), doi:10.1051/e3sconf/201911202011.

2. Pierre Aumond, L. Jacquesson, and Arnaud Can. Probabilistic modeling framework for multisource sound mapping. In *Proceedings of Euronoise*, –. 2018. URL: https://www.euronoise2018.eu/docs/papers/102_Euronoise2018.pdf, doi:10.1016/j.apacoust.2018.04.017.

3. Pierre Aumond, Léo Jacquesson, and Arnaud Can. Probabilistic modeling framework for multisource sound mapping. *Applied Acoustics*, 139:34–43, October 2018. URL: https://www.sciencedirect.com/science/article/pii/S0003682X17311283 (visited on 2025-06-06), doi:10.1016/j.apacoust.2018.04.017.

4. Ingrid Legriffon and Elise Ruaud. Drone fleet noise impact calculation - a methodology. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 1675–1680. Nantes, France, 2024. URL: https://hal.science/hal-04711864v1/, doi:10.3397/IN_2024_3049.

5. J Siliézar, P Aumond, P Chapron, M Péroche, and A Can. Méthode d'évaluation de l'audibilité d'un système d'alerte SAIP. In *16ème Congrès Français d'Acoustique*, 6 p. MARSEILLE, France, April 2022. URL: https://hal.science/hal-03777367 (visited on 2025-06-06).

1. Jérôme Bisson, Sabrina Côté, Henri Curry, William Fauteux, Jérémie Hatier, Othmane Labsir, Léa Rivard, Audrey Guy, Gaëlle Belleau-Magnat, Philippe Apparicio, Johanne Roby, and Olivier ROBIN. Setting up light and noise maps to define the contours of a reserve of darkness and silence in Sherbrooke, Québec, Canada. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(6):5334–5342, October 2024. doi:10.3397/IN_2024_3575.

1. Sacha Baclet, Siddharth Venkataraman, Erik Gomez, and Hamza Bouchouireb. A machine learning-and compressed sensing-based approach for surrogate modelling in environmental acoustics: towards fast evaluation of building façade road traffic noise levels. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 6040–6051. Glasgow, Scotland, 2023. URL: https://www.ingentaconnect.com/content/ince/incecp/2023/00000265/00000001/art00006, doi:10.3397/IN_2022_0898.

2. Antoine Lesieur, Pierre Aumond, Arnaud Can, and Vivien Mallet. Une cartographie du bruit plus rapide et plus précise combinant méta-modélisation et assimilation de données. In *16ème Congrès Français d'Acoustique, CFA2022*. Marseille, France, April 2022. Société Française d'Acoustique and Laboratoire de Mécanique et d'Acoustique. URL: https://hal.science/hal-03848396 (visited on 2025-06-06).
3. Antoine Lesieur, Pierre Aumond, Vivien Mallet, and Arnaud Can. Meta-modeling for urban noise mapping. *J Acoust Soc Am*, 148(6):3671, December 2020. URL: https://pubmed.ncbi.nlm.nih.gov/33379895/, doi:10.1121/10.0002866.
4. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Inverse modeling and joint state-parameter estimation with a noise mapping meta-model. *Journal of the Acoustical Society of America*, 149(6):3961–3974, 2021. URL: https://pubs.aip.org/asa/jasa/article/149/6/3961/1076275, doi:10.1121/10.0004984.
5. Martin Spitznagel and Janis Keuper. Urban Sound Propagation: a Benchmark for 1-Step Generative Modeling of Complex Physical Systems. *arXiv.org*, March 2024. URL: https://arxiv.org/abs/2403.10904v2 (visited on 2025-06-10).
6. Martin Spitznagel, Jan Vaillant, and Janis Keuper. PhysicsGen: Can Generative Models Learn from Images to Predict Complex Physical Relations? In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 11125–11134. 2025. URL: https://openaccess.thecvf.com/content/CVPR2025/html/Spitznagel_PhysicsGen_Can_Generative_Models_Learn_from_Images_to_Predict_Complex_CVPR_2025_paper.html (visited on 2025-06-10).

1. Antoine Lesieur, Pierre Aumond, Arnaud Can, and Vivien Mallet. Une cartographie du bruit plus rapide et plus précise combinant méta-modélisation et assimilation de données. In *16ème Congrès Français d'Acoustique, CFA2022*. Marseille, France, April 2022. Société Française d'Acoustique and Laboratoire de Mécanique et d'Acoustique. URL: https://hal.science/hal-03848396 (visited on 2025-06-06).
2. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Data assimilation for urban noise mapping with a meta-model. *Applied Acoustics*, 178:107938, 2021. URL: https://www.sciencedirect.com/science/article/abs/pii/S0003682X21000311, doi:10.1016/j.apacoust.2021.107938.
3. Antoine Lesieur, Vivien Mallet, Pierre Aumond, and Arnaud Can. Faster and more accurate noise mapping combining meta-modeling and data assimilation. In *Proceedings of Euronoise 2021*. Madeira, Portugal, 2021. URL: https://cense.ifsttar.fr/fileadmin/contributeurs/CENSE/Documents/Euronoise_2021_WP4.pdf.

1. Kerem Ege, Alain Berry, Olivier Robin, and Etienne Parizet. The Sherbrooke-Lyon Research Training Program in Acoustics - Projet Samuel de Champlain 2022-2024. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(10):1746–1752, October 2024. doi:10.3397/IN_2024_3068.

1. Pierre Aumond, Nicolas Fortin, and Arnaud Can. Overview of the NoiseModelling open-source software version 3 and its applications. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 261, 2005–2011. 2020. Issue: 4. URL: https://www.youtube.com/watch?v=V1-niMT9cYE.
2. Jesús López Baeza, Julia L. Sievert, André Landwehr, Jonas Luft, Philipp Preuner, Jürgen Bruns-Berentelg, Ariel Noyman, and Joerg Rainer Noennig. CityScope Platform for Real-Time Analysis and Decision-Support in Urban Design Competitions. *IJEPR*, 10(4):121–137, October 2021. Publisher: IGI Global Scientific Publishing. URL: https://www.igi-global.com/article/cityscope-platform-for-real-time-analysis-and-decision-support-in-urban-design-competitions/www.igi-global.com/article/cityscope-platform-for-real-time-analysis-and-decision-support-in-urban-design-competitions/278826 (visited on 2025-06-10), doi:10.4018/IJEPR.20211001.oa8.
3. Erwan Bocher, Gwenaël Guillaume, Judicaël Picaut, Gwendall Petit, and Nicolas Fortin. NoiseModelling: An Open Source GIS Based Tool to Produce Environmental Noise Maps. *ISPRS International Journal of Geo-Information*, 8(3):130, March 2019. Publisher: MDPI. URL: https://hal.science/hal-02057736 (visited on 2025-06-06), doi:10.3390/ijgi8030130.
4. Junta Tagusari. Dynamic Generation of Noise Maps: Accurate and Efficient Assessment of Noise Exposure. 2023. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4612924, doi:10.2139/ssrn.4612924.

5. Junta Tagusari. H-RISK with NoiseModelling: a QGIS plugin to predict environmental noise and estimate health risks. *Journal of Open Source Software*, 9(93):6023, 2024. URL: https://joss.theoj.org/papers/10.21105/joss.06023, doi:10.21105/joss.06023.
6. Junta Tagusari. Prediction of road traffic noise in the vicinity of trunk roads in Japan using digital road map platform. *Acoustical Science and Technology*, June 2024. URL: https://www.jstage.jst.go.jp/article/ast/advpub/0/advpub_e24.50/_pdf, doi:10.1250/ast.e24.50.
7. Junta Tagusari. Small-scale noise mapping for arbitrary regions using open source noise prediction module "NoiseModelling" and global databases. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 270(9):2621–2627, October 2024. doi:10.3397/IN_2024_3212.

5.6 Community

5.6.1 How to contact?

- 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

5.6.2 Former and/or actual users

NoiseModelling benefits from a large community of users. Some of them are listed below:

Academic labs

- UMRAE (Université Gustave Eiffel - Cerema) ()
- KTH Royal Institue of Technology (Stockholm -)
- Utrecht University ()
- Université de Sherbrooke ()
- Groupe WAVES (Ghent University -)
- Dipartimento di Ingegneria Civile/DICIV (Università Degli Studi di Salerno -)
- HFT Stuttgart Technical University of Applied Science ()
- LETG - CNRS (Nantes,)
- LAboratoire de Géographie et d'Aménagement de Montpellier (LAGAM) - Université Paul Valéry (Montpellier,)
- South China University of Technology ()
- Università di Pisa - ARPAT ()

Private company

- UrbanMetrix (previously Archimethod SA) ()
- OrbiWise ()
- Neovya ()
- Quiet Places Ltd ()
- MobiLysis ()

Teaching

- ENSIM (Le Mans,)
- Université de Sherbrooke ()
- INSA Rennes ()
- Master Acoustique Université du Mans ()
- École Centrale de Nantes ()

Other

- Direction générale de la prévention des risques (DGPR) - ()
- Corps grand-ducal d'incendie et de secours - CGDIS (G.D. Luxembourg)
- Ministerio para la Transición Ecológica y el Reto Demográfico ()

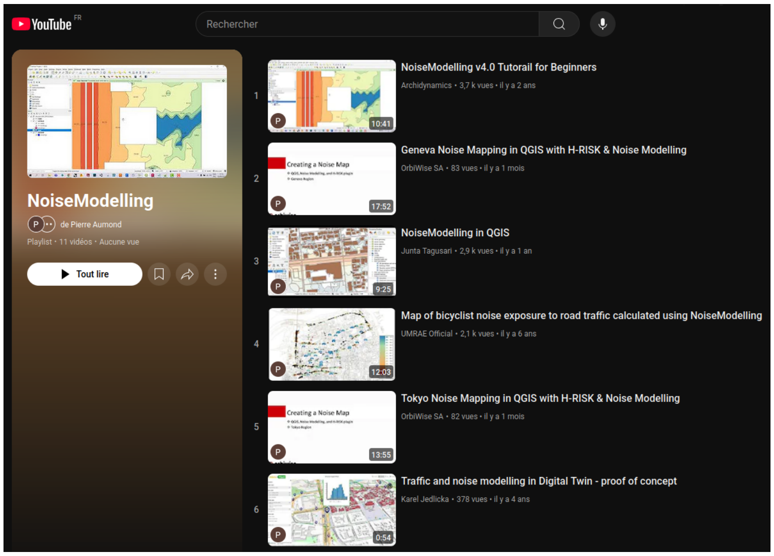
5.6.3 NoiseModelling Days

Every year, the NoiseModelling Days bring together users and contributors to discuss the tool.

To find out more, please visit this page (or this one for previous editions).

5.6.4 NoiseModelling in videos

Click on the thumbnail below to access the NoiseModelling YouTube playlist.



5.7 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.

5.7.1 Table definition

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

- **THE_GEOM** *
  - Description: building's geometry, or thin wall (linestring). It can be in 2D (stuck to the ground) or in 3D (see *Geometry modelling* section below)
  - Type: Geometry (`POLYGON` or `MULTIPOLYGON` or `LINESTRING`)
- **HEIGHT**
  - Description: building's height above the ground *(in meters)*. Optional, used only if the geometry is defined in 2D, ignored otherwise
  - Type: Double
- **POP**

    – Description: number of inhabitant in the building
    – Type: Double

- G
    – Description: Wall absorption value if g is [0, 1] or wall surface impedance ([N.s.m-4] static air flow resistivity of material) if G is [20, 20000] (default is 0.1 if the column G does not exists)
    – Type: Double

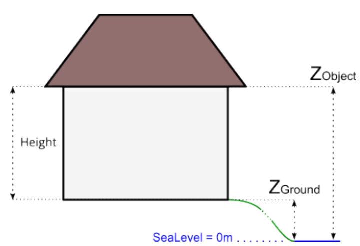
Note: If you want to generate a scene without buildings, create two fictitious buildings, placed in two corners of the scene, and assign them a height of 0 meter.

5.7.2 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), . . .

Warning: If the buildings geometries are defined with a Z coordinate for every point, the `HEIGHT` field is ignored

5.7.3 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

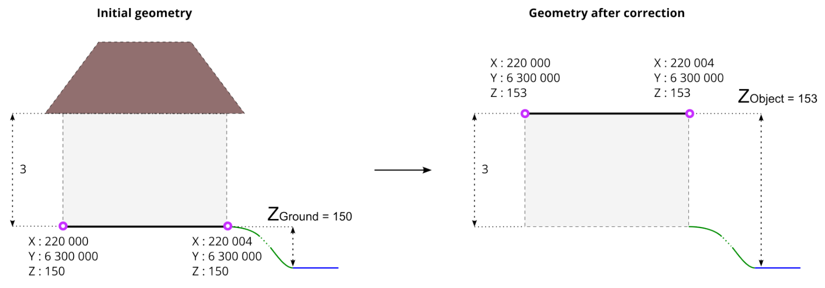
- If `HEIGHT` > 0 then `Zobject` = `HEIGHT`
- If `HEIGHT` = 0 then `Zobject` = 0 and Warning message *"Be carreful, some buildings are 0 meter high"*
- If `HEIGHT` null or < 0 then Error message *"Not possible to determine Z coordinates"*

2. The geometry has a Z coordinate

Warning: If the buildings geometries are defined with a Z coordinate for every point, the `HEIGHT` field is ignored

- **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 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`.



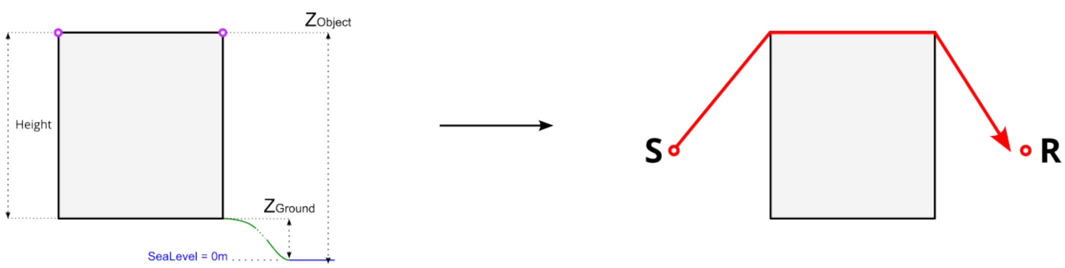
5.7.4 Ray path

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

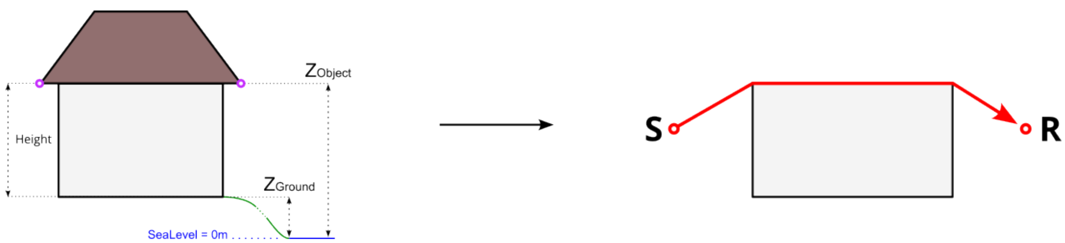
In the 4 examples below,

- 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.

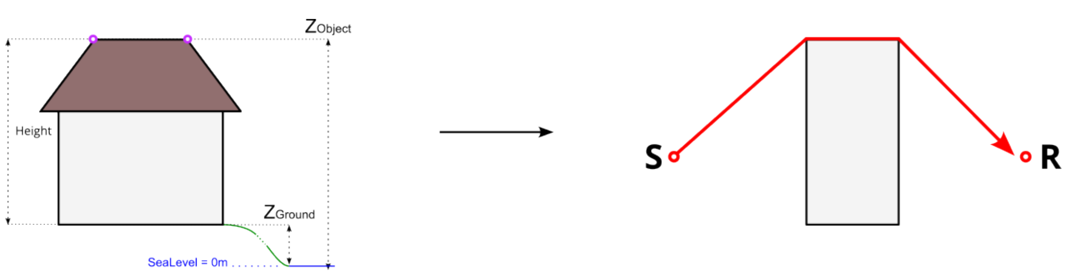
Case 1 : there is no roof



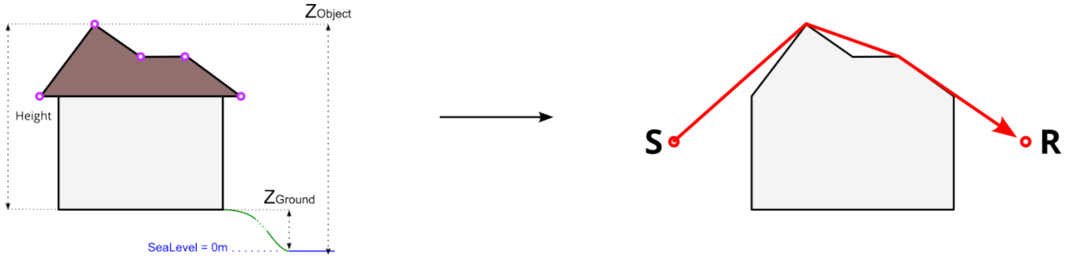
Case 2 : `Zobject` is on the gutter level



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



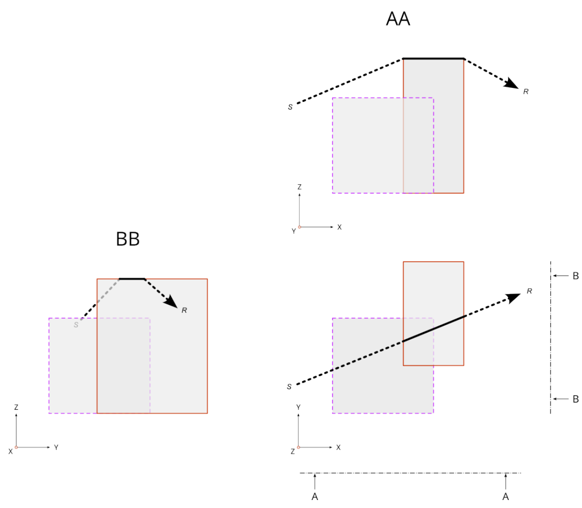
Case 4 : Complex roof shape



5.7.5 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.



5.8 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.

5.8.1 Table SOURCES_GEOM definition

> **Warning:**
>
> - In the list below, the columns noted with * are mandatory
> - This description is only valid *Noise level from source* Block. For the other Blocks, 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
- `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

5.8.2 Table SOURCES_EMISSION definition

If you have custom time periods (ex. `8h00-9h00`). You can place the traffic data into another table with the `PERIOD` column:

The script `Road_Emission_from_Traffic` can convert this traffic into the `LW_ROADS` table used for emission (dB for each octave bands) as an input for the `Noise_level_from_source` script.
Being able to see the noise emission of each roads is very useful for validation and verification of the input data before doing the propagation step.

- **`IDSOURCE`** *
  - Description: An identifier linked to the primary key of the `SOURCES_GEOM` table.
  - Type: Integer
- **`PERIOD`** *
  - Description: Identifier of the time. ex. `8h00-9h00`
  - Type: String
- **`LV`**
  - Description: Hourly average light vehicle count
  - Type: Double
- **`MV`**
  - 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`**
  - Description: Hourly average heavy duty vehicles, touring cars, buses, with three or more axles count
  - Type: Double
- **`WAV`**
  - Description: Hourly average mopeds, tricycles or quads 50 cc count
  - Type: Double
- **`WBV`**
  - Description: Hourly average motorcycles, tricycles or quads > 50 cc count
  - Type: Double
- **`LV_SPD`**
  - Description: Hourly average light vehicle speed *(km/h)*

- Type: Double

- `MV_SPD`
  - Description: Hourly average medium heavy vehicles speed *(km/h)*
  - Type: Double
- `HGV_SPD`
  - Description: Hourly average heavy duty vehicles speed *(km/h)*
  - Type: Double
- `WAV_SPD`
  - Description: Hourly average mopeds, tricycles or quads 50 cc speed *(km/h)*
  - Type: Double
- `WBV_SPD`
  - 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
- `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

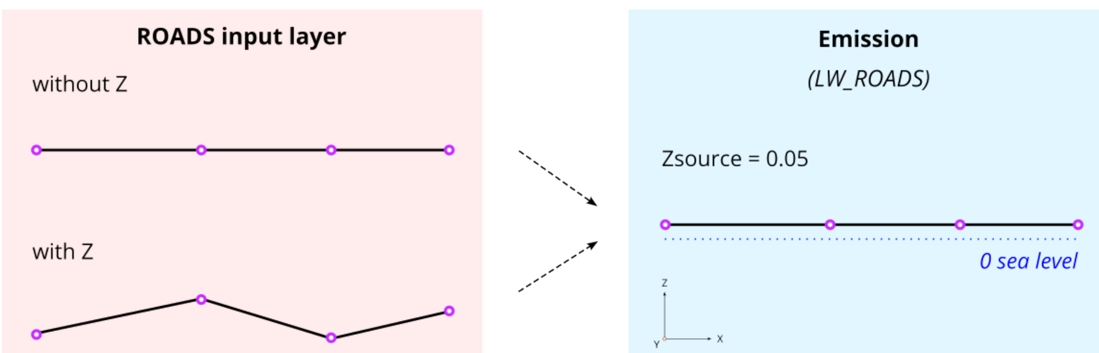
5.8.3 Geometry modelling

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`.

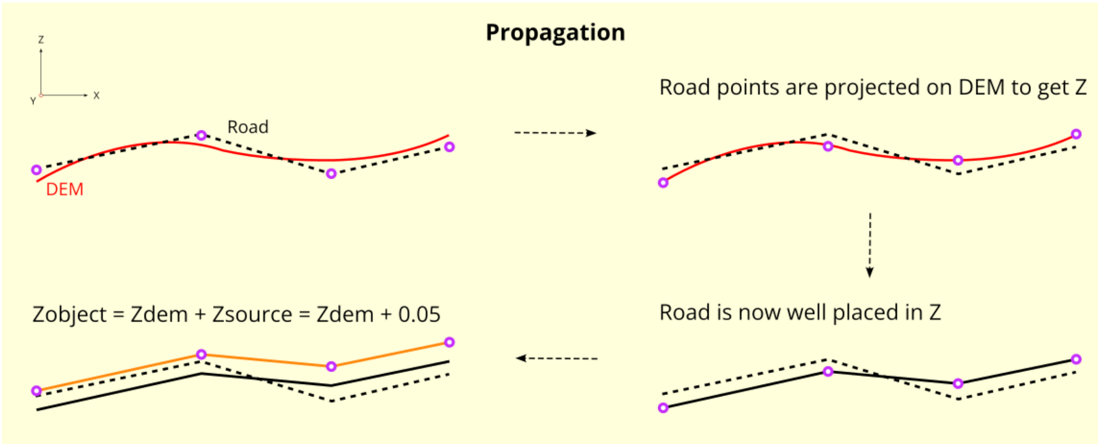


Warning: Whether you have Z values, the emission layer must be at an altitude of 5cm (above sea level) : `Zsource` = 0.05

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).



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.

5.9 Sound source

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 `SOURCES_GEOM` and `SOURCES_EMISSION`, expected by the `Noise_level_from_source` processing block.

The goal is to define any sound source with a custom noise spectrum level.

5.9.1 SOURCES_GEOM Table definition

- **PK** *
  - Description: An identifier (must be a PRIMARY KEY). The field name is not important, it could be IDSOURCE.
  - Type: Integer
- **THE_GEOM** *
  - Description: Geometry of the source (`LINESTRING` , `MULTILINESTRING` or `POINT`). The geometry **must** have a Z attribute.
  - Type: Geometry

Warning:

- `Zobject` = `Zdem + Zsource`
- If there is no DEM, the altitude will be equal to geometry Z attribute

Note: For backward compatibility in this table you can define here the noise level for Day Evening and Night, for 500 Hz the expected field is `HZD500` `HZE500` `HZN500`. NoiseModelling will output the level for this periods and also compute the global `DEN` period levels according to standard.

If you provide only the geometry NoiseModelling will compute only the attenuation on the output table `RECEIVERS_LEVEL`.

To provide the emission level you must define an emission table for each time period.

5.9.2 SOURCES_EMISSION Table definition

- **IDSOURCE** *
  - Description: An identifier linked to the primary key of the `SOURCES_GEOM` table.
  - Type: Integer
- **PERIOD** *
  - Description: Identifier of the time. ex. *8h00-9h00*
  - Type: String
- **HZ63**
  - Description: emission levels in dB for 63 Hz
  - Type: Double
- **HZ125**
  - Description: emission levels in dB for 125 Hz
  - Type: Double
- **HZ250**
  - Description: emission levels in dB for 250 Hz

  - Type: Double

- **HZ500**
  - Description: emission levels in dB for 500 Hz
  - Type: Double
- **HZ1000**
  - Description: emission levels in dB for 1000 Hz
  - Type: Double
- **HZ2000**
  - Description: emission levels in dB for 2000 Hz
  - Type: Double
- **HZ4000**
  - Description: emission levels in dB for 4000 Hz
  - Type: Double
- **HZ8000**
  - Description: emission levels in dB for 8000 Hz
  - Type: Double

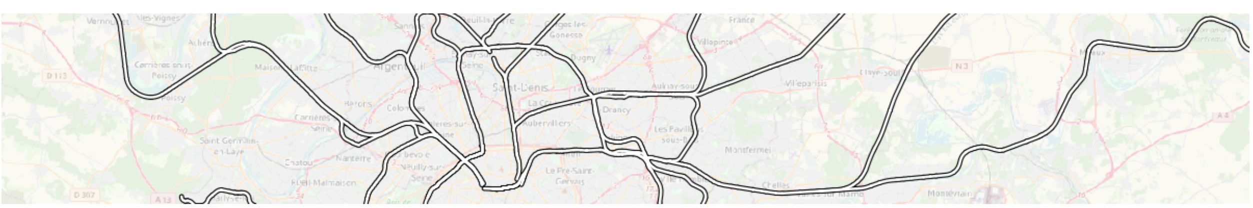
Note: You can define partially the bands. We defined here only octave bands but NoiseModelling is supporting third-octaves `HZ50, HZ63, HZ80, HZ100, HZ125, HZ160, HZ200, HZ250, HZ315, HZ400, HZ500, HZ630, HZ800, HZ1000, HZ1250, HZ1600, HZ2000, HZ2500, HZ3150, HZ4000, HZ5000, HZ6300, HZ8000, HZ10000`.

5.10 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

5.10.1 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, listed in the RailwayEmissionCnossos.json file**
    - `SNCF1` = Mono-bloc sleeper on soft rail pad
    - `SNCF2` = Mono-bloc sleeper on medium rail pad
    - `SNCF3` = Mono-bloc sleeper on stiff rail pad
    - `SNCF4` = Bi-bloc sleeper on soft rail pad
    - `SNCF5` = Bi-bloc sleeper on medium rail pad
    - `SNCF6` = Bi-bloc sleeper on stiff rail pad
    - `SNCF7` = Wooden sleeper (Traverse en bois)
  - Type: Varchar
- **`ROUGHNESS`**
  - **Description: Rail roughness identifier, listed in the RailwayEmissionCnossos.json file**
    - `SNCF1` = Classic lines
    - `SNCF2` = TGV (for France) lines
  - Type: Varchar
- **`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).

5.10.2 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

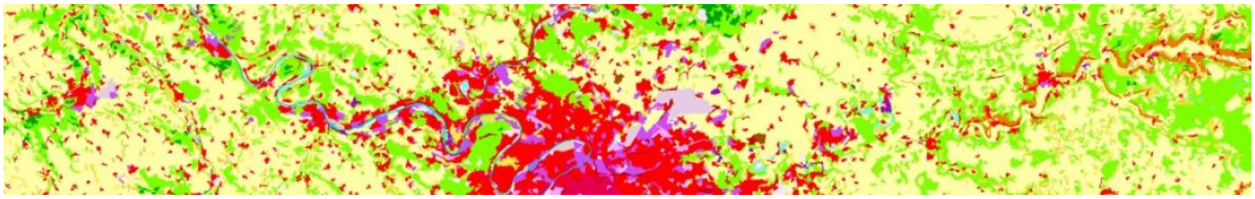
- **TRAINTYPE** *
  - Description: Type of vehicle, listed in the RailwayTrainsets.json file *(mainly for french SNCF)*
  - Type: Varchar
- **TRAINSPD** *
  - Description: Maximum train speed *(in km/h)*
  - Type: Double
- **TDAY**
  - Description: Hourly average train count, during the day (6-18h)
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

5.11 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.



5.11.1 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

5.11.2 Topology

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

5.12 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` script which will format your DEM in the right format

5.12.1 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

5.12.2 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.

5.13 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.

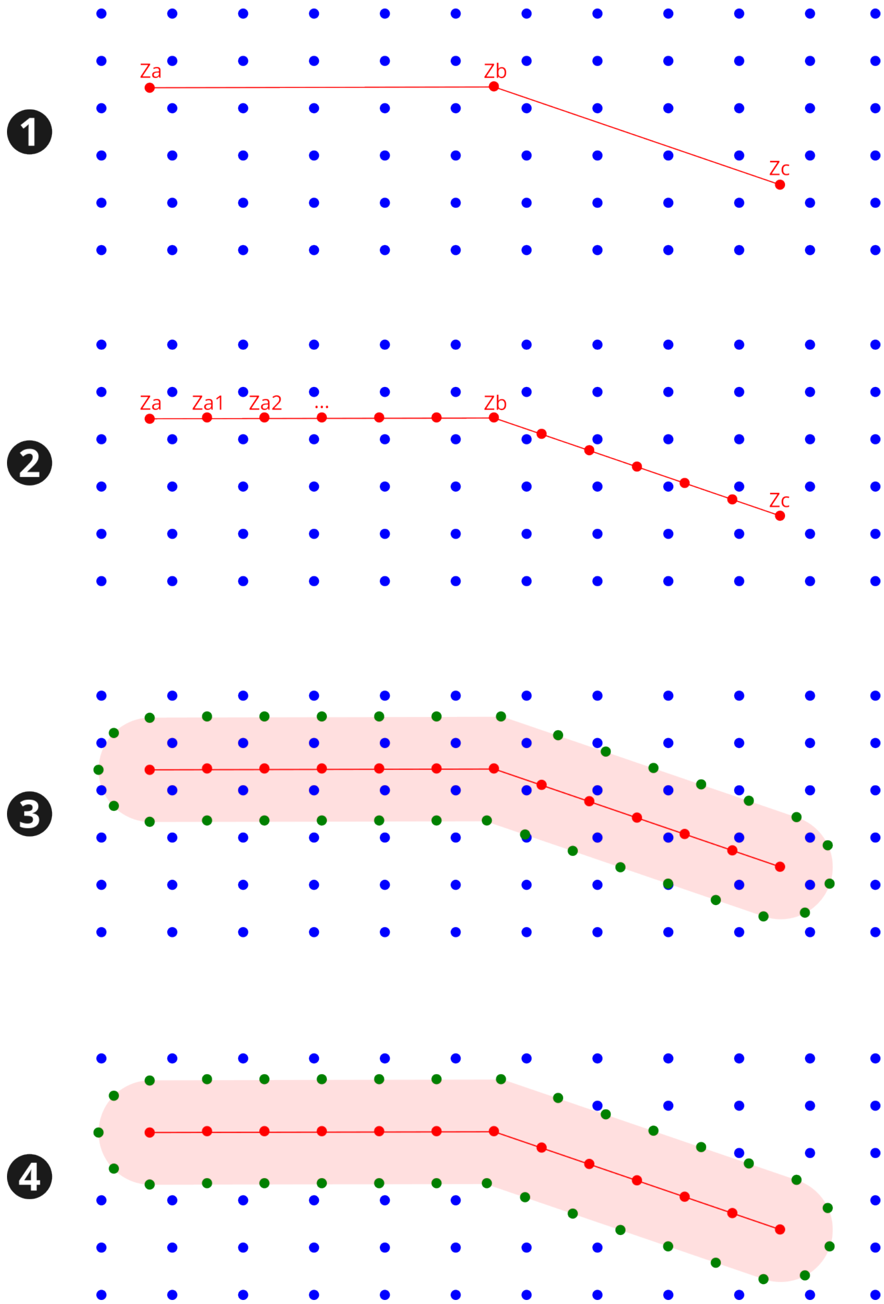
The other tables are accessible via the left menu in the `Input tables` section.

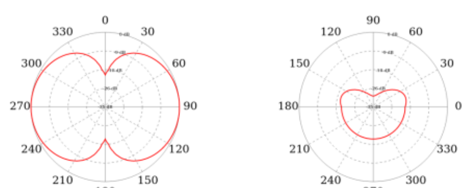
Note: If you want to see how to use this table, have a look to the tutorial "*Noise Map from Point Source - GUI*" , in the section `Step 5 (bonus): Change the directivity`

5.13.1 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
  - Type: Double


1
Za
Zb
Zc
2
Za
Za1
Za2
...
Zb
Zc
3
4


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

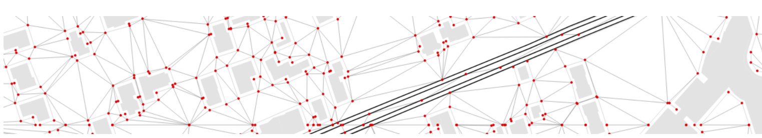
- **HZ63**
  - Description: attenuation levels in dB for 63 Hz
  - Type: Double
- **HZ125**
  - Description: attenuation levels in dB for 125 Hz
  - Type: Double
- **HZ250**
  - Description: attenuation levels in dB for 250 Hz
  - Type: Double
- **HZ500**
  - Description: attenuation levels in dB for 500 Hz
  - Type: Double
- **HZ1000**
  - Description: attenuation levels in dB for 1000 Hz
  - Type: Double
- **HZ2000**
  - Description: attenuation levels in dB for 2000 Hz
  - Type: Double
- **HZ4000**
  - Description: attenuation levels in dB for 4000 Hz
  - Type: Double
- **HZ8000**
  - Description: attenuation levels in dB for 8000 Hz
  - Type: Double

5.14 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.



5.14.1 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

5.14.2 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).

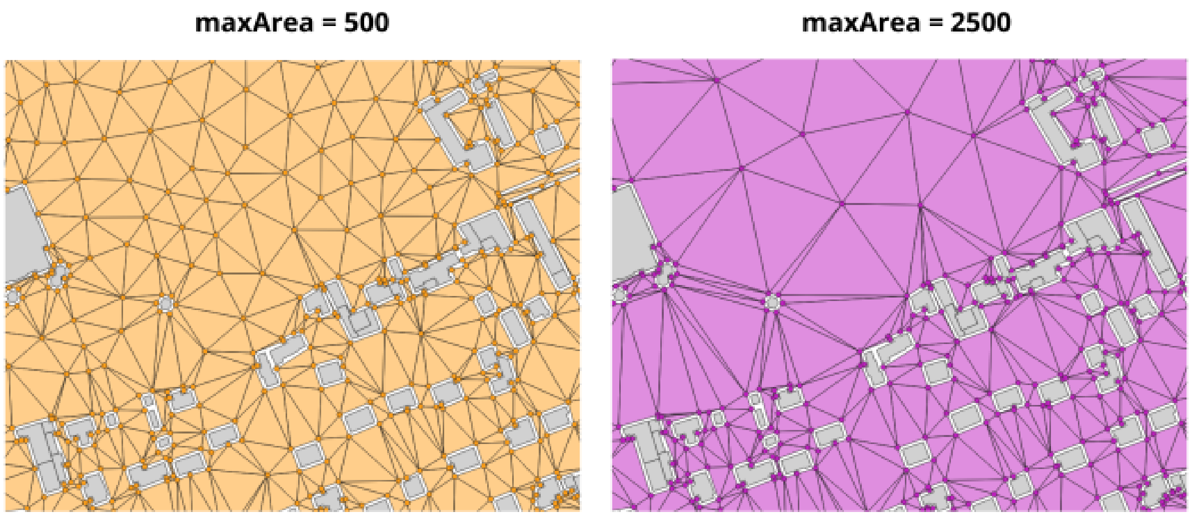
These parameters can be mandatory or optional. When necessary, we indicates the default values and those we recommend (from an acoustic point of view).

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

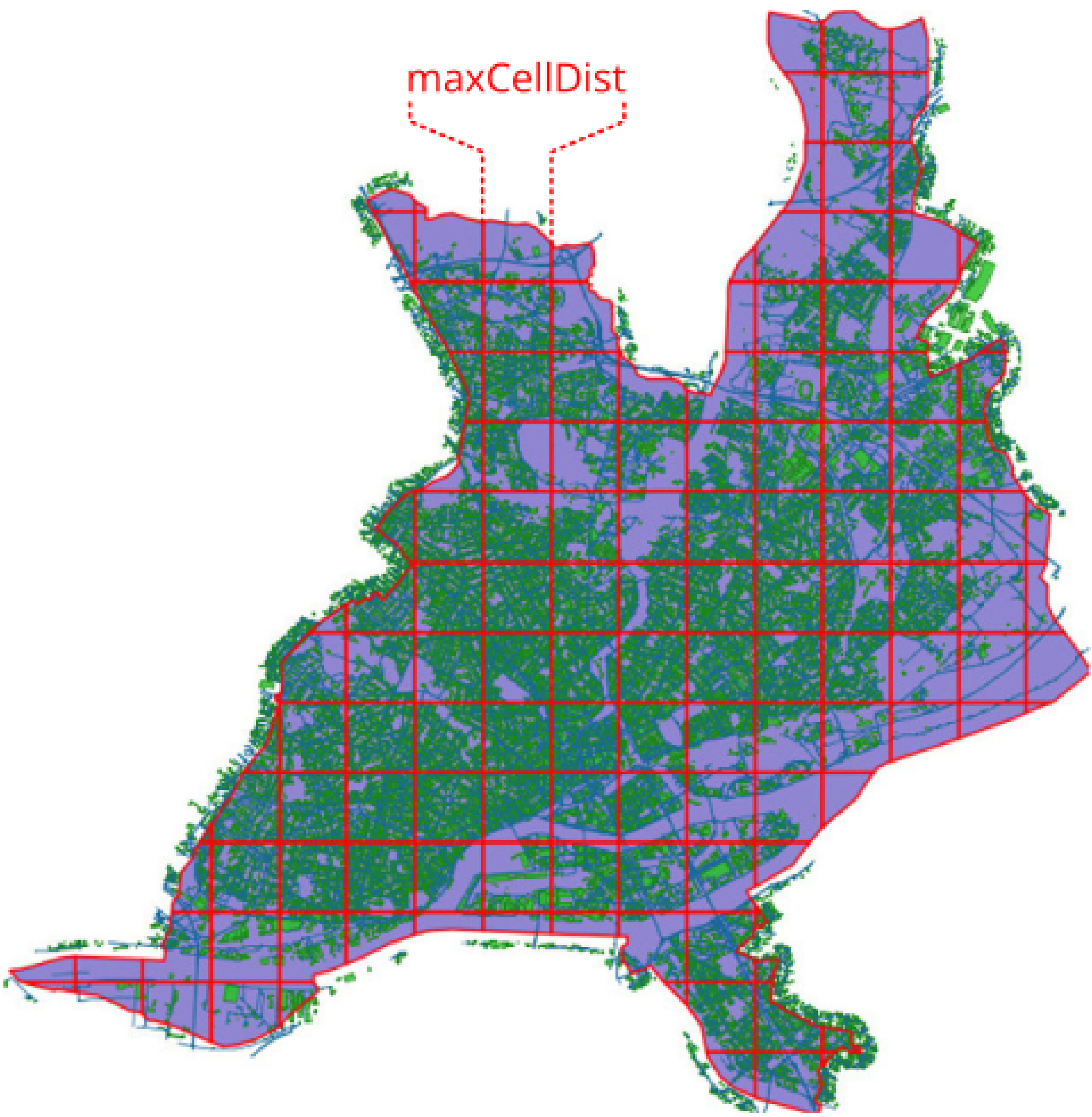
- Parameter name: `height`
- Description: Receiver height relative to the ground (meters)
- Type: Double
- Default value: `4`
- Recommanded value:

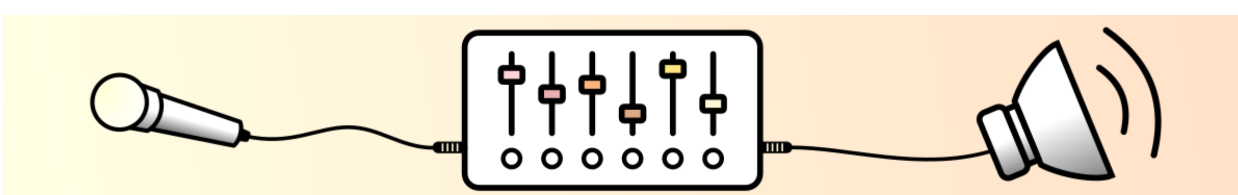
5.15 Acoustic parameters

In the different NoiseModelling Blocks, 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).

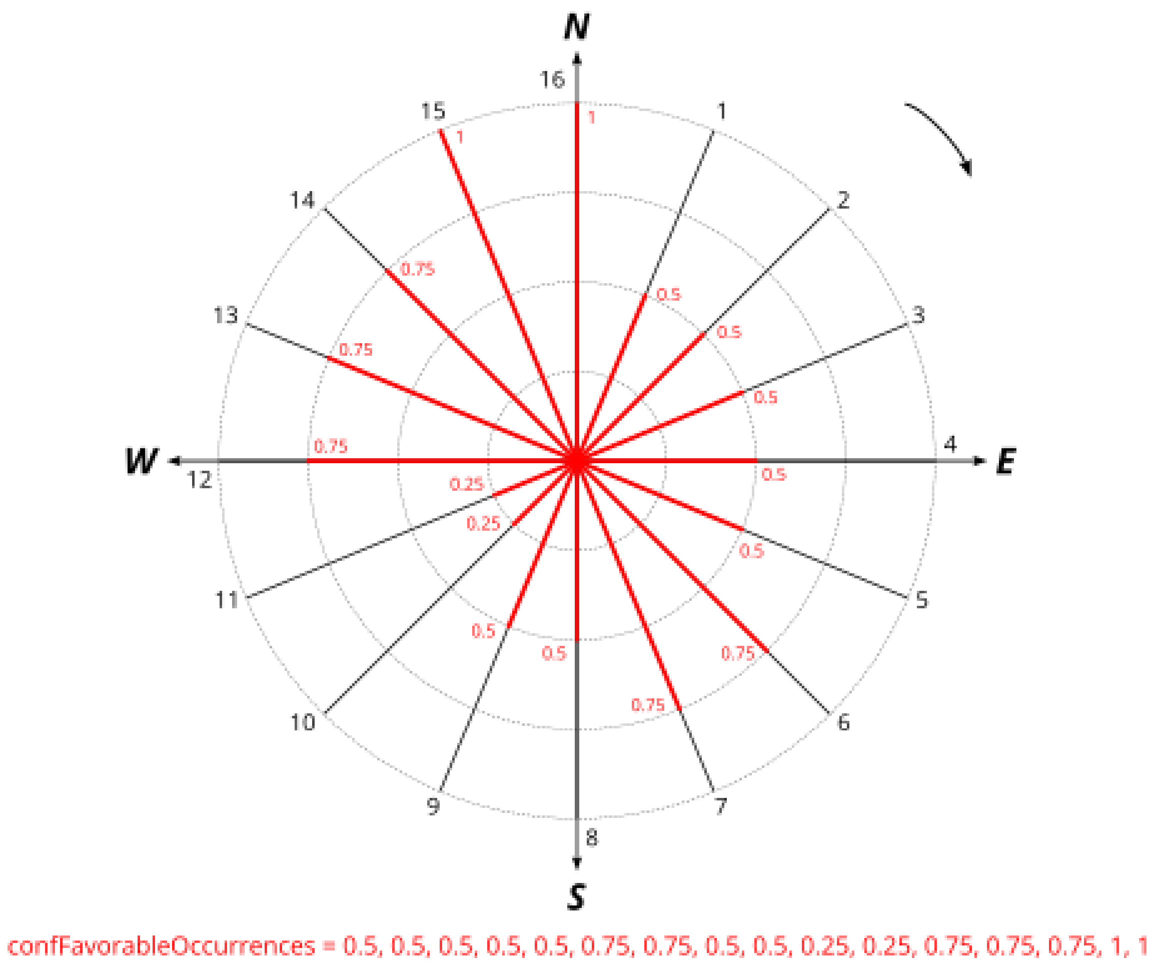
The following parameters may be found in the scripts dealing with noise emission or propagation (*e.g* *Noise level from source*, …)


maxCellDist



5.15.1 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`
- Recommended value:



5.15.2 Relative humidity

- Parameter name: `confHumidity`
- Description: Humidity for noise propagation (%) [0,100]
- Type: Double
- Default value: `70`
- Recommended value: depends on the average conditions at the location where you perform the simulation

5.15.3 Air temperature

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

5.15.4 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`
- Recommended value: `1` or `2`

5.15.5 Diffraction on horizontal edges

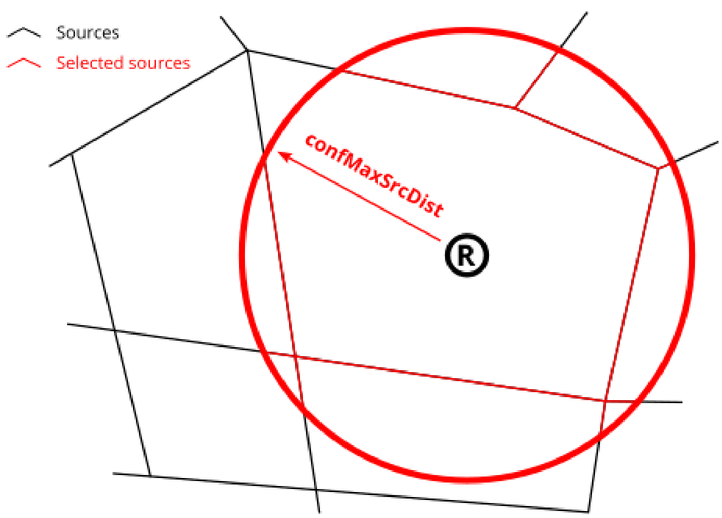
- Parameter name: `confDiffHorizontal`
- Description: Compute or not the diffraction on horizontal edges
- Type: Boolean
- Default value: `False`
- Recommended value: `True`

5.15.6 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`
- Recommended value:

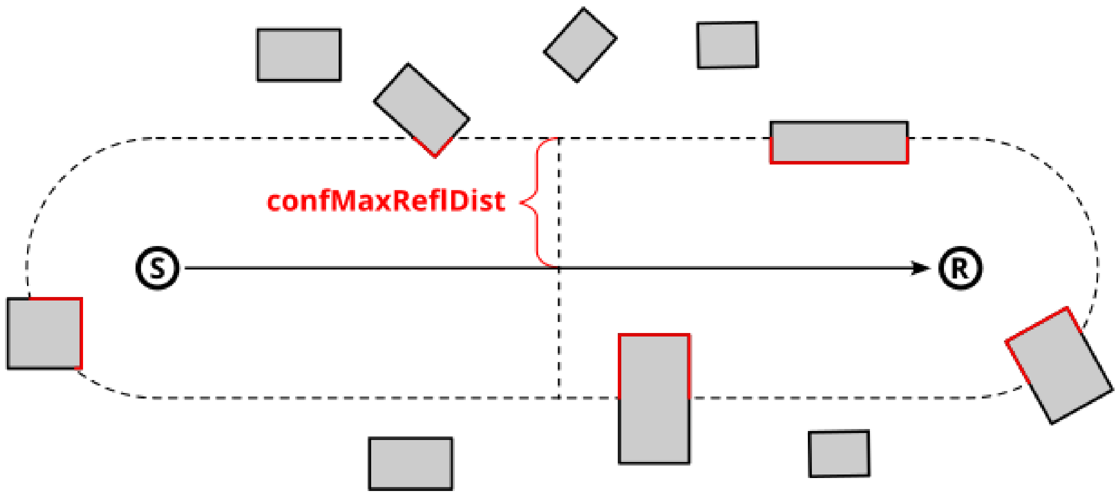
5.15.7 Maximum source-receiver distance

- Parameter name: `confMaxSrcDist`
- Description: Maximum distance between source and receiver (meters)
- Type: Double
- Default value: `150`
- Recommended value: Between `500` and `800`



5.15.8 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`
- Recommended value: Between `350` and `800`



5.15.9 Ignore reflections close to the receiver wall

- Parameter name: `confMinWallReflDist`
- Description: Optional maximum receiver-to-wall distance (meters) below which reflection cut profiles are ignored. When a receiver is located very close to a reflective wall, the reflected path may be physically unreliable or negligible. Setting this parameter filters out such reflection paths. Use `0` to keep all reflections (filter disabled).
- Type: Double
- Default value: `0` (disabled)
- Recommended value: Set the distance you wish; classically below `2` m. This parameter is typically used when assessing noise impact on people inside buildings, in order to avoid the last-reflection effect (approximately +3 dB).

5.15.10 Wall absorption coefficient

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

5.15.11 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`
- Recommended value:

5.15.12 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))
- Recommended value: `0`

5.15.13 Max Error (dB)

- Parameter name: `confMaxError`
- Description: Threshold for excluding negligible sound sources in calculations. Default value: **0.1**. This parameter is ignored if no emission level is specified or if you set it to 0 dB. This parameter have a great impact on computation time.
- Type: Double
- Default value: `0.1` dB
- Recommended value: `0.1` dB

Maximum error algorithm explanation

In order to reduce computation time, we can ignore far away sound source that will not change the noise level at the receiver location.

Before looking for propagation path, all sound sources are fetched in the radius of `confMaxSrcDist` then sorted by distance from the receiver position.

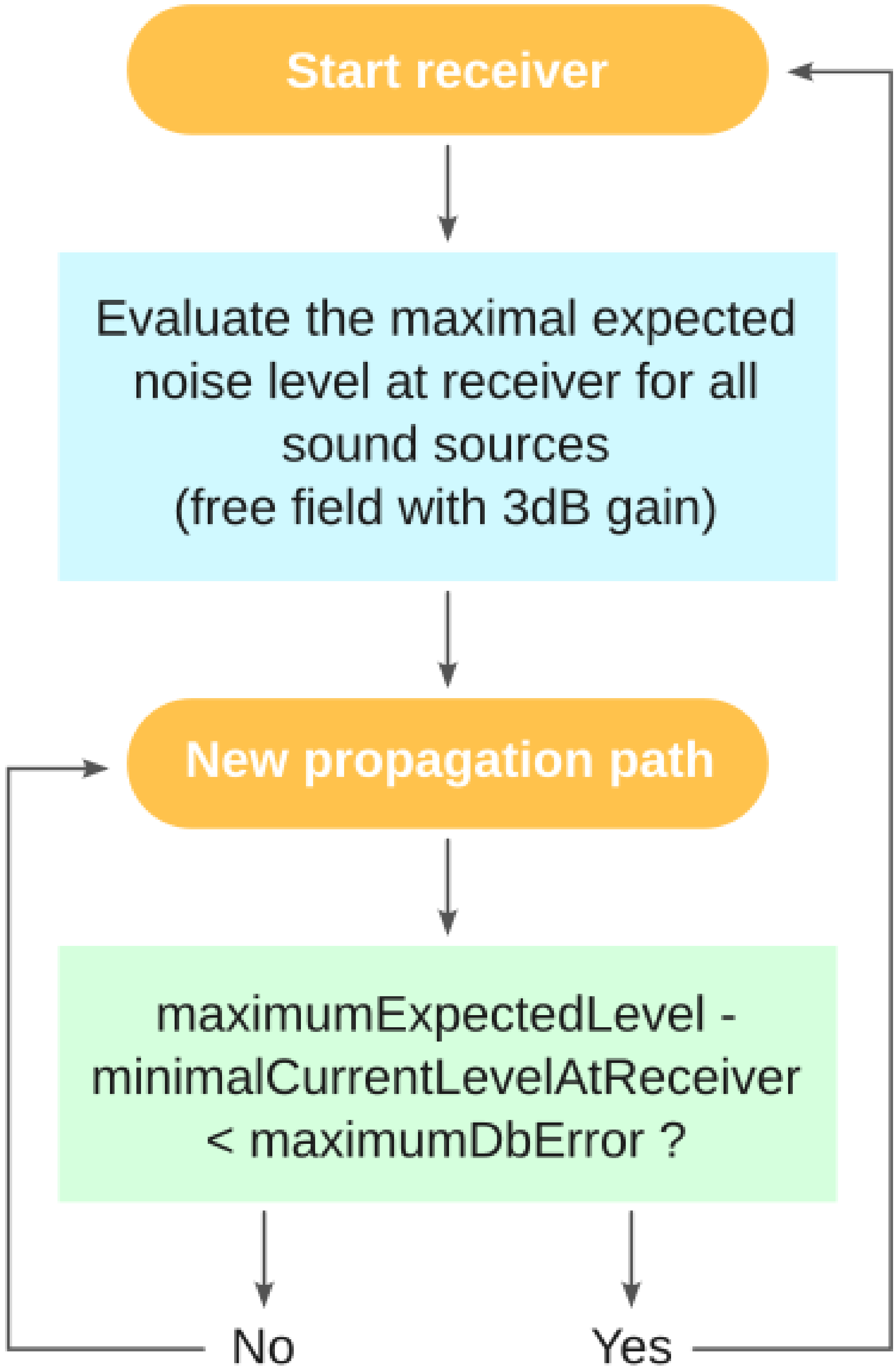
After each propagation path is found, we evaluate if the difference between the accumulated noise level of all previous sound sources with the expected maximum noise level at the receiver with all sound sources is greater than the maximum error parameter.

If the next sound sources contribution is negligible we move to the next receiver point.

Minimal and maximal values are over all emission periods specified on the sound sources. The maximal expected noise level value is updated after each sound source is processed.

5.16 Get Started - GUI

Below we present a simple example to help you discover NoiseModelling through its Graphical User Interface (GUI).


Start receiver
Evaluate the maximal expected noise level at receiver for all sound sources (free field with 3dB gain)
New propagation path
maximumExpectedLevel - minimalCurrentLevelAtReceiver < maximumDbError ?
No
Yes

5.16.1 Step 1: Open NoiseModelling

See *Step 3: Start NoiseModelling GUI* in the *Installation guide*.

5.16.2 Step 2: Load input files

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

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

In the `resources/` sub-folder of the NoiseModelling installation, you will find all the data that will be used in the tutorials.

You will import these layers into your database using the `Import File` Blocks.

- Drag the `Import File` Block into the Builder window
- Select the `Path of the input File` box and enter `resources/buildings.shp` in the `pathFile` field *(on the right-side column)*
- Then click on `Run Process` after selecting one of the input/output boxes of your process

Repeat this operation for the 4 other files:

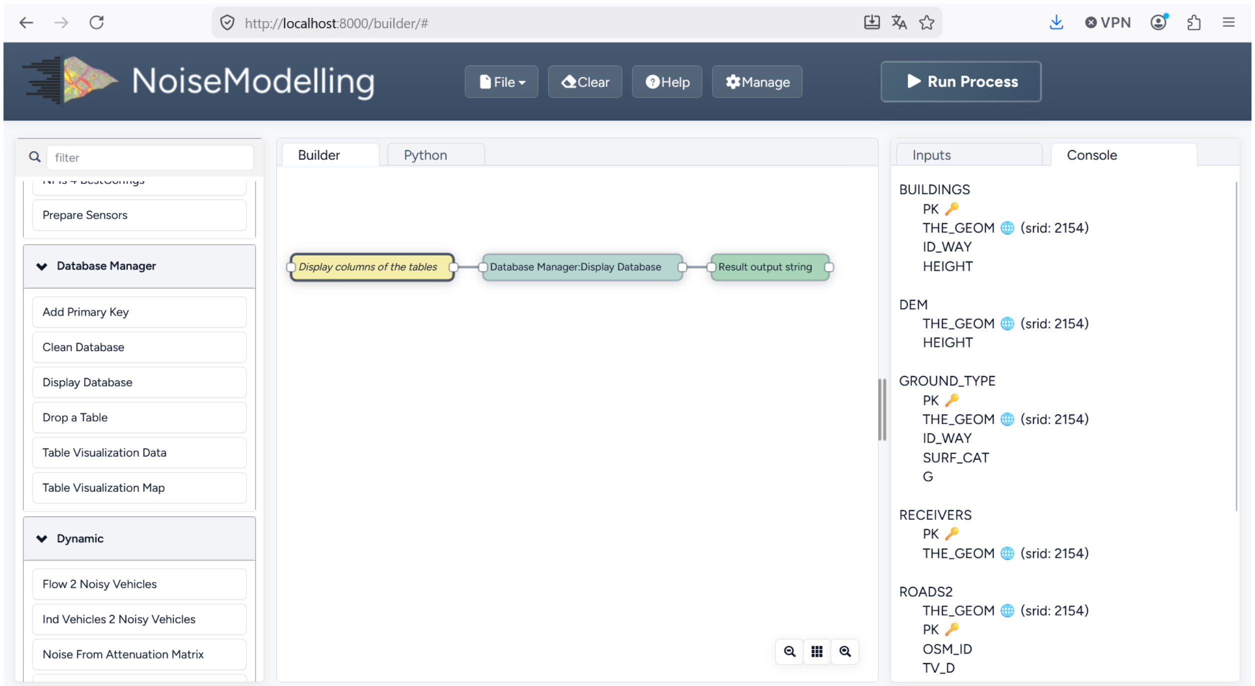
- `resources/ground_type.shp`
- `resources/receivers.shp`
- `resources/ROADS2.shp`
- `resources/dem.geojson`

Files are uploaded to the database when the Console window displays the name of the layer.

Note:

- If you get the message `Error opening database`, please refer to the note in *Step 2: Download NoiseModelling* in the *Installation guide*.
- The process is supposed to be quick (<5 sec.). In case of a timeout, try restarting NoiseModelling (see *Step 3: Start NoiseModelling GUI* in the *Installation guide*).
- 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 a solid border when they are ready to run
- Read the *Builder* page for more information

Once done, you can check whether the tables were correctly imported into the database. To do so, drag/drop and execute the `Display_Database` Block (in the "Database_Manager" section). You should see on the right panel the table list (and their columns if you checked the option in the `Display columns of the tables` Block).



5.16.3 Step 3: Convert road traffic into noise emission sources lines

The first step is to convert the traffic information on the roads to noise levels (vehicles per hour to an average value in dB)

Drag & Drop the `Road Emission from Traffic` Block into the Builder window.

Enter the name of the corresponding table in your database:

- Roads table name: `ROADS2` This table contains the road geometries with traffic data for day, evening and night

When ready, you can press `Run Process`.

As a result, the table `LW_ROADS` will be created in your database. This table contain the noise emission of your roads. The next step will run a simulation of the noise propagation to the receivers position.

5.16.4 Step 4: Run Calculation

To run the calculation, drag the `Noise_level_from_sources` Block into the Builder window.

Then, select the orange Blocks and enter the name of the corresponding table in your database:

- Building table name: `BUILDINGS`
- Source table name: `LW_ROADS` This table contains the road geometries with the noise emission values for day, evening and night
- Receivers table name: `RECEIVERS` Locations where noise levels are evaluated
- DEM table name: `DEM` Digital elevation model
- Ground absorption table: `GROUND_TYPE` Nature of the ground
- Diffraction on horizontal edges: check it (sound propagation goes over buildings)
- Maximum source-receiver distance: set `2000` meters (do not look for sound sources further than 2 km)

- Order of reflection: set `0` to disable it (faster but less accurate)

The beige Blocks correspond to optional 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 table `RECEIVERS_LEVEL` will be created in your database. This table corresponds to the noise levels computed at receiver points. The column PERIOD corresponds to the 4 different periods of the day (D, E, N and DEN).

5.16.5 Step 5: Export (& see) the results

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

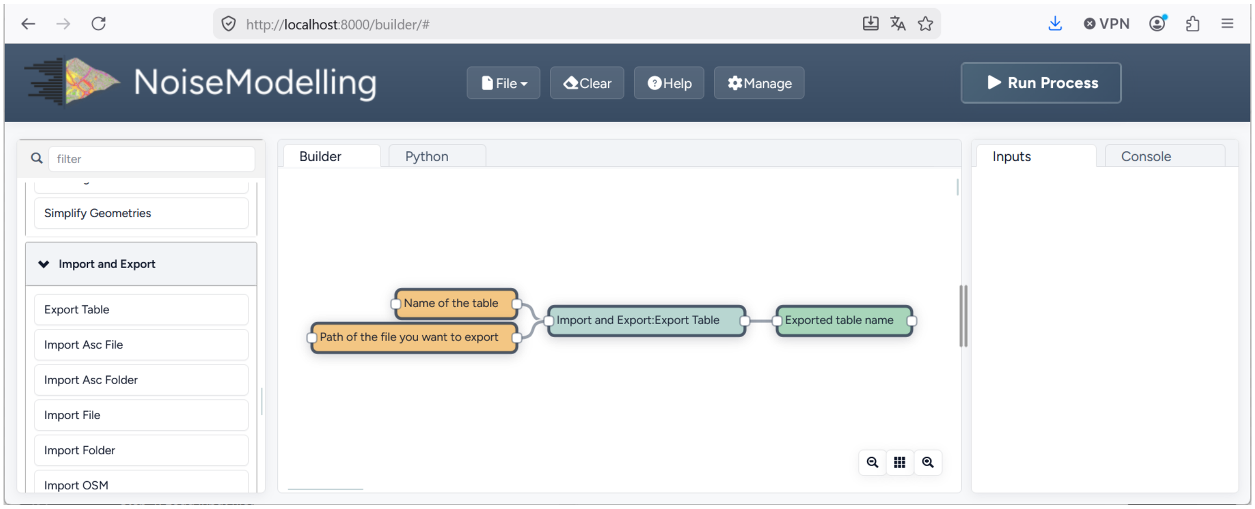
Warning: Don't forget to add the file extension (*e.g.* `c:/home/receivers_level.geojson` or `c:/home/receivers_level.shp`). (Read more info about file extensions here: *Tutorials - FAQ*)

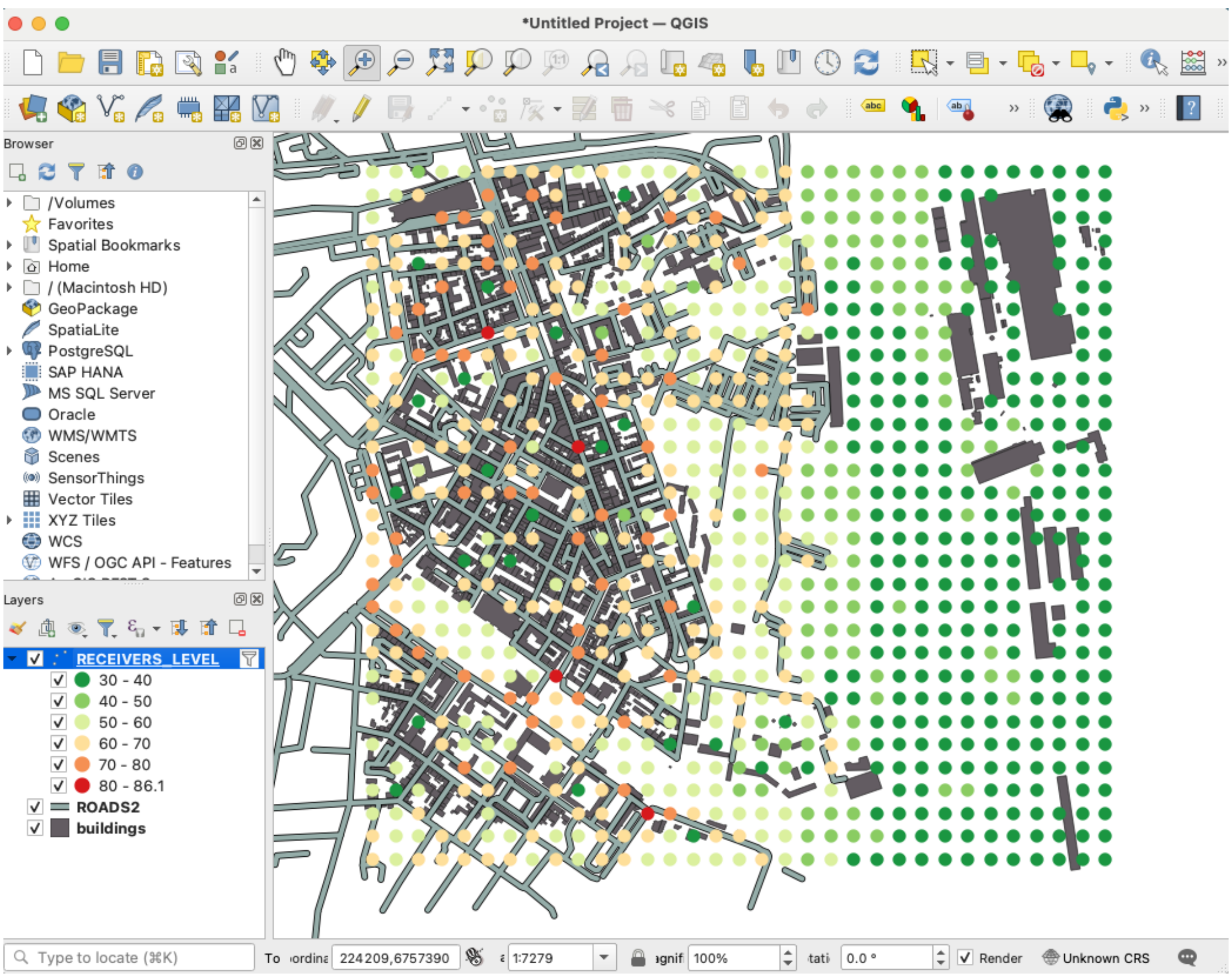
For example, you can export the tables in `.shp` format. This format can be read with most GIS tools such as the free and open-source QGIS and SAGA software.

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

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

To display the result for a specific period, filter the rendering by the field PERIOD in QGIS.


http://localhost:8000/builder/#
NoiseModelling
File
Clear
Help
Manage
Run Process
Builder
Python
Inputs
Console
Simplify Geometries
Import and Export
Export Table
Import Asc File
Import Asc Folder
Import File
Import Folder
Import OSM
Name of the table
Path of the file you want to export
Import and Export:Export Table
Exported table name


*Untitled Project — QGIS
Browser
/Volumes
Favorites
Spatial Bookmarks
Home
/ (Macintosh HD)
GeoPackage
SpatiaLite
PostgreSQL
SAP HANA
MS SQL Server
Oracle
WMS/WMTS
Scenes
SensorThings
Vector Tiles
XYZ Tiles
WCS
WFS / OGC API - Features
Layers
RECEIVERS_LEVEL
30 - 40
40 - 50
50 - 60
60 - 70
70 - 80
80 - 86.1
ROADS2
buildings
Type to locate (⌘K)
224209,6757390
1:7279
100%
0.0 °
Render
Unknown CRS

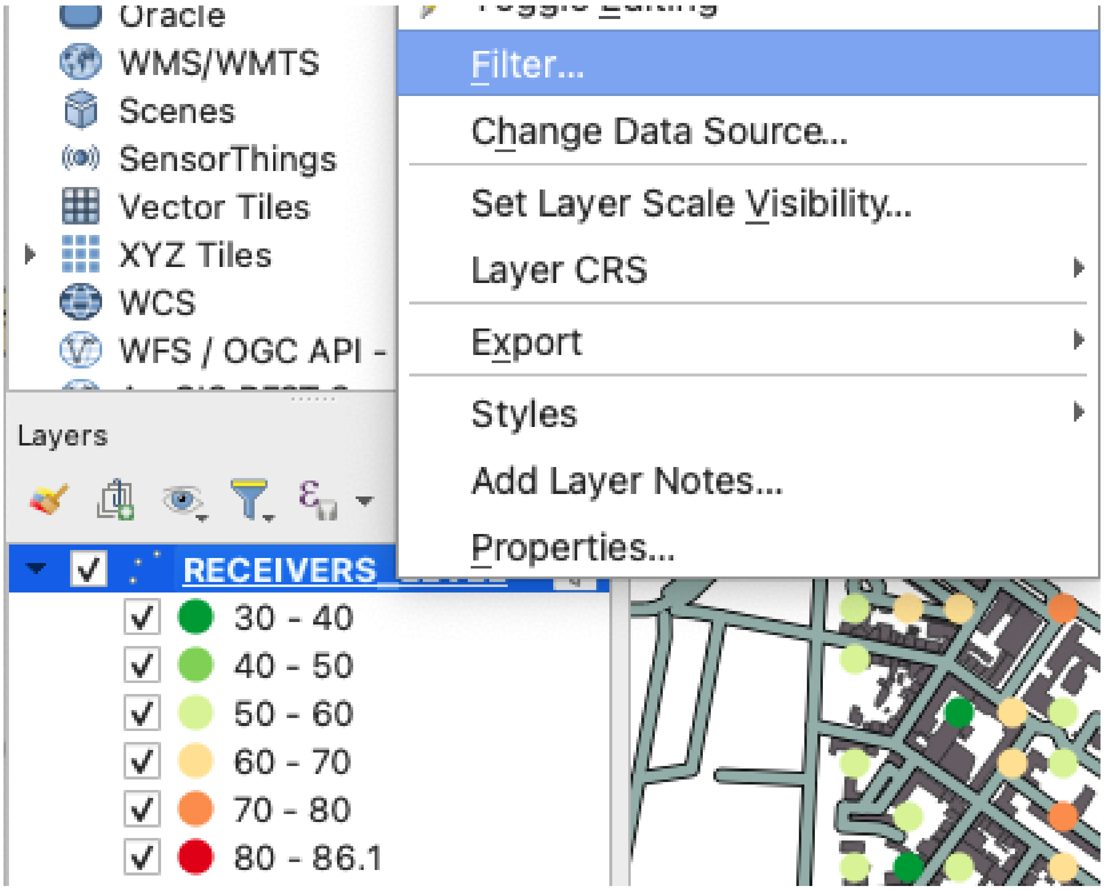


Fig. 2: Popup menu

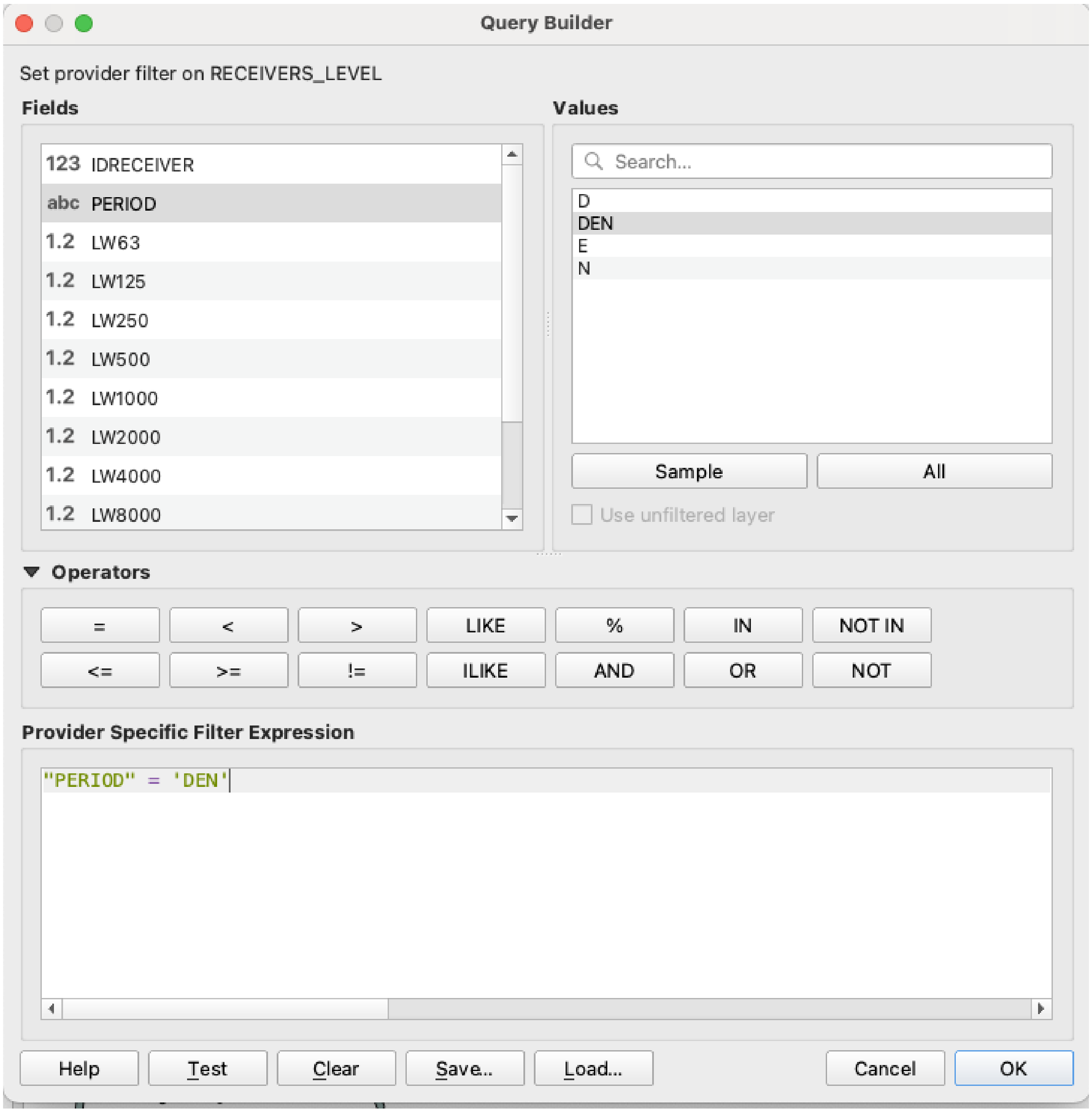


Fig. 3: Filter window

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

5.16.6 Step 6: Know the possibilities

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

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

5.17 Noise Map from Point Source - GUI

In this tutorial, we are going to produce a noise map, based on a unique source point. The exercise 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

5.17.1 Step 1: Create the source point

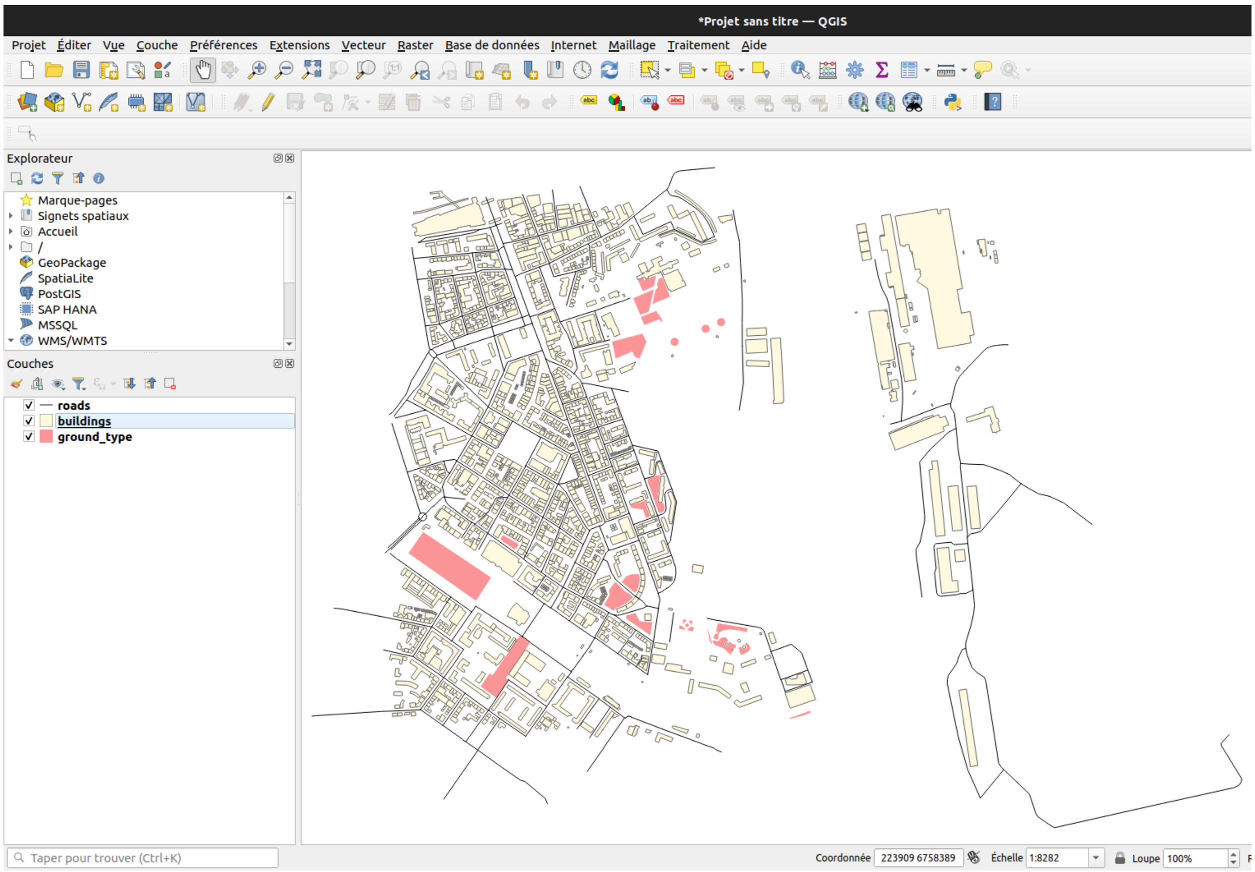
To create the source point, we will use the free and opensource GIS software QGIS.

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/resources/`). 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:


*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%

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

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

- `Name` : HZD500 . Source noise level during the day (D) at a frequency of 500 Hz
- `Type` : Decimal number
- `Length` : 5
- `Precision` : 2

You should have something like this

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 edition 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.

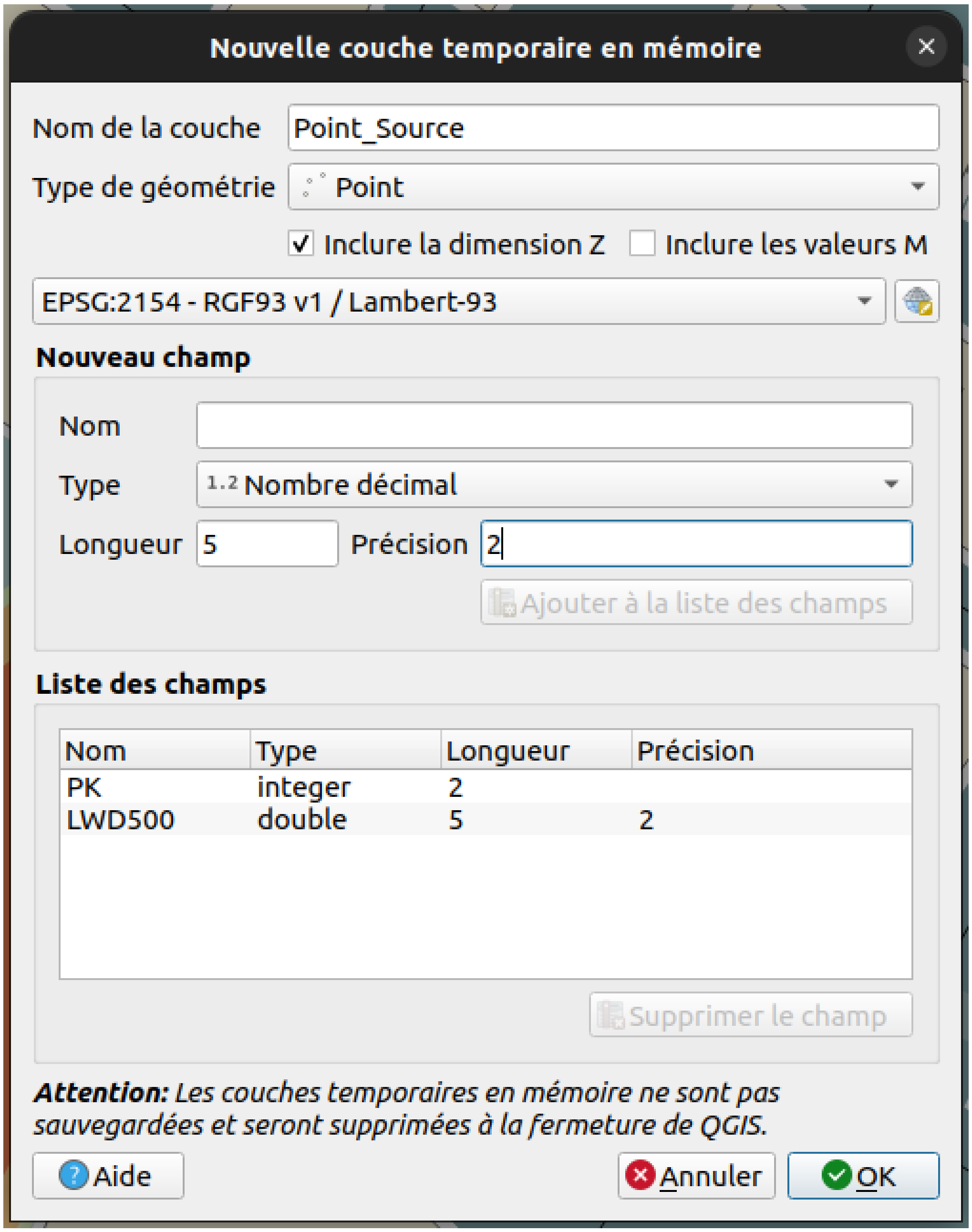
To have an interesting resulting noise map, choose to place your source point next to buildings.

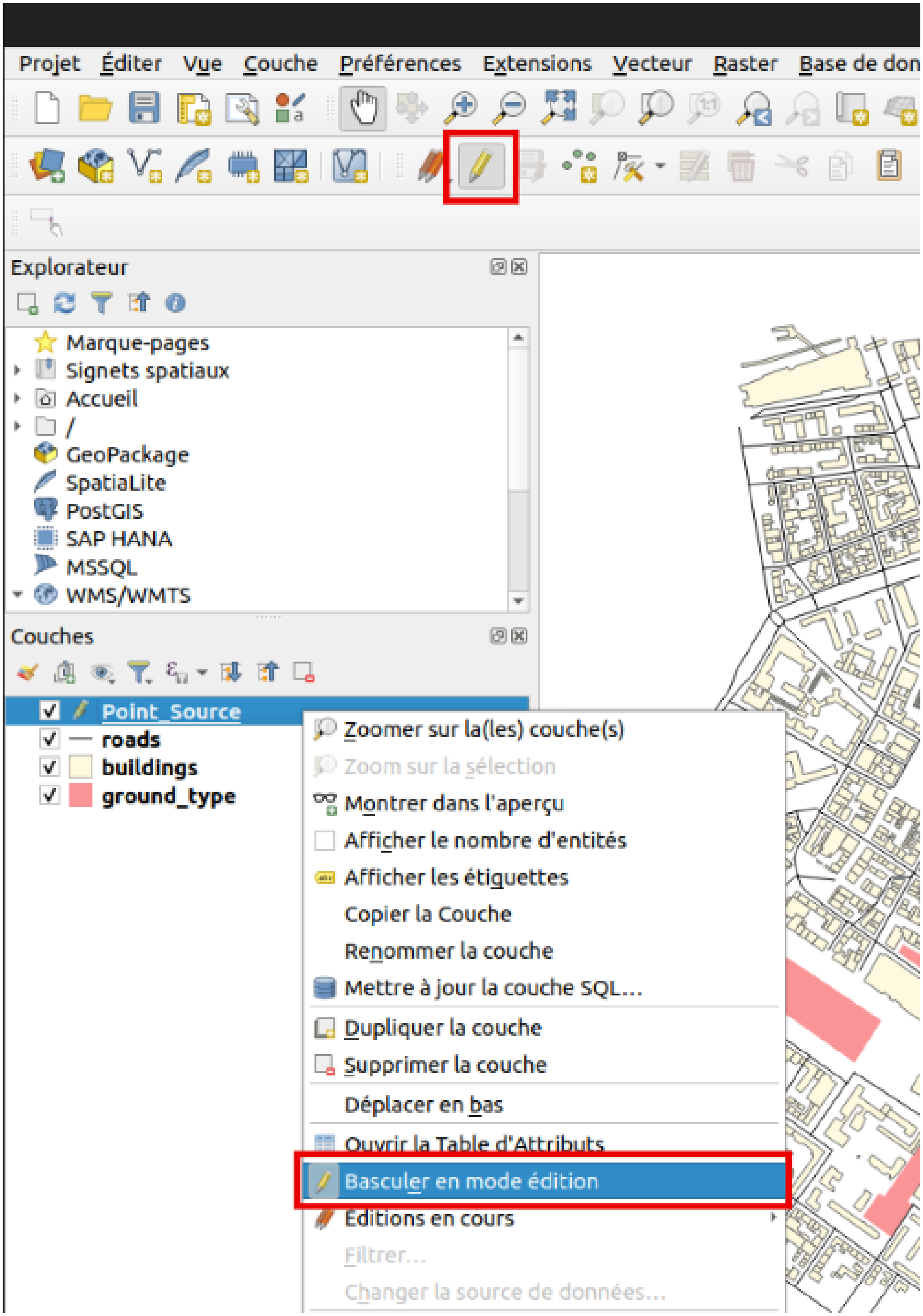
Click on the map where you want to create the source point. Once clicked, a new dialog appears and you are invited to fill the following attributes:

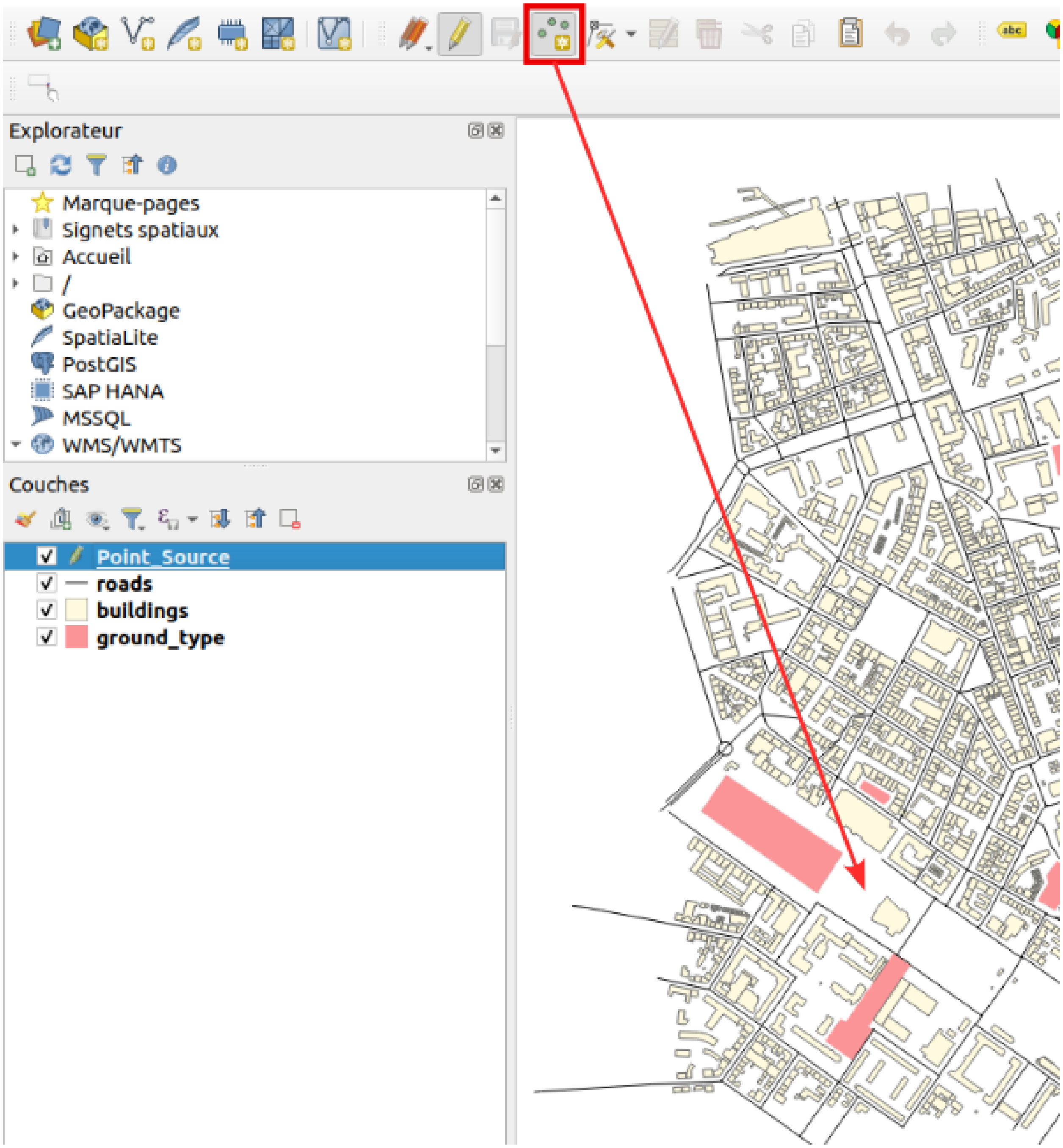
- `PK`: 1
- `HZD500` : 90

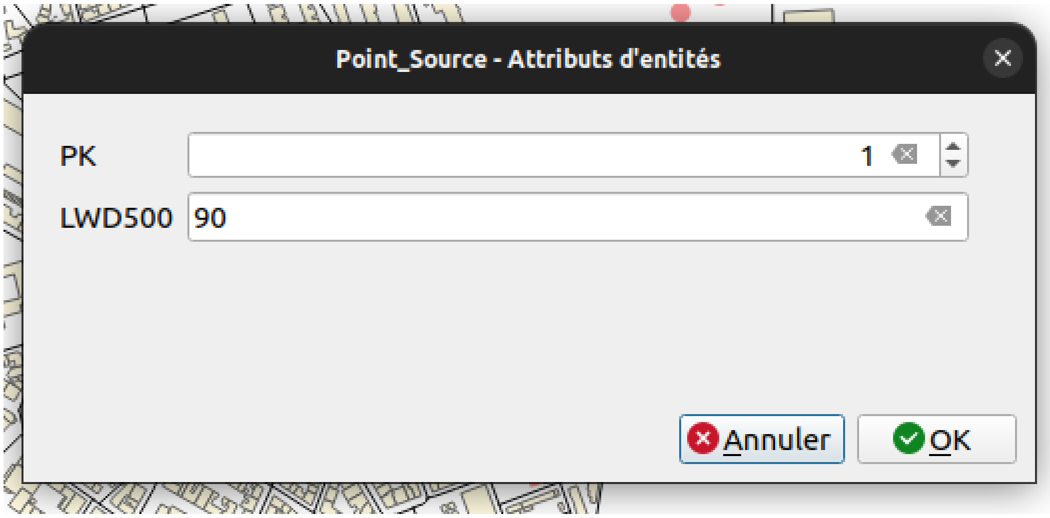
Once done, click on `OK`. The source point is now visible in the map (the blue point in the illustration below).

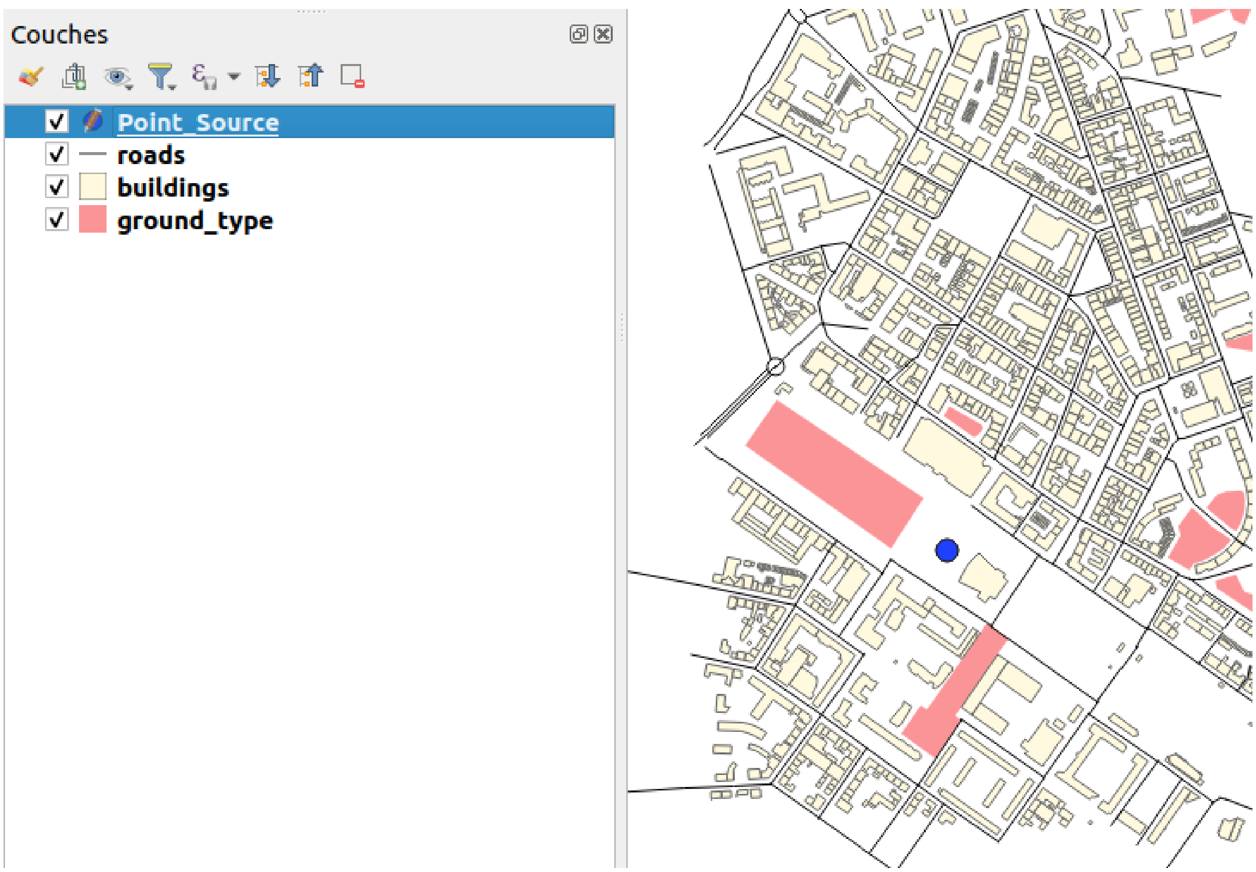
Now, we have to save this temporary layer into a flat file. To do so, just make a right-click on the layer name and choose the `Make permanent` option.

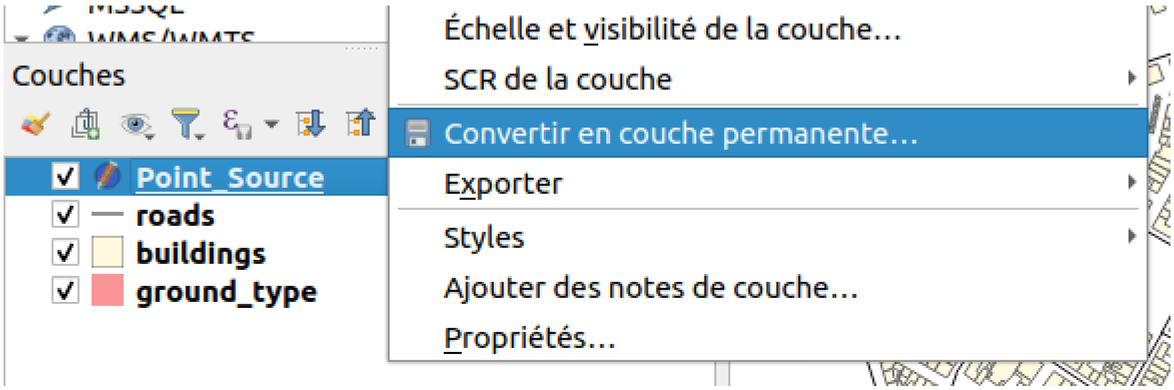

Nouvelle couche temporaire en mémoire
Nom de la couche
Point_Source
Type de géométrie
Point
Inclure la dimension Z
Inclure les valeurs M
EPSG:2154 - RGF93 v1 / Lambert-93
Nouveau champ
Nom
Type
1.2 Nombre décimal
Longueur
5
Précision
2
Ajouter à la liste des champs
Liste des champs
Nom Type Longueur Précision
PK integer 2
LWD500 double 5 2
Supprimer le champ
Attention: Les couches temporaires en mémoire ne sont pas sauvegardées et seront supprimées à la fermeture de QGIS.
Aide
Annuler
OK


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
Annuler
OK


Couches
Point_Source
roads
buildings
ground_type



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.

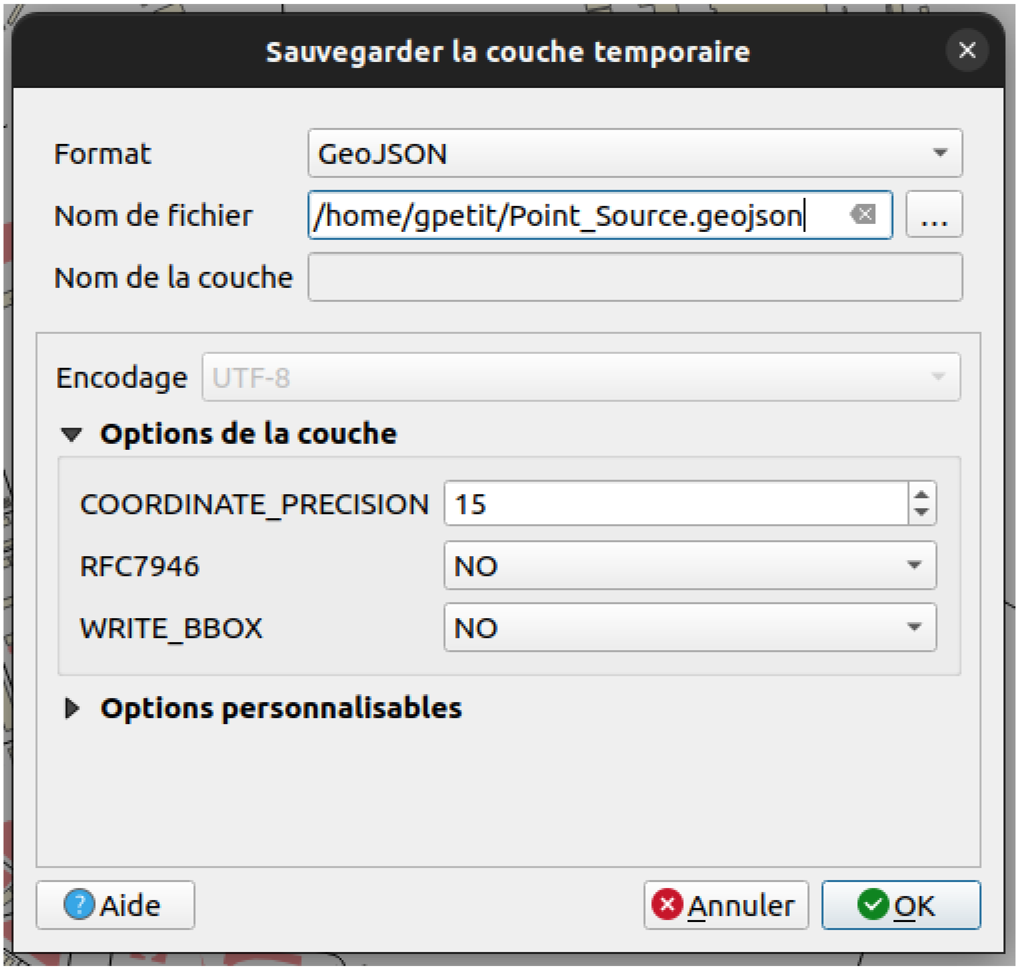
For your information, you can open `.geojson` files in most of text editor. If you do so with `Point_Source.geojson`, you will have something like this:

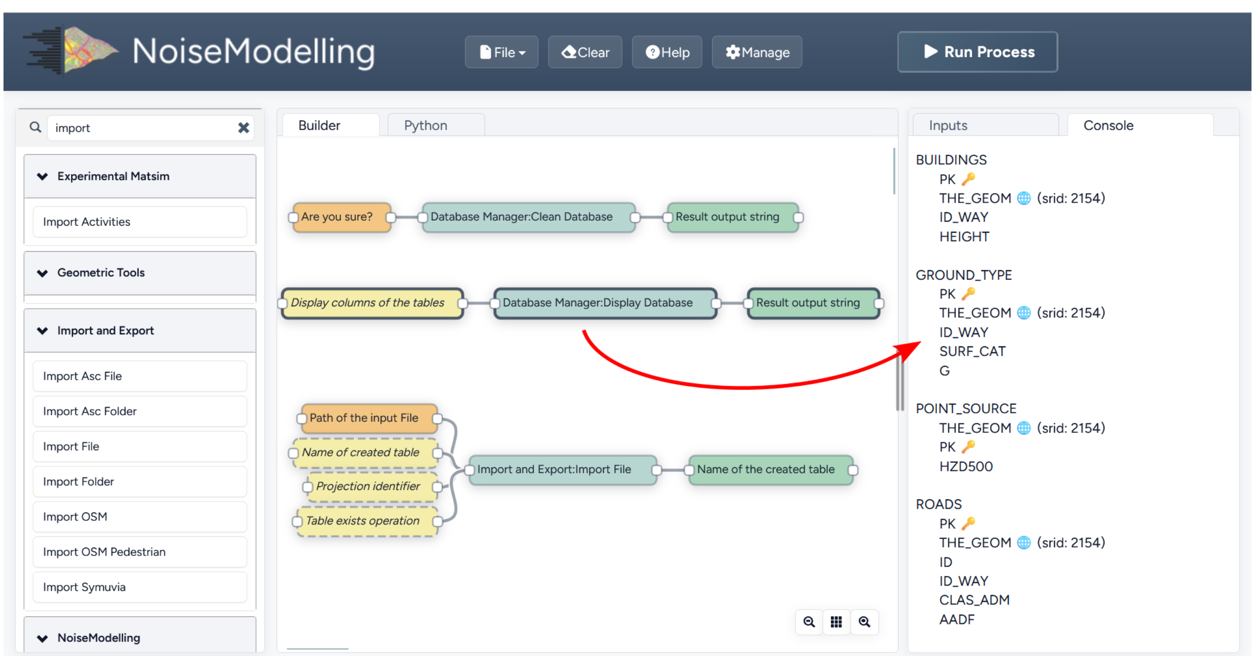
```
{
"type": "FeatureCollection",
"name": "Point_Source",
"crs": {"type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::2154" } },
"features": [{"type": "Feature", "properties": { "PK": 1, "HZD500": 90.0 },
              "geometry": {"type": "Point", "coordinates": [223771.0727, 6757583.2983, 0.
↪0]}}
             }]
}
```

5.17.2 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` script, you should see your four tables like below:


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
Aide
Annuler
OK


NoiseModelling
File
Clear
Help
Manage
Run Process
import
Experimental Matsim
Import Activities
Geometric Tools
Import and Export
Import Asc File
Import Asc Folder
Import File
Import Folder
Import OSM
Import OSM Pedestrian
Import Symuvia
NoiseModelling
Builder
Python
Are you sure?
Database Manager:Clean Database
Result output string
Display columns of the tables
Database Manager:Display Database
Result output string
Path of the input File
Name of created table
Projection identifier
Table exists operation
Import and Export:Import File
Name of the created table
Inputs
Console
BUILDINGS
PK
THE_GEOM (srid: 2154)
ID_WAY
HEIGHT
GROUND_TYPE
PK
THE_GEOM (srid: 2154)
ID_WAY
SURF_CAT
G
POINT_SOURCE
THE_GEOM (srid: 2154)
PK
HZD500
ROADS
PK
THE_GEOM (srid: 2154)
ID
ID_WAY
CLAS_ADM
AADF

5.17.3 Step 3: Generate the noise map

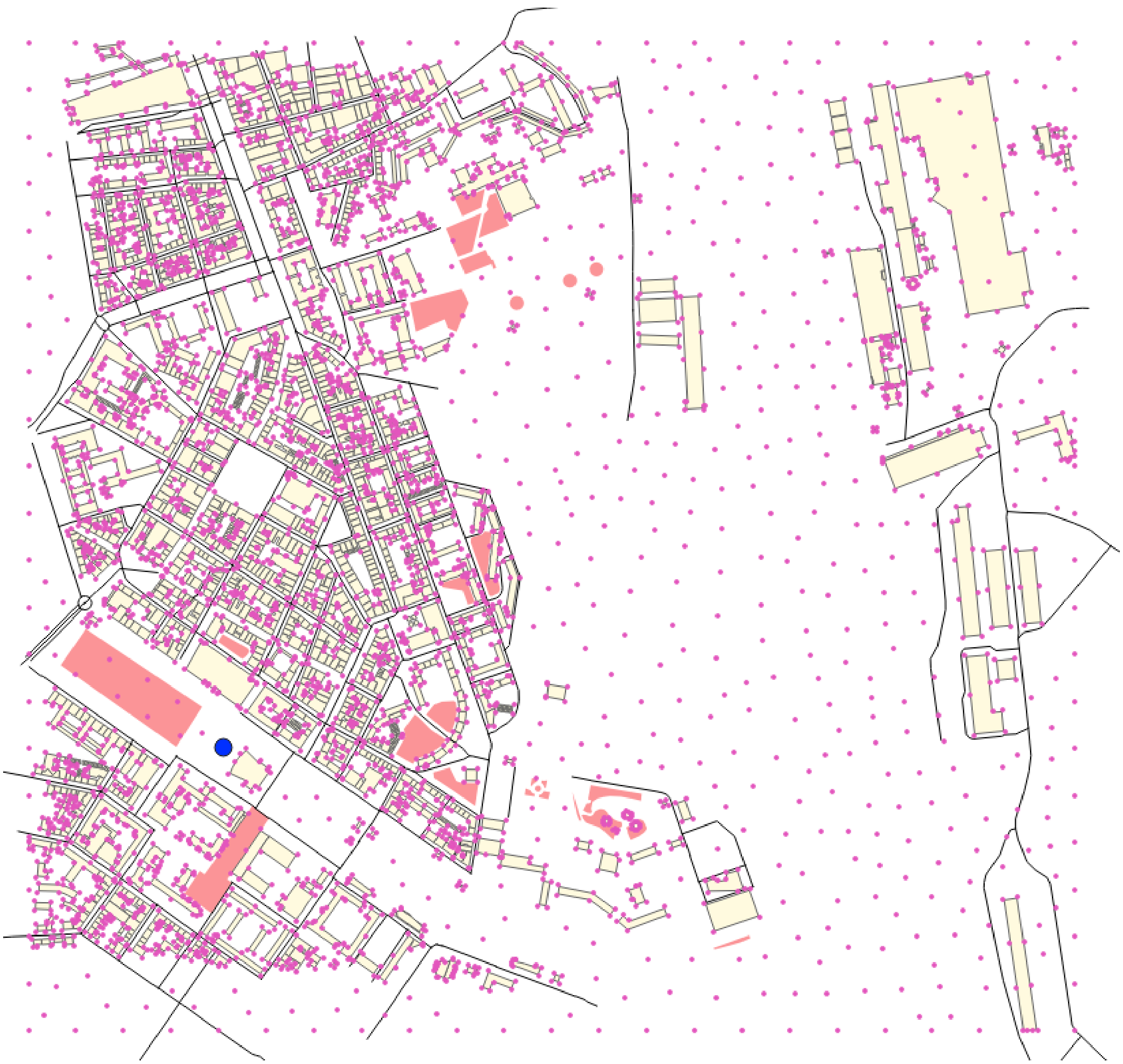
We are now ready to generate the noise map, based on a unique source point.

Create the receivers grid

Use the `Receivers:Delaunay_Grid` 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 a new table : `RECEIVERS` *(illustrated below with the purple small points)*



Calculate noise levels

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

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

Once ready, click on `Run Process` button.

A new table `RECEIVERS_LEVEL` is created.

Generate noise level isosurfaces

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

- `Sound levels table` : `RECEIVERS_LEVEL`

A new table `CONTOURING_NOISE_MAP` is created.

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

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

Filter the table according to a single period (ex. DEN):

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`.

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

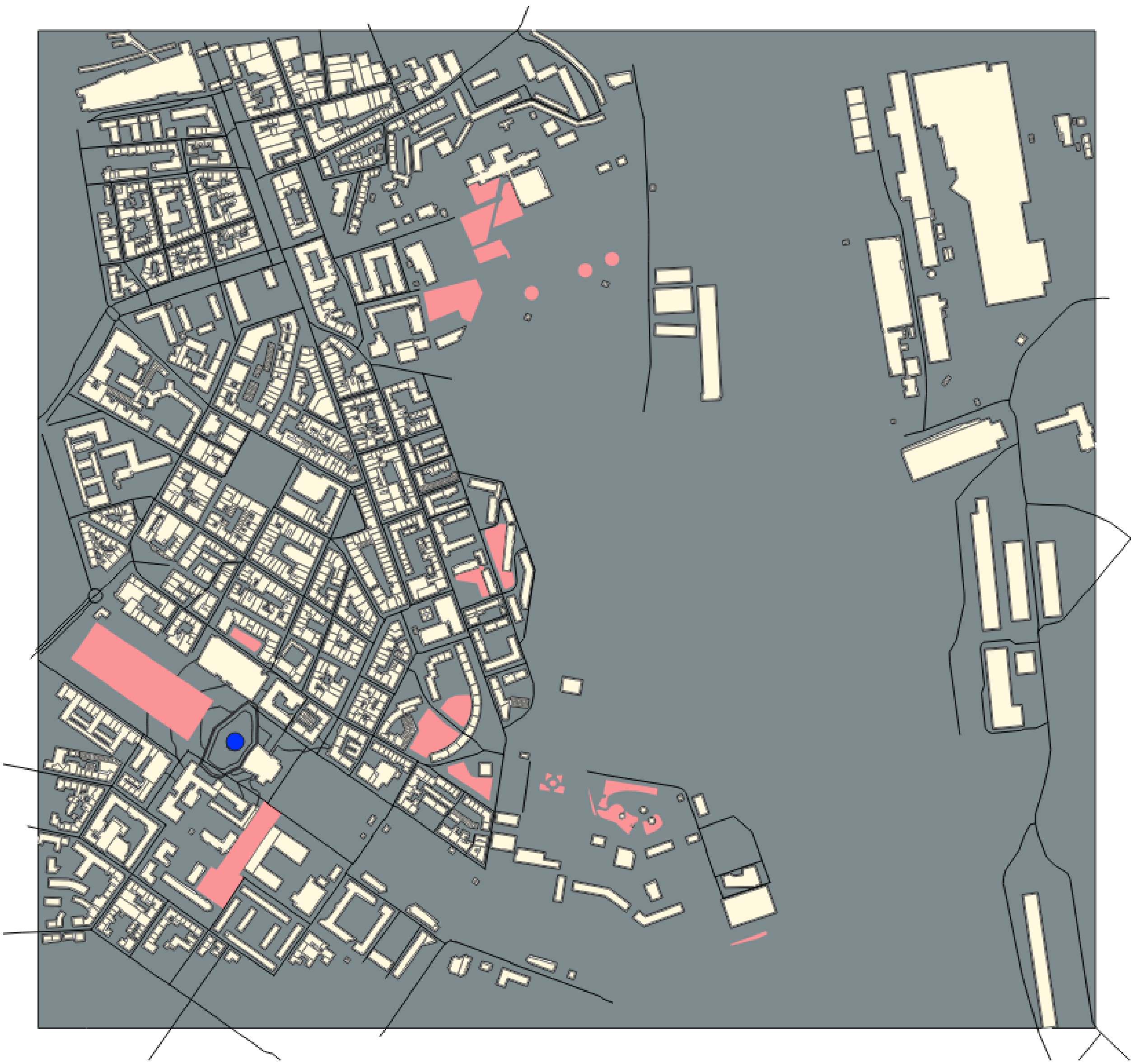
---

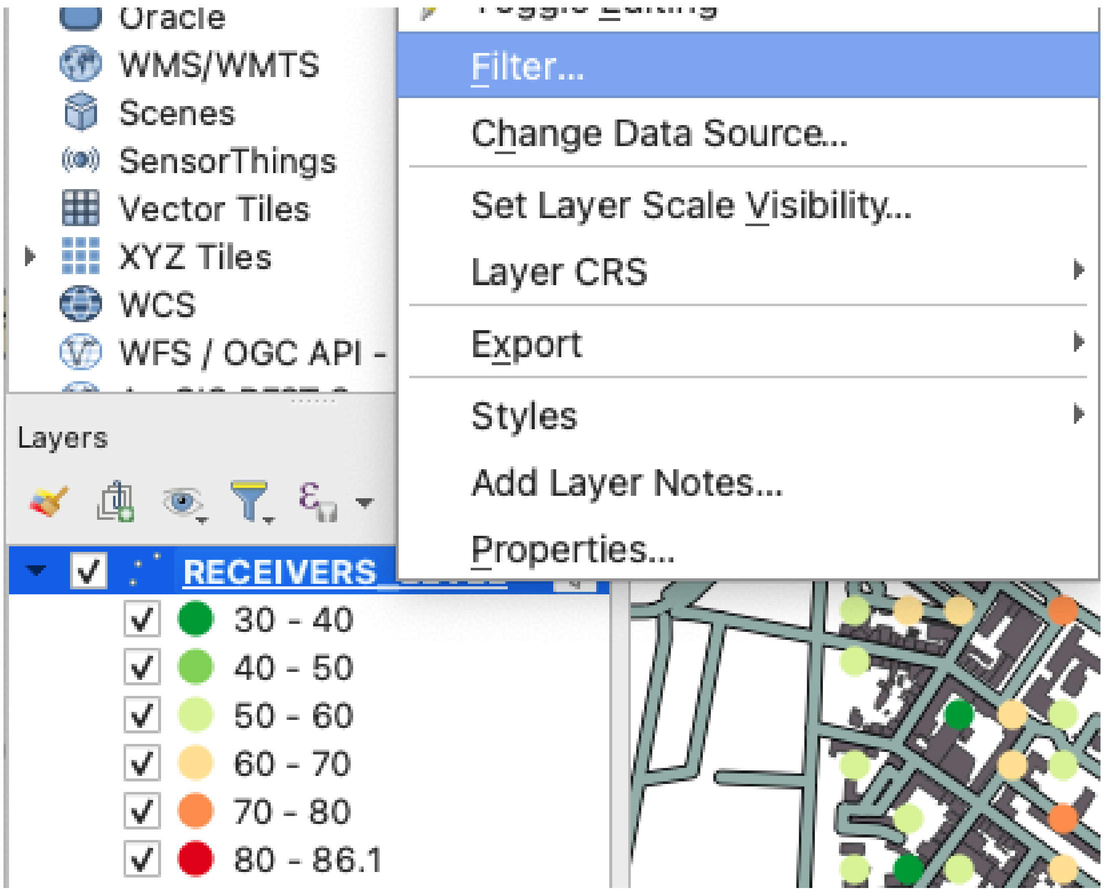
Note: If you want to know more about noise map styles, you should read the "*Noise Map Color Scheme*" page.

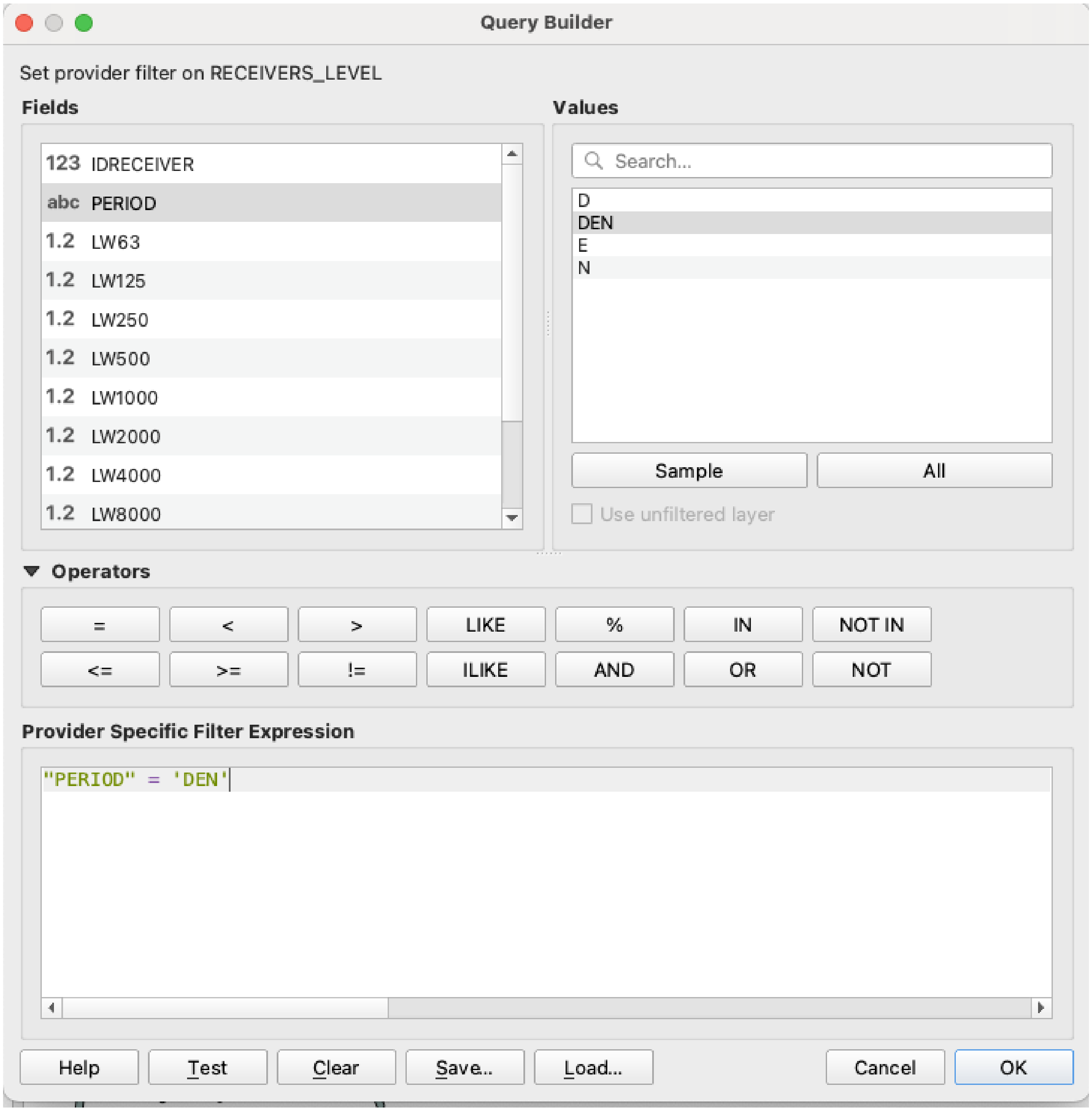
Once downloaded, make a double click on the layer `CONTOURING_NOISE_MAP`. It will opens the property panel. Here, click on the `Symbology` tab. In the `Style` menu *(at the bottom)*, choose `Load style`. Then in the opened dialog, click on the `...` icon to search the `style_beate_tomio.sld` file. Once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

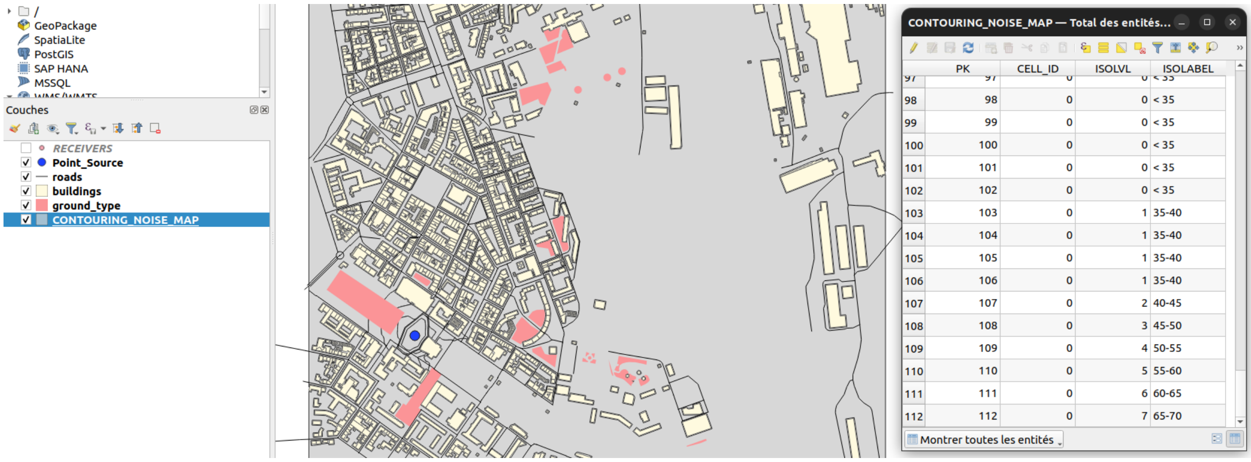
The style with its different colors is now displayed.

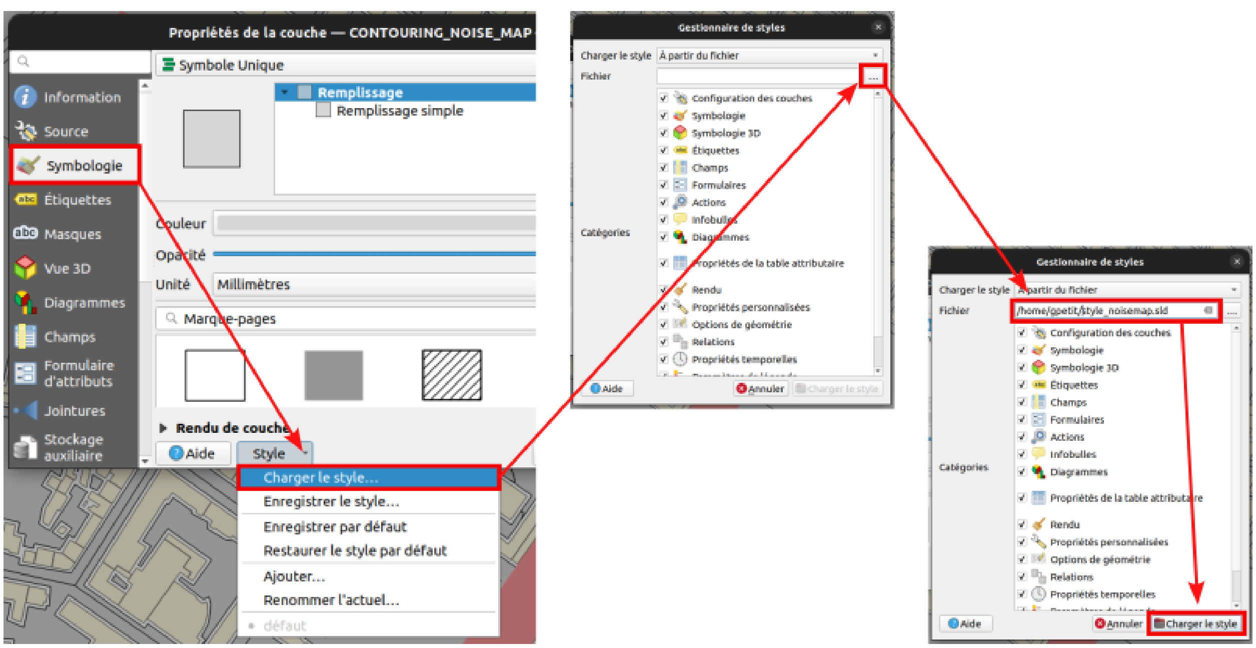
Press `OK` to apply and close the dialog. Your noise map is now well colorized and you can navigate into it to see the influence of buildings on noise levels.

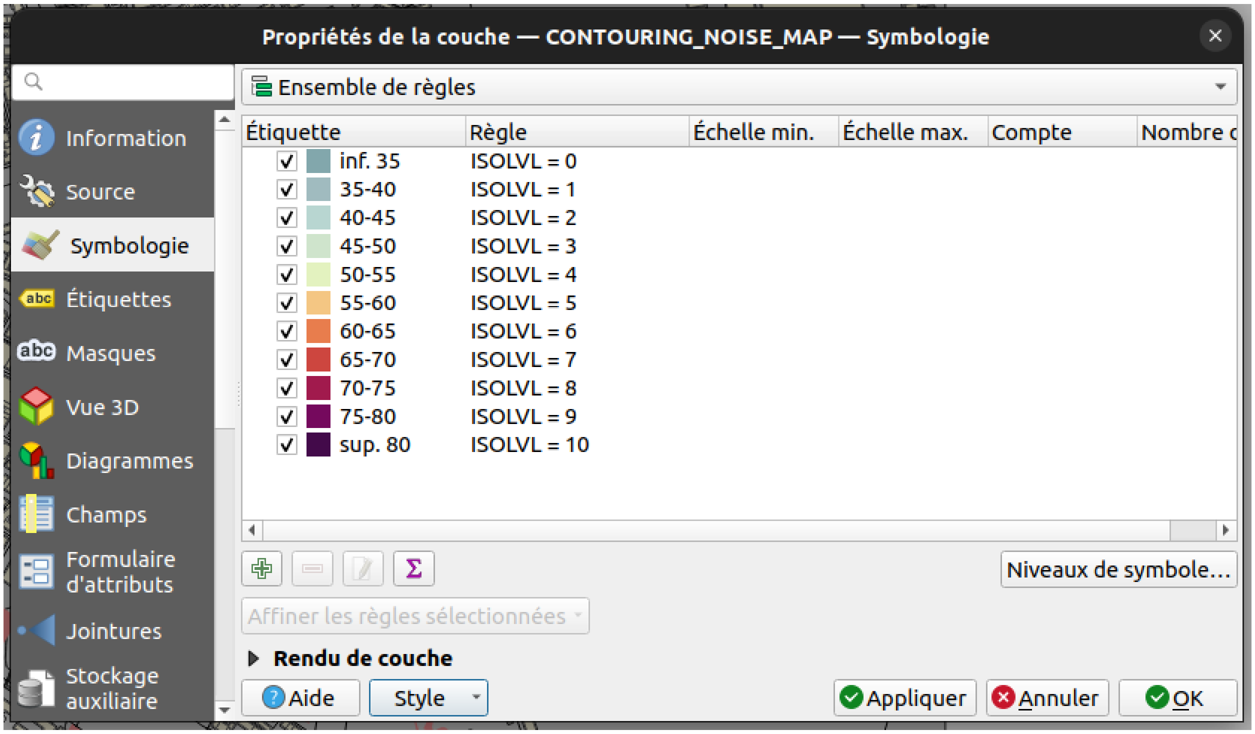


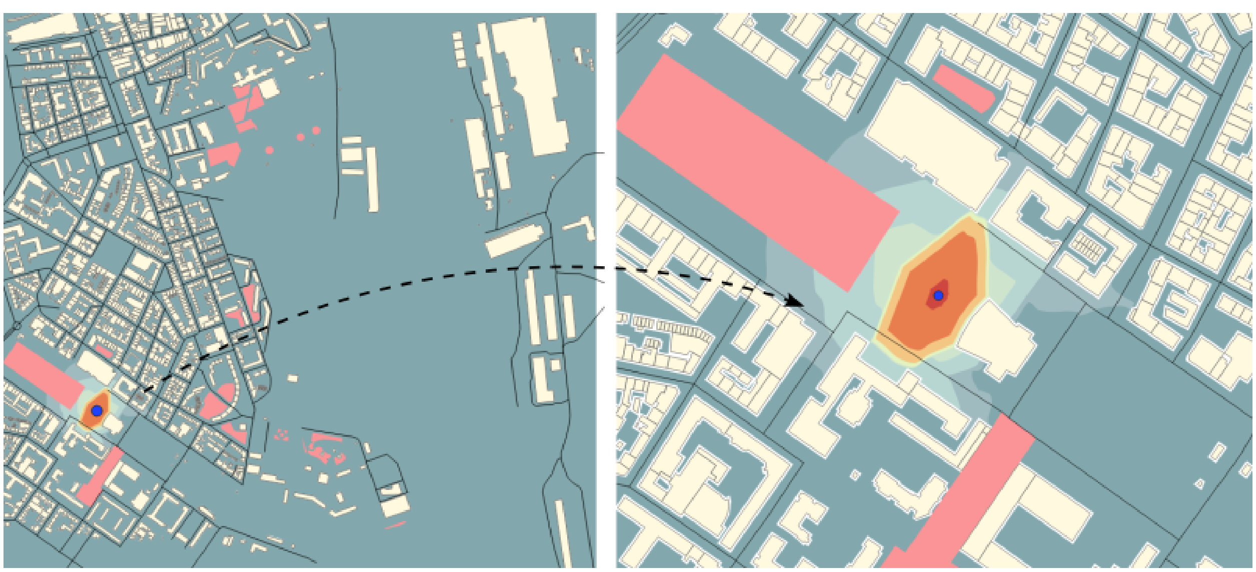

Oracle
WMS/WMTS
Scenes
SensorThings
Vector Tiles
XYZ Tiles
WCS
WFS / OGC API -
Layers
RECEIVERS
30 - 40
40 - 50
50 - 60
60 - 70
70 - 80
80 - 86.1
Filter...
Change Data Source...
Set Layer Scale Visibility...
Layer CRS
Export
Styles
Add Layer Notes...
Properties...


Query Builder
Set provider filter on RECEIVERS_LEVEL
Fields
123 IDRECEIVER
abc PERIOD
1.2 LW63
1.2 LW125
1.2 LW250
1.2 LW500
1.2 LW1000
1.2 LW2000
1.2 LW4000
1.2 LW8000
Values
Search...
D
DEN
E
N
Sample
All
Use unfiltered layer
Operators
=
<
>
LIKE
%
IN
NOT IN
<=
>=
!=
ILIKE
AND
OR
NOT
Provider Specific Filter Expression
"PERIOD" = 'DEN'
Help
Test
Clear
Save...
Load...
Cancel
OK


GeoPackage
SpatiaLite
PostGIS
SAP HANA
MSSQL
Couches
RECEIVERS
Point_Source
roads
buildings
ground_type
CONTOURING_NOISE_MAP
CONTOURING_NOISE_MAP — Total des entités...
PK
CELL_ID
ISOLVL
ISOLABEL
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
Montrer toutes les entités


Propriétés de la couche — CONTOURING_NOISE_MAP
Symbole Unique
Information
Source
Symbologie
Étiquettes
Masques
Vue 3D
Diagrammes
Champs
Formulaire d'attributs
Jointures
Stockage auxiliaire
Remplissage
Remplissage simple
Couleur
Opacité
Unité
Millimètres
Marque-pages
Rendu de couche
Aide
Style
Charger le style…
Enregistrer le style…
Enregistrer par défaut
Restaurer le style par défaut
Ajouter…
Renommer l'actuel…
défaut
Gestionnaire de styles
Charger le style
À partir du fichier
Fichier
Catégories
Configuration des couches
Symbologie
Symbologie 3D
Étiquettes
Champs
Formulaires
Actions
Infobulles
Diagrammes
Propriétés de la table attributaire
Rendu
Propriétés personnalisées
Options de géométrie
Relations
Propriétés temporelles
Aide
Annuler
Charger le style
Gestionnaire de styles
Charger le style
À partir du fichier
Fichier
/home/gpetit/style_noisemap.sld
Catégories
Configuration des couches
Symbologie
Symbologie 3D
Étiquettes
Champs
Formulaires
Actions
Infobulles
Diagrammes
Propriétés de la table attributaire
Rendu
Propriétés personnalisées
Options de géométrie
Relations
Propriétés temporelles
Aide
Annuler
Charger le style

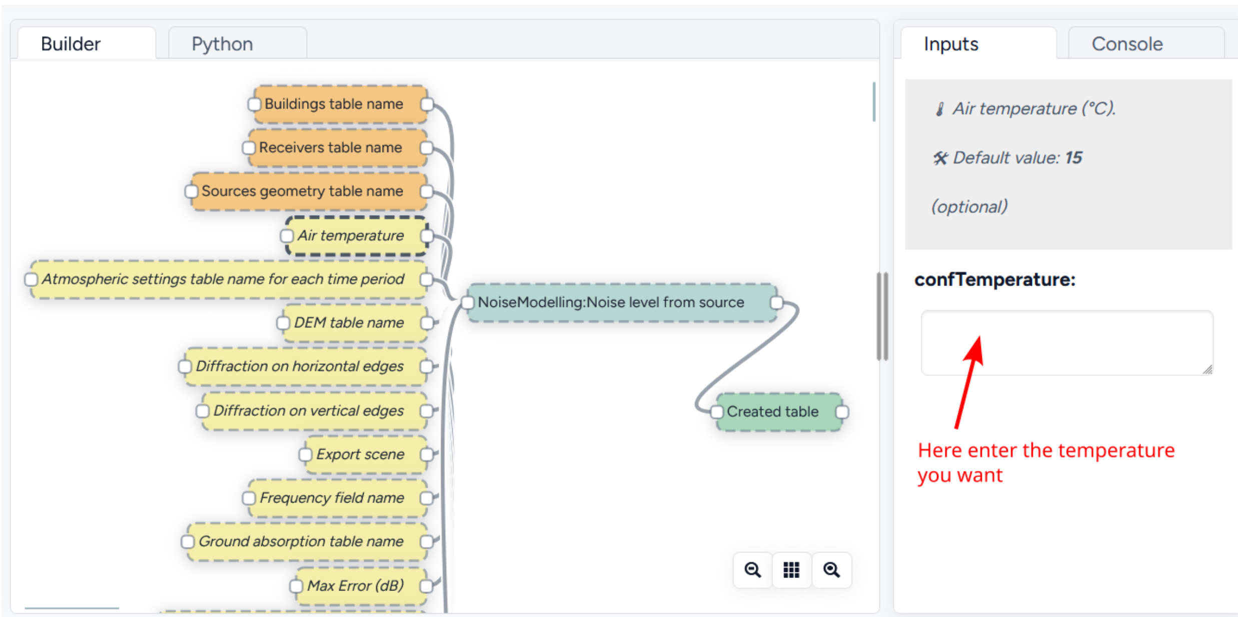

Propriétés de la couche — CONTOURING_NOISE_MAP — Symbologie
Ensemble de règles
Information
Source
Symbologie
Étiquettes
Masques
Vue 3D
Diagrammes
Champs
Formulaire d'attributs
Jointures
Stockage auxiliaire
Étiquette
Règle
Échelle min.
Échelle max.
Compte
inf. 35 ISOLVL = 0
35-40 ISOLVL = 1
40-45 ISOLVL = 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
Niveaux de symbole…
Affiner les règles sélectionnées
Rendu de couche
Aide
Style
Appliquer
Annuler
OK



5.17.4 Step 4: Change the default parameters

To produce this noise map, we used, in most Blocks, 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 optional parameters *(the yellow boxes)* just select them and fill the needed information in the right-side menu.



5.17.5 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.

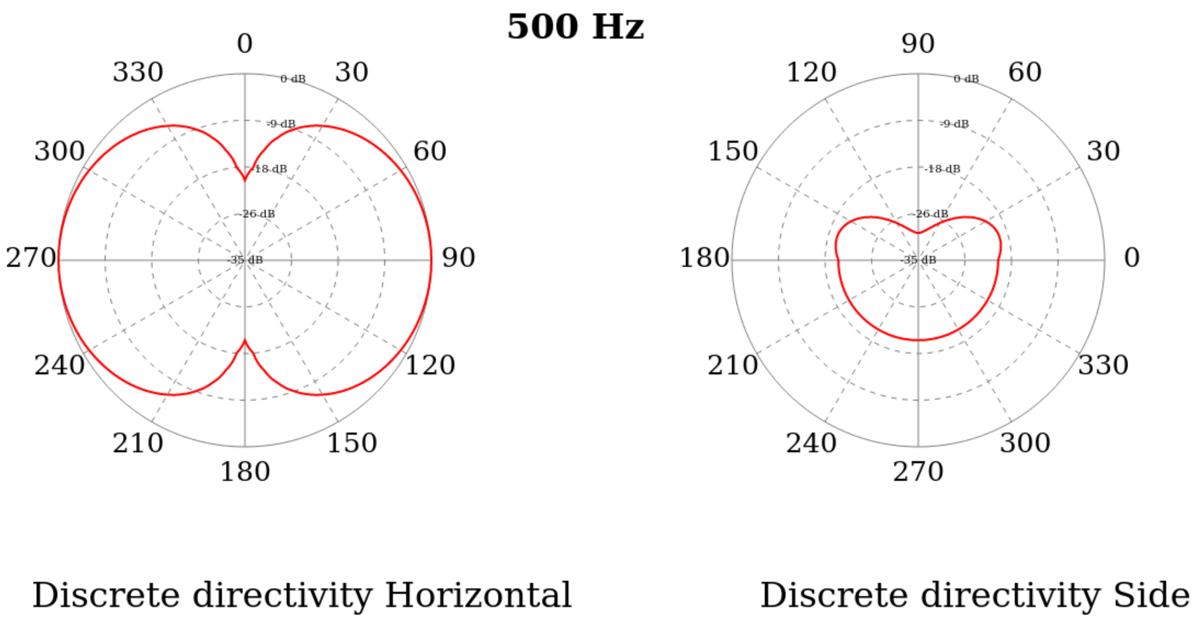
- `DIR_ID` : identifier of the directivity sphere
- `THETA` : vertical angle in degrees, 0 (front), -90 (bottom), 90 (top), from -90 to 90
- `PHI`: horizontal angle in degrees, 0 (front) / 90 (right), from 0 to 360
- `HZ500` : attenuation levels in dB for 500 Hz

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	HZ500
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
  - `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.

```
{ "PK": 1, "HZD500": 100.0}
```

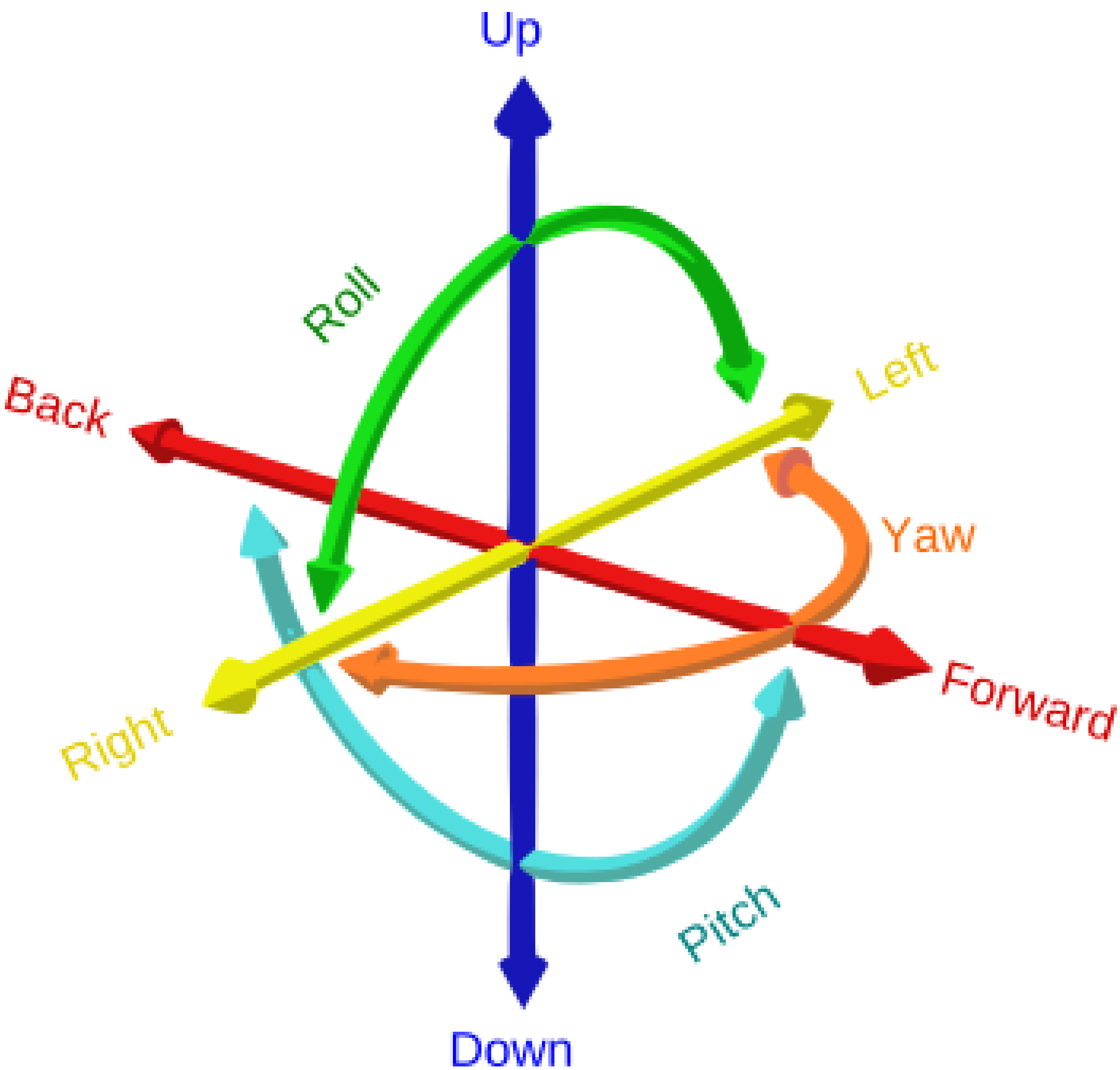
by

```
{ "PK": 1, "HZD500": 100.0, "YAW": 45, "PITCH": 0, "ROLL": 0, "DIR_ID" : 1 }
```

Here we can see that the Yaw is set to 45°. Pitch and Roll are equal to 0, and the directivity is defined as 1 and will refer to the directivity 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, "HZD500": 100.0, "YAW": 45,
↪"PITCH": 0, "ROLL": 0, "DIR_ID" : 1 },
              "geometry": {"type": "Point", "coordinates": [223771.0727, 6757583.2983, 0.
↪0]}
             }]
}
```


Up
Roll
Back
Left
Yaw
Forward
Right
Pitch
Down

Import data

Now, in NoiseModelling we have to:

- Import the `Directivity.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 `resources/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 file` script.

Generate the Delaunay triangulation

Use the `Receivers:Delaunay_Grid` 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` 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`
- `DEM table name` : `DEM`

Create isosurface

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

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

Export and visualize resulting tables

Use the `Import_and_Export:Export_Table` 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 and filter the period to `DEN`. Apply the `noisemap_style.sld` style, and compare with `CONTOURING_NOISE_MAP.shp` produced in Step 3.

5.18 Noise Map from OSM - GUI

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

5.18.1 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 block `Clean_Database`. Don't forget to check the `Are you sure` check box before running the process.

5.18.2 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: Developed 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` 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 address of your `my_area.osm.pbf` file (*e.g.* `/home/noisemodelling/my_area.osm.pbf`)
3. If needed, check the 4 other optional 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".

5.18.3 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` script.
- A spatial layer can be visualized using `Table_Visualization_Map` script.
- A data table can be visualized using `Table_Visualization_Data` 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` script, in Shapefile, CSV or GeoJSON format.
- View the files: Then open these files into your preferred Geographic Information System (*e.g* QGIS, OpenJUMP, . . .). 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).

5.18.4 Step 3: Generate a Receiver table

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

Use functions/Receivers/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 Builder or in your GIS to check that it meets your expectations.

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

5.18.5 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 Block.

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

5.18.6 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 (use the created table LW_ROADS as *Source geometry table name*) propagated to the receivers according to the CNOSSOS-EU. propagation laws.

5.18.7 Step 6: Create Isosurfaces map

Create an interpolation of levels between receivers points using the block *Create Isosurface*.

Set RECEIVERS_LEVEL as `Name of the noise table`.

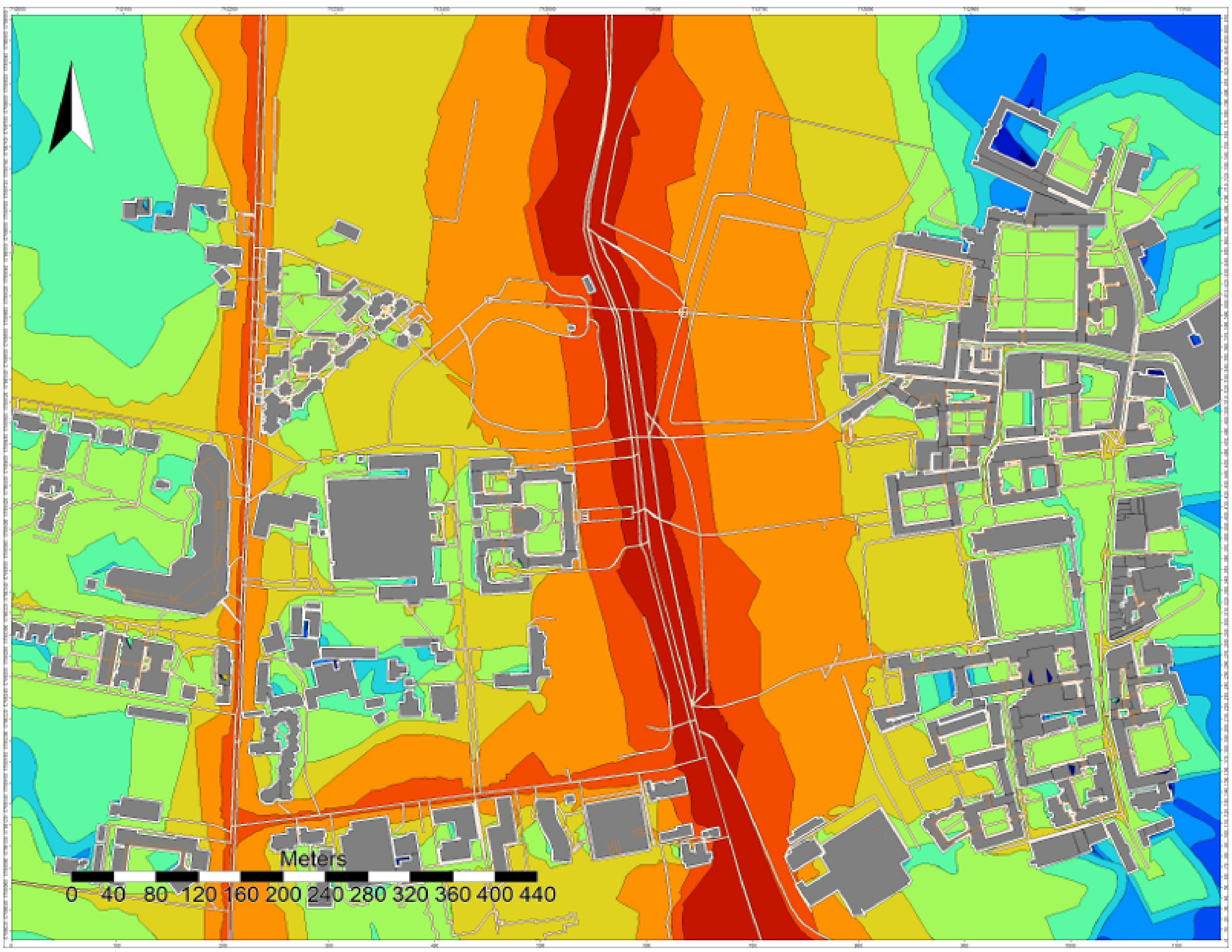
5.18.8 Step 7: View the result

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. Drop the file in QGIS then filter by the *period* column (ex. DEN).
You can then apply a (revert) color gradient on ISOLABEL field.


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

5.19 MATSim - GUI

5.19.1 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 a small part of the center of Nantes, the 6th most populated city in France.

5.19.2 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

5.19.3 The data

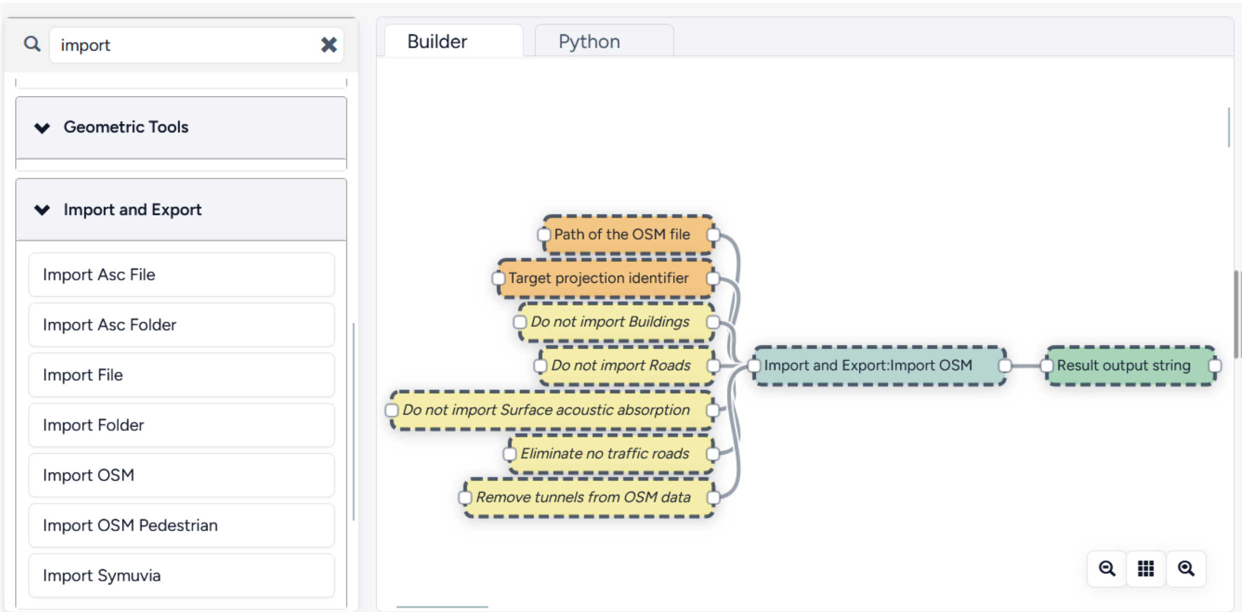
You can download and unzip the data in any folder from here: https://github.com/Universite-Gustave-Eiffel/NoiseModelling/releases/download/v5.X-Matsim-Test-Scenario/scenario_matsim.zip

The data folder should contain the following files:

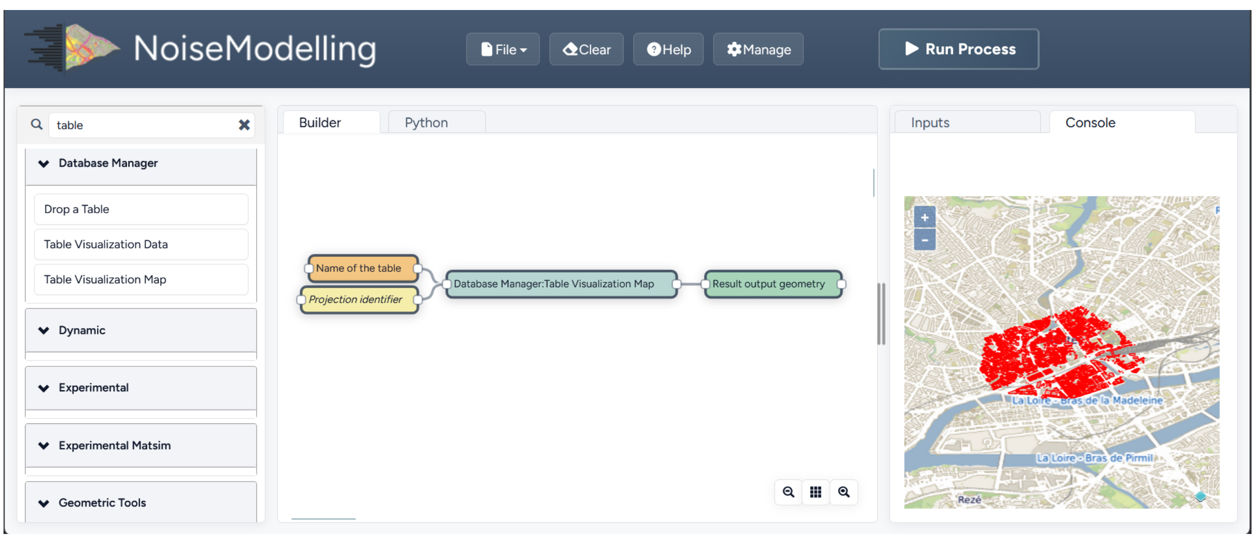
- `nantes_mini.osm.pbf`: the Openstreetmap data of the area. We'll use it to import buildings into NoiseModelling.
- `detailed_network.csv`: A file containing the 'true' geometries of the road segments (called "links" in MATSim)
- `output_allVehicles.xml.gz`: A file containing the various vehicles used by the agents in the simulation.
- `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 planned schedule.
- `output_experienced_plans.xml.gz`: A file containing the list of agents and their final experienced schedule which may differ from the initial planned one.
- `output_persons.csv.gz`: A file containing the simple list of agents and some of there socioeconomic characteristics

5.19.4 Step 1: Import Buildings

The first thing we're going to do is to import buildings. We use the `Import_OSM` block to do that. There are several options. Put the `nantes_mini.osm.pbf` path in the 'pathFile' input and set the 'SRID' input to 2154 *(which is the EPSG code for the french regulatory system)*. Since we're only interested in buildings, you can also check the `Do not import roads` option as well as the `` `Do not import Surface acoustic absorption`` option.



You should end up with a `BUILDINGS` table containing the city center buildings. If you want to visualize the buildings in NoiseModelling, you can use the `Table visualization Map` block (in the *Database Manager* section) with `Name of the table` set to *BUILDINGS* and `Projection identifier` set to *2154*.



5.19.5 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` 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

An important option is `The size of time bins in seconds` which represents the time period you want to aggregate the traffic data over. Here let's use 900 to get data every 15 minutes.

One optional, but very important input, is the `Network CSV` file path option. 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 buildings into account (simplified links can pass through buildings) and since source-receiver distance has a big impact on noise levels. That's why the `detailed_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 0,1% of the area total population (that is a population factor of 0.001).

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/detailed_network.csv`
- Export additional traffic data ?: `true`
- Path of the Matsim output folder: `/path/to/your/scenario_matsim`
- populationFactor: `0.001`
- The size of time bins in seconds: `900`
- outTableName: "" (not set, use default)
- Skip unused links ?: `true`
- Projection identifier: `2154`

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

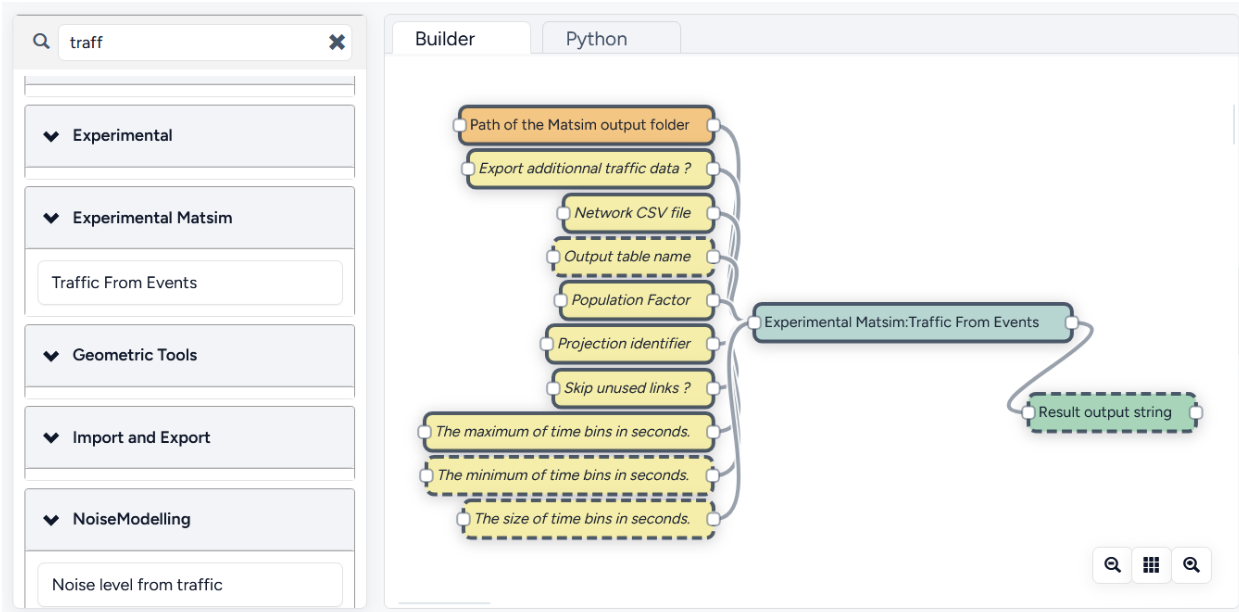
5.19.6 Step 3: Import MATSim Activities

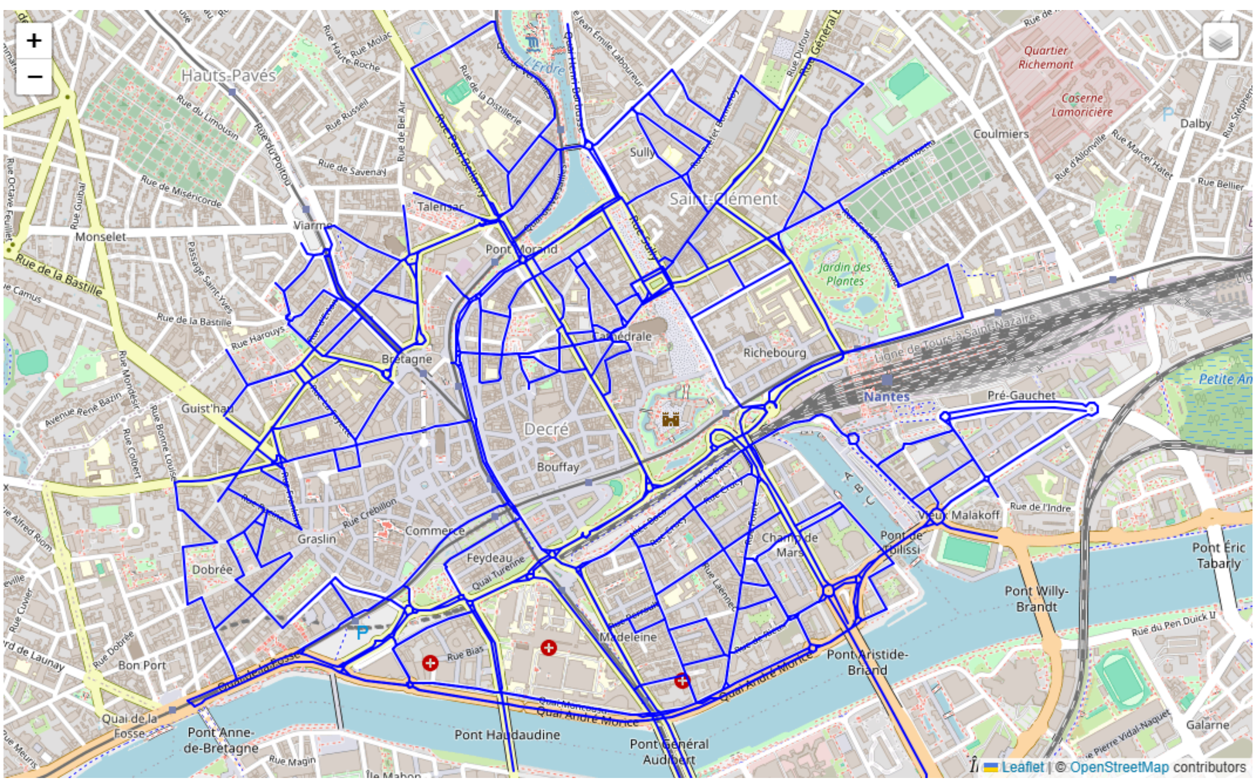
The next step consists in importing the activities locations from the MATSim simulation.In MATSim, activities are also called facilities.

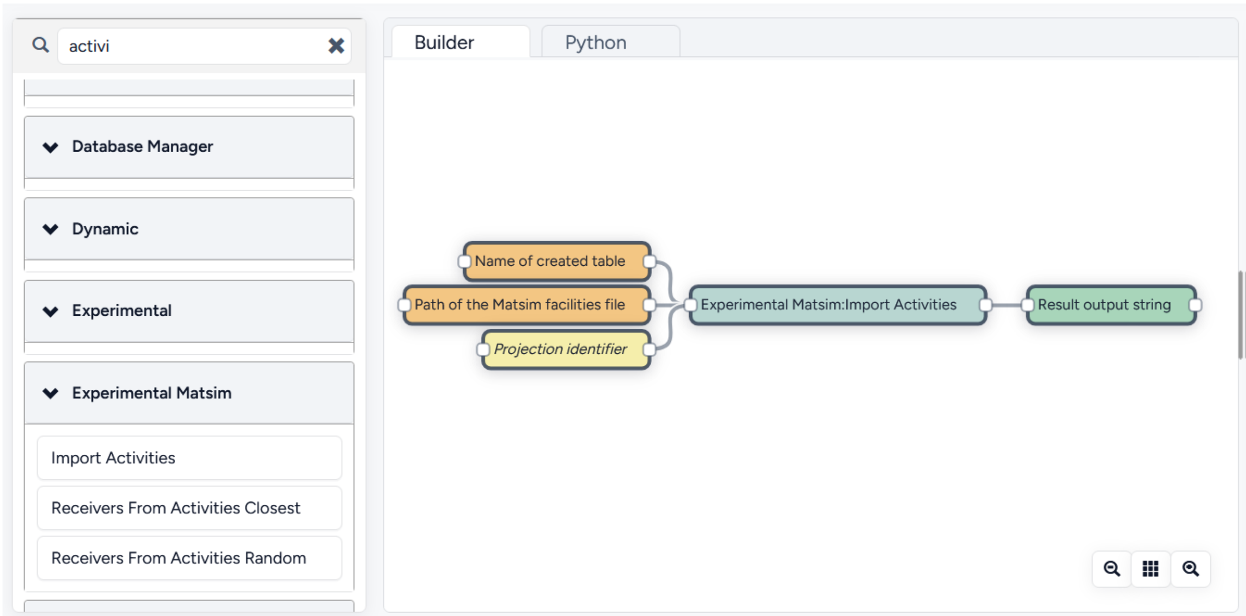
Let's use the `Import_Activities` bloc. The inputs descriptions are quite straightforward:

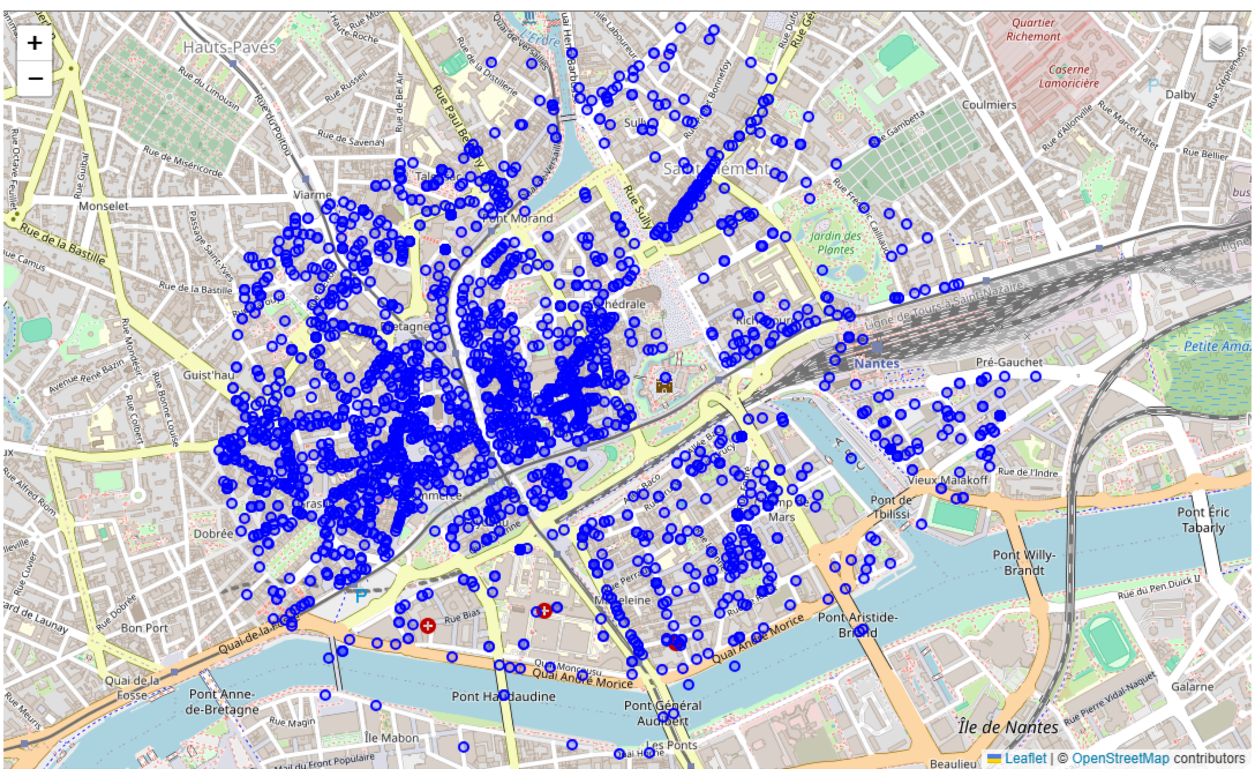
- Name of created table: `ACTIVITIES`
- Projection identifier: `2154`
- Path of MatSim facilities file: `/path/to/your/scenario_mastim/output_facilities.xml.gz`

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


traff
Experimental
Experimental Matsim
Traffic From Events
Geometric Tools
Import and Export
NoiseModelling
Noise level from traffic
Builder
Python
Path of the Matsim output folder
Export additionnal traffic data ?
Network CSV file
Output table name
Population Factor
Projection identifier
Skip unused links ?
The maximum of time bins in seconds.
The minimum of time bins in seconds.
The size of time bins in seconds.
Experimental Matsim:Traffic From Events
Result output string


Hauts-Pavés
Saint-Clément
Monselet
Rue de la Bastille
Viarme
Talensac
Pont Morand
Cathédrale
Richebourg
Bretagne
Decré
Bouffay
Graslin
Commerce
Feydeau
Dobrée
Bon Port
Quai de la Fosse
Pont Anne-de-Bretagne
Pont Haudaudine
Pont Général Audibert
Madeleine
Champ de Mars
Pont Aristide-Briand
Pont Willy-Brandt
Pont Éric Tabarly
Vieux Malakoff
Nantes
Pré-Gauchet
Coulmiers
Dalby
Quartier Richemont
Caserne Lamoricière
Jardin des Plantes
Galarne
Guist'hau
Leaflet | © OpenStreetMap contributors


activi
Database Manager
Dynamic
Experimental
Experimental Matsim
Import Activities
Receivers From Activities Closest
Receivers From Activities Random
Builder
Python
Name of created table
Path of the Matsim facilities file
Projection identifier
Experimental Matsim:Import Activities
Result output string


Hauts-Pavés
Monselet
Rue de la Bastille
Saint-Clément
Jardin des Plantes
Nantes
Pré-Gauchet
Quartier Richemont
Caserne Lamoricière
Coulmiers
Dalby
Petite Amazonie
Vieux Malakoff
Pont de Tbilissi
Pont Éric Tabarly
Pont Willy-Brandt
Pont Aristide-Briand
Pont Haudaudine
Pont Général Audibert
Pont Anne-de-Bretagne
Quai de la Fosse
Quai André Morice
Île de Nantes
Île Mabon
Galarne
Beaulieu
Bon Port
Dobrée
Guist'hau
Leaflet | © OpenStreetMap contributors

5.19.7 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` 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` bloc. Or we can randomly choose a receiver on the closest building of each activity using the `Receivers_From_Activity_Random` bloc.

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

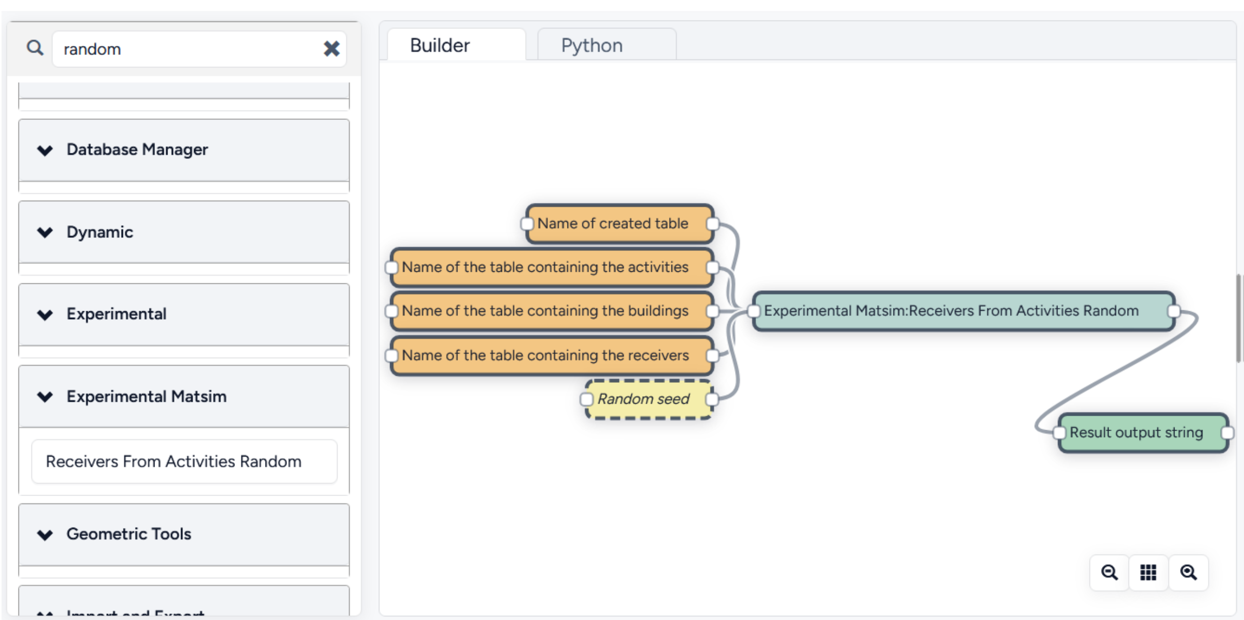
Let's calculate all the receivers around our buildings using the `Building_Grid` bloc with the following inputs:

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

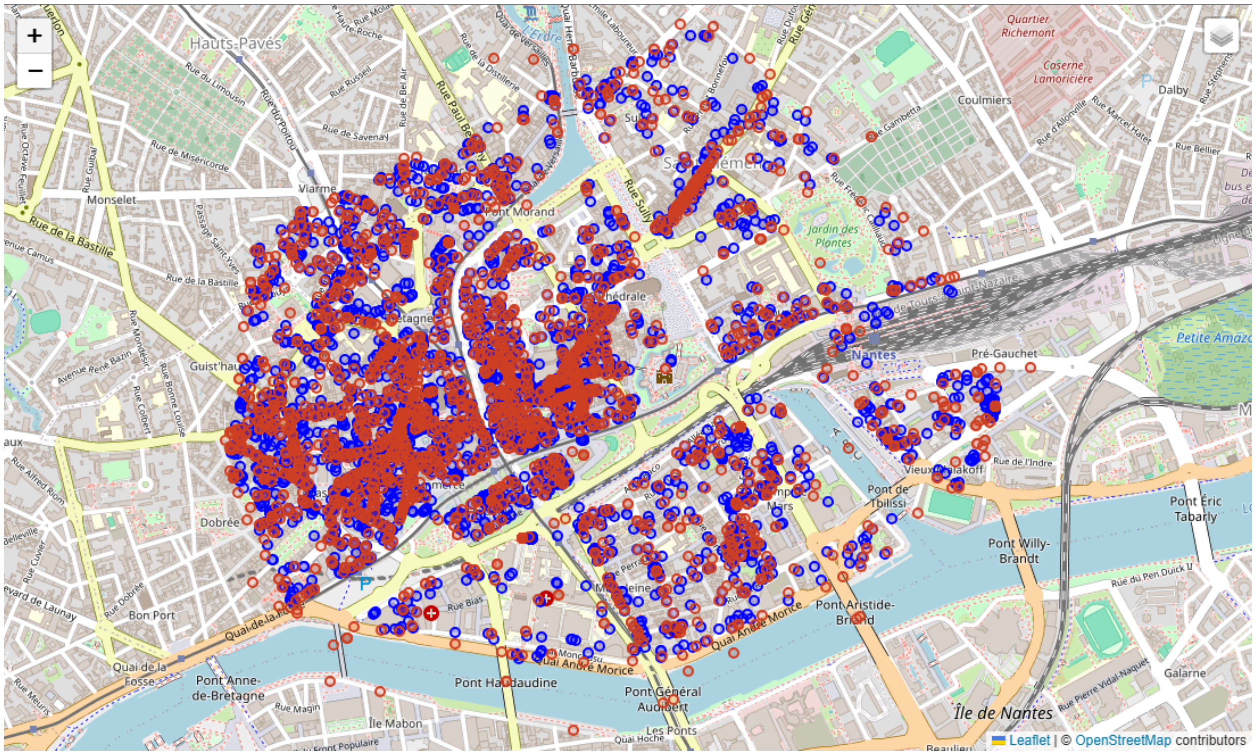
That will place receivers around all the buildings, at 4 meter high and 5 meters apart.

Now, we must use the `Receivers_From_Activity_Random` 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 original matsim position (`ORIGIN_GEOM`, in red below). You can inspect the results to see where each activity is placed now.



5.19.8 Step 5: Calculate Noise Receiver Levels

In this step, we want to calculate a noise level for every receiver, every 15 minutes (96 maps in total).

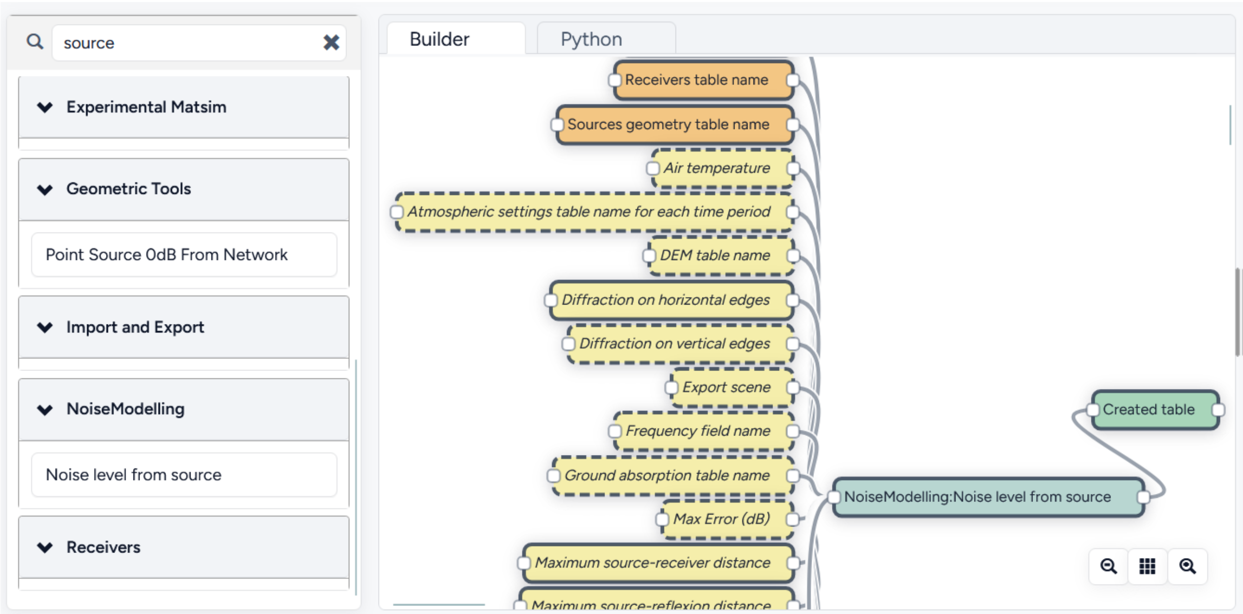
We'll use the `Noise_level_from_source` bloc. When the using the `Sources emission table name`, the bloc will automatically take into account the time bins defined in the emission table. For more details about the different parameters, browse the NoiseModelling general documentation.

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

The parameters we will use are the following:

- Buildings table name: `BUILDINGS`
- Receivers table name: `ACTIVITY_RECEIVERS`
- Sources geometry table name: `MATSIM_ROADS`
- Sources emission table name: `MATSIM_ROADS_LW`
- Maximum source-receiver distance: `250`
- Maximum source reflexion distance: `50`
- Order of reflexion: `1`
- Diffraction on vertical edges: `false`
- Diffraction on horizontal edges: `true`

We should end up with a table called `RECEIVERS_LEVEL` that contains a list of noise levels for every receiver, at every 15-minute interval.



We have our noise maps !

5.19.9 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 `RECEIVERS_LEVEL` table data into a Shapefile. We'll simply use the `Export_Table WPS` bloc with the following parameters:

- Name of the table: `RECEIVERS_LEVEL`
- 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.

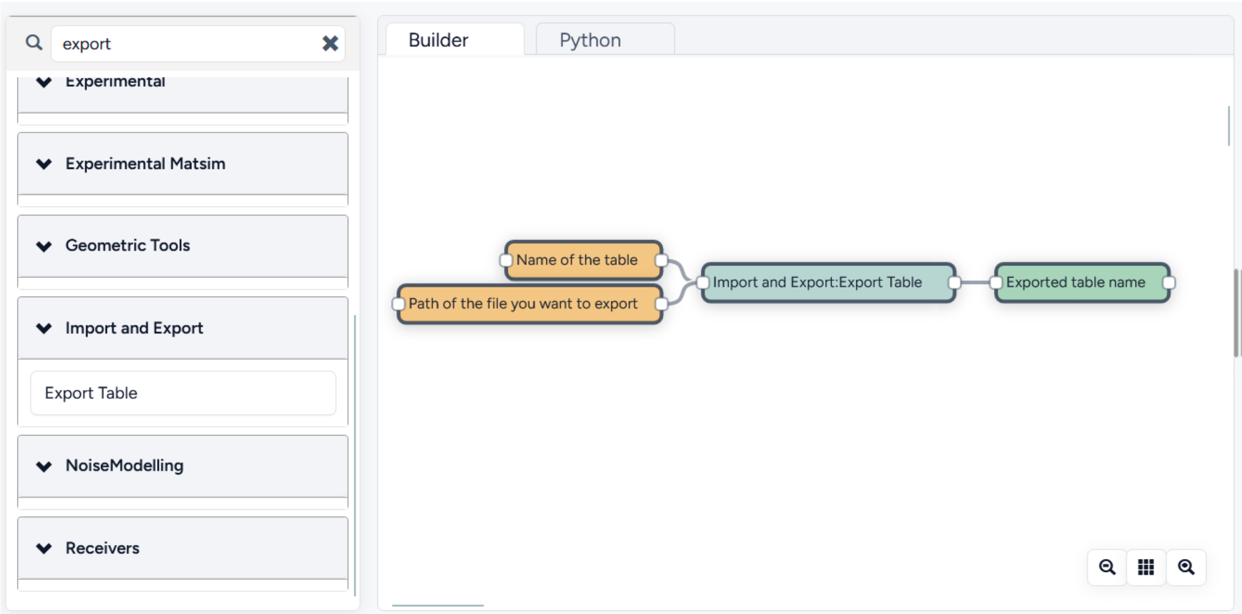
- In `Layer Add Layer Add vector layer`, you can enter the path of your `results.shp` file. Then click on `Add`.
- In `Layer Add Layer Add XYZ Layer`, you can add the OpenStreetMap background.

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

We now need to choose a time-slice we want to visualize, let's pick the timeBin of 10h (36000 seconds). 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 :

```
PERIOD = 36000
```



The last step is to color the dots based on the LEQA field. The following configuration can be used:

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

Going Further

Now maybe we just want to compute actual noise maps instead of just the noise levels at some specific points. In this case we'd want to use a different receiver grid using the `Delaunay_Grid` bloc.

Then we can use the same `Noise_level_from_source` blocs to calculate the noise levels at every receiver at every timestep.

You can then run a `Create_Isosurface` bloc to create a noise map, it will also automatically detect the PERIOD and produce one noise map per time bin.

Here is an example of a noise map for the 10h to 10h15 time period:

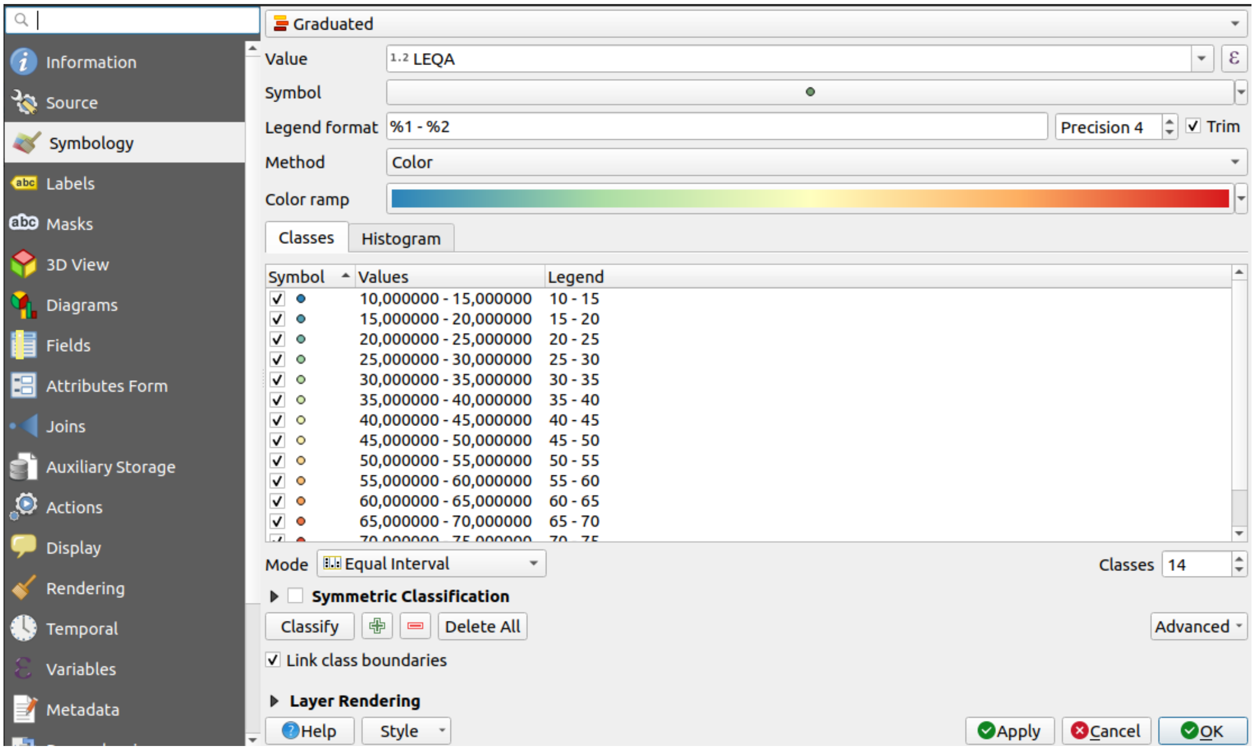
5.20 Dynamic Maps - GUI

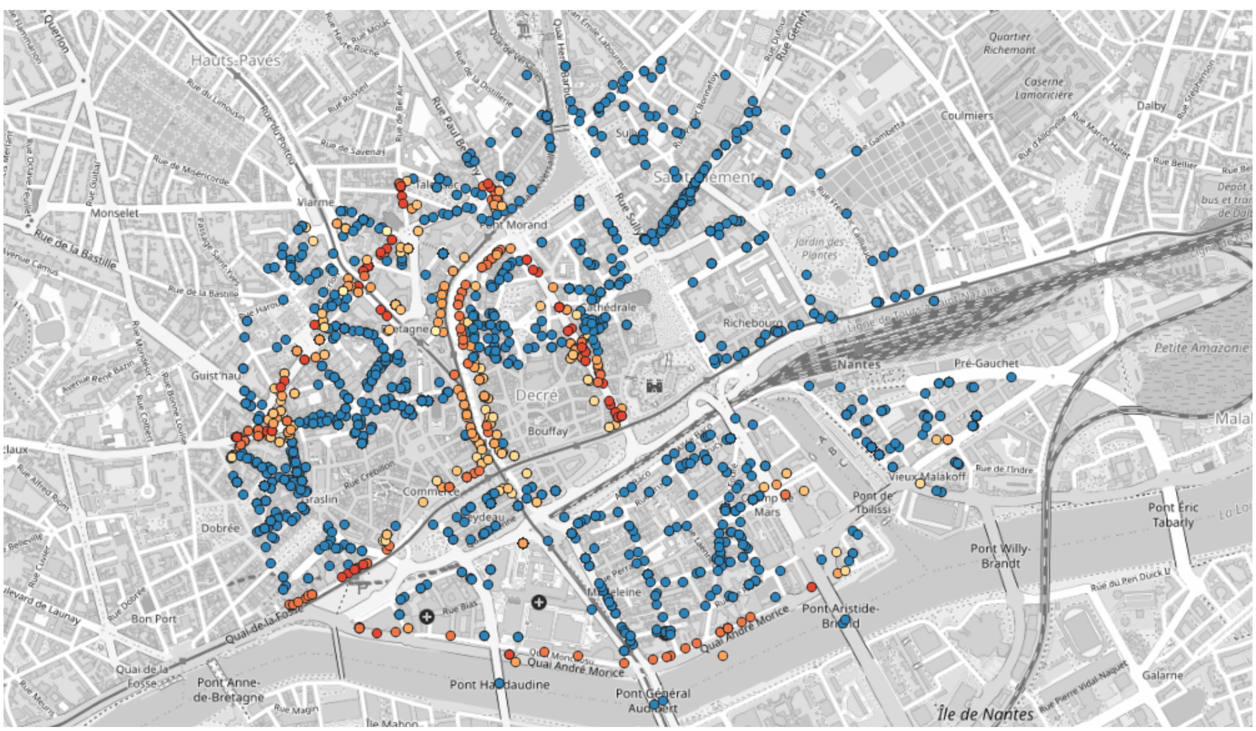
Many publications have emerged showcasing the use of **NoiseModelling** to create dynamic maps (see scientific production).

If you'd like to achieve similar results but you feel a bit lost, this tutorial is here to help you navigate through the process.

There are three main approaches to creating dynamic maps using NoiseModelling:

1. **A road network with a single traffic flow** You have a road network and a single traffic flow associated with a specific time period (e.g., 24h). You want to compute dynamic indicators such as **L10**, **L90**, or the **number of events exceeding a threshold** or to get time series for the same time period.
2. **A road network with traffic flows at regular intervals** You have a road network and traffic flow data available at regular intervals (e.g., hourly or every 15 minutes), and you want to generate a dynamic noise map every 15 min.


Information
Source
Symbology
Labels
Masks
3D View
Diagrams
Fields
Attributes Form
Joins
Auxiliary Storage
Actions
Display
Rendering
Temporal
Variables
Metadata
Graduated
Value
1.2 LEQA
Symbol
Legend format
%1 - %2
Precision 4
Trim
Method
Color
Color ramp
Classes
Histogram
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
Classify
Delete All
Advanced
Link class boundaries
Layer Rendering
Help
Style
Apply
Cancel
OK


Hauts-Pavés
Quartier Richemont
Caserne Lamoricière
Coulmiers
Dalby
Monselet
Rue de la Bastille
Pont Morand
Jardin des Plantes
Cathédrale
Richebourg
Nantes
Pré-Gauchet
Petite Amazonie
Decré
Bouffay
Guist'hau
Graslin
Commerce
Dobrée
Vieux Malakoff
Pont de Tbilissi
Pont Éric Tabarly
Pont Willy-Brandt
Bon Port
Pont Anne-de-Bretagne
Pont Haudaudine
Pont Général Audibert
Quai André Morice
Pont Aristide-Briand
Île de Nantes
Galarne
Madeleine

3. **A road network with associated spatio-temporal data of moving sources** You have spatio-temporal information about vehicles moving around a network (e.g., from traffic simulations such as Symuvia or SUMO; or from trajectories of drones). You want to compute **time-series data at each receiver** corresponding to the passage of these sources.

5.20.1 A word of caution

Warning: In all these cases, we assume that the **sound attenuation between the source and receiver remains constant throughout the calculation**. This is a strong approximation, and there are ways to account for variations, but this tutorial will not cover such specific cases.

Dynamic mapping has its subtleties, and it's important to be aware of them to avoid errors. We recommend referring to the following documents for a better understanding of these concepts:

- Can, A., & Aumond, P. (2018). Estimation of road traffic noise emissions: The influence of speed and acceleration. Transportation Research Part D: Transport and Environment, 58, 155-171.
- Gozalo, G. R., Aumond, P., & Can, A. (2020). Variability in sound power levels: Implications for static and dynamic traffic models. Transportation Research Part D: Transport and Environment, 84, 102339.
- Le Bescond, V., Can, A., Aumond, P., & Gastineau, P. (2021). Open-source modeling chain for the dynamic assessment of road traffic noise exposure. Transportation Research Part D: Transport and Environment, 94, 102793.

Assumptions are freely made, specific formats are expected, and so on. To understand the required data formats and check the expected structure of the input tables, please refer also to the example input tables and spatial layers!

5.20.2 Case 1: A Road Network with a Single Traffic Flow

Import the road network (with arbitrary traffic flows) and buildings from an OSM file

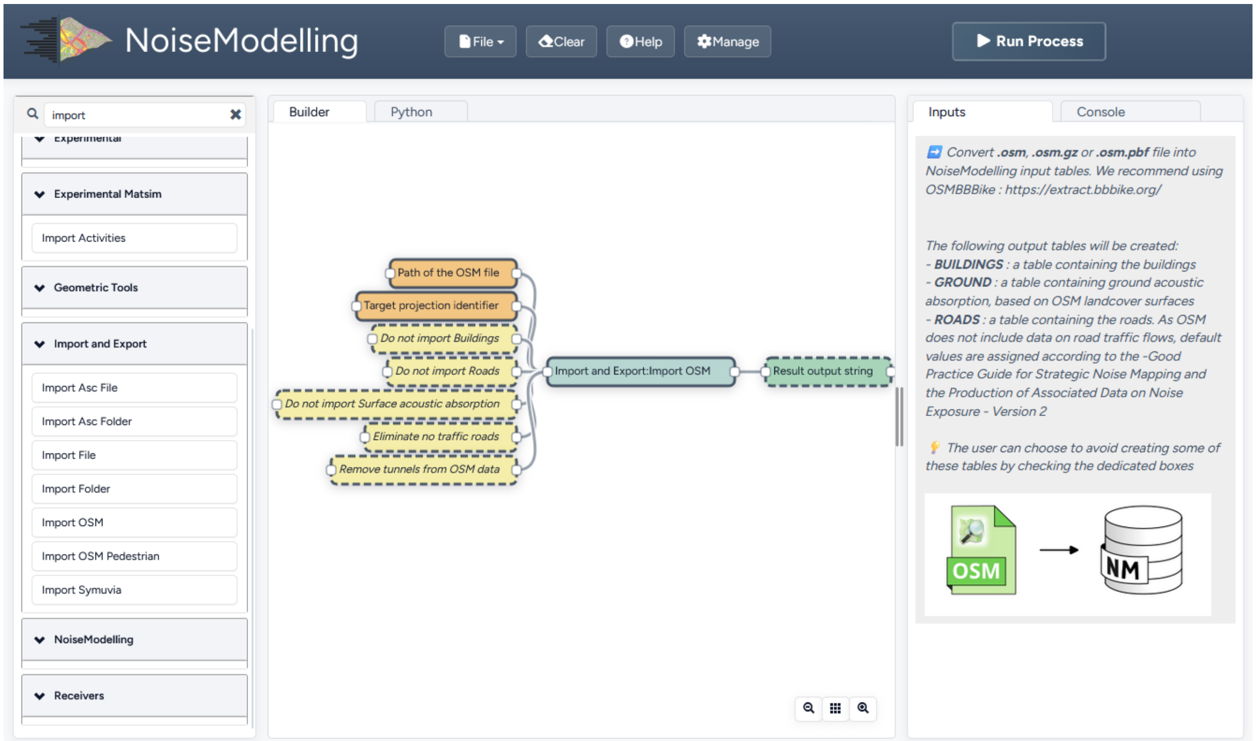
Use `Import_OSM` Block

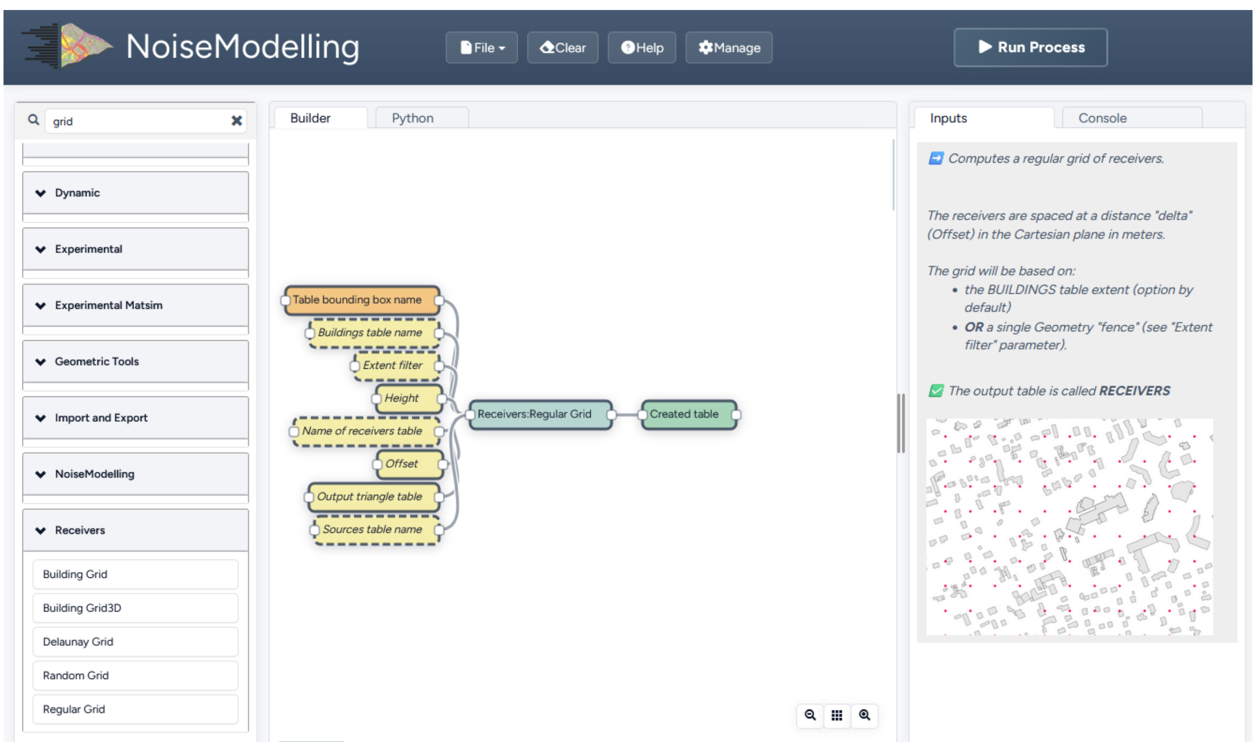
1. `Path of the osm file`: Enter the path of the provided Open Street map file (can be relative to NoiseModelling): `resources/map.osm.gz`
2. `Target projection identifier`: Enter the official France projection for this tutorial files `2154`
3. `Remove tunnels`: Check it
4. `Do not import surface`: Check it as we will not use this output

Create a receiver grid using 25 meters step in a grid pattern

Use `Regular_Grid` Block

1. `Table bounding box name`: Enter `ROADS` The receivers will use the envelope of the ROADS table.
2. `Offset`: Enter `15` for 15 meters distance
3. `Output triangle table`: Check it in order to be able to generate the iso contours
4. `height`: Enter `1.5`


NoiseModelling
File
Clear
Help
Manage
Run Process
import
Experimental Matsim
Import Activities
Geometric Tools
Import and Export
Import Asc File
Import Asc Folder
Import File
Import Folder
Import OSM
Import OSM Pedestrian
Import Symuvia
NoiseModelling
Receivers
Builder
Python
Path of the OSM file
Target projection identifier
Do not import Buildings
Do not import Roads
Do not import Surface acoustic absorption
Eliminate no traffic roads
Remove tunnels from OSM data
Import and Export:Import OSM
Result output string
Inputs
Console
Convert .osm, .osm.gz or .osm.pbf file into NoiseModelling input tables. We recommend using OSMBBBike : https://extract.bbbike.org/
The following output tables will be created:
- BUILDINGS : a table containing the buildings
- GROUND : a table containing ground acoustic absorption, based on OSM landcover surfaces
- ROADS : a table containing the roads. 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
The user can choose to avoid creating some of these tables by checking the dedicated boxes
OSM
NM


NoiseModelling
File
Clear
Help
Manage
Run Process
grid
Dynamic
Experimental
Experimental Matsim
Geometric Tools
Import and Export
NoiseModelling
Receivers
Building Grid
Building Grid3D
Delaunay Grid
Random Grid
Regular Grid
Builder
Python
Table bounding box name
Buildings table name
Extent filter
Height
Name of receivers table
Offset
Output triangle table
Sources table name
Receivers:Regular Grid
Created table
Inputs
Console
Computes a regular grid of receivers.
The receivers are spaced at a distance "delta" (Offset) in the Cartesian plane in meters.
The grid will be based on:
• the BUILDINGS table extent (option by default)
• OR a single Geometry "fence" (see "Extent filter" parameter).
The output table is called RECEIVERS

Convert traffic to dynamic traffic flow

From the network with traffic flow to individual trajectories with associated Lw

1. The Probabilistic method, this method places randomly the vehicles on the network according to the traffic flow
2. The Poisson method places the vehicles on the network according to the traffic flow following a poisson law, it keeps a coherence in the time series of the noise level

Use the `Dynamic:Flow_2_Noisy_Vehicles` Block:

1. `Method`: Select `TNP` to use the Poisson method
2. `Roads table name`: Enter `ROADS`
3. `timestep`: Enter `1`
4. `duration`: Enter `60`
5. `gridStep`: Enter `10`

Compute noise level at receiver points for each receiver-period combination

Use the `NoiseModelling:Noise_level_from_source` Block

1. `Buildings table name`: Enter `BUILDINGS`
2. `Source geometry table name`: Enter `SOURCES_GEOM` Contains only the geometries of the sources (points)
3. `Source emission table name`: Enter `SOURCES_EMISSION` Contains for each source index and period the noise emission
4. `Receivers table name`: Enter `RECEIVERS`
5. `Max Error (dB)`: Enter `3` Will skip further sources, reduces the computation time for this tutorial
6. `Maximum source receiver distance`: Enter `800`
7. `Diffraction on horizontal edges`: Check it
8. `Order of reflexion`: Enter `0`

Compute noise indicators

This step is optional, it computes the LA10, LA50 and LA90 time-aggregated indicators at each receiver from the table RECEIVERS_LEVEL

Use the `Acoustic_Tools:DynamicIndicators` WPS Block

1. `tableName`: Enter `RECEIVERS_LEVEL`
2. `columnName`: Enter `LAEQ`

Compute iso-surfaces for each time period

Generate a dynamic iso-contour map for each time period based on the LAEQ of the receivers.

Use the `Acoustic_Tools:Create_Isosurface` WPS Block

1. `Sound levels table`: Enter `RECEIVERS_LEVEL`
2. `Polygon smoothing coefficient`: Enter `0`

Export Map to QGis

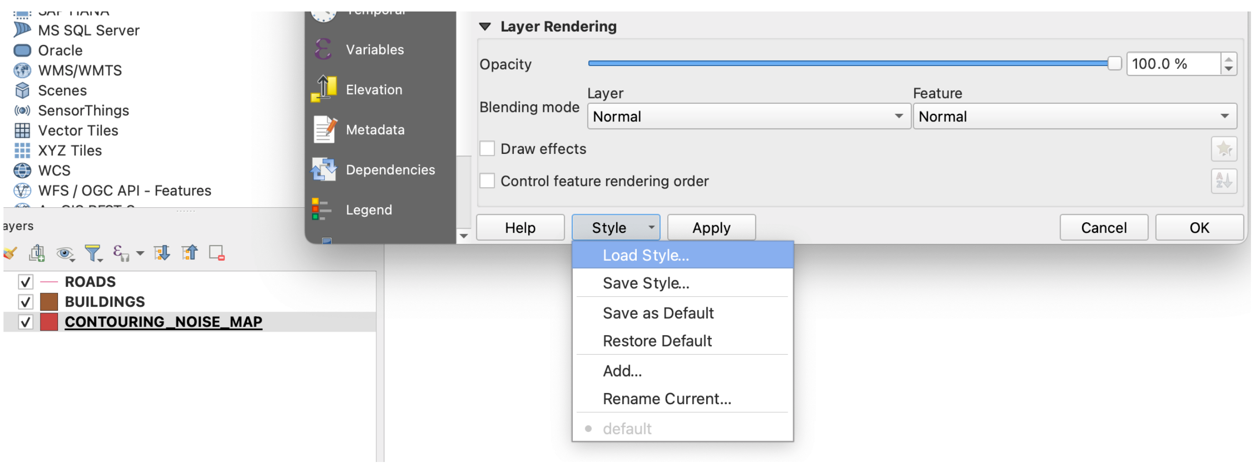
Use `Export_Table` Block to export the following tables as files in any folder.

1. `CONTOURING_NOISE_MAP`
2. `BUILDINGS`
3. `ROADS`

Configure QGis to display time dependant map

Load the 3 files in QGIS. `Contouring_noise_map` must be ordered as the last layer (rendered in the bottom)

Load the style for contouring noise map:



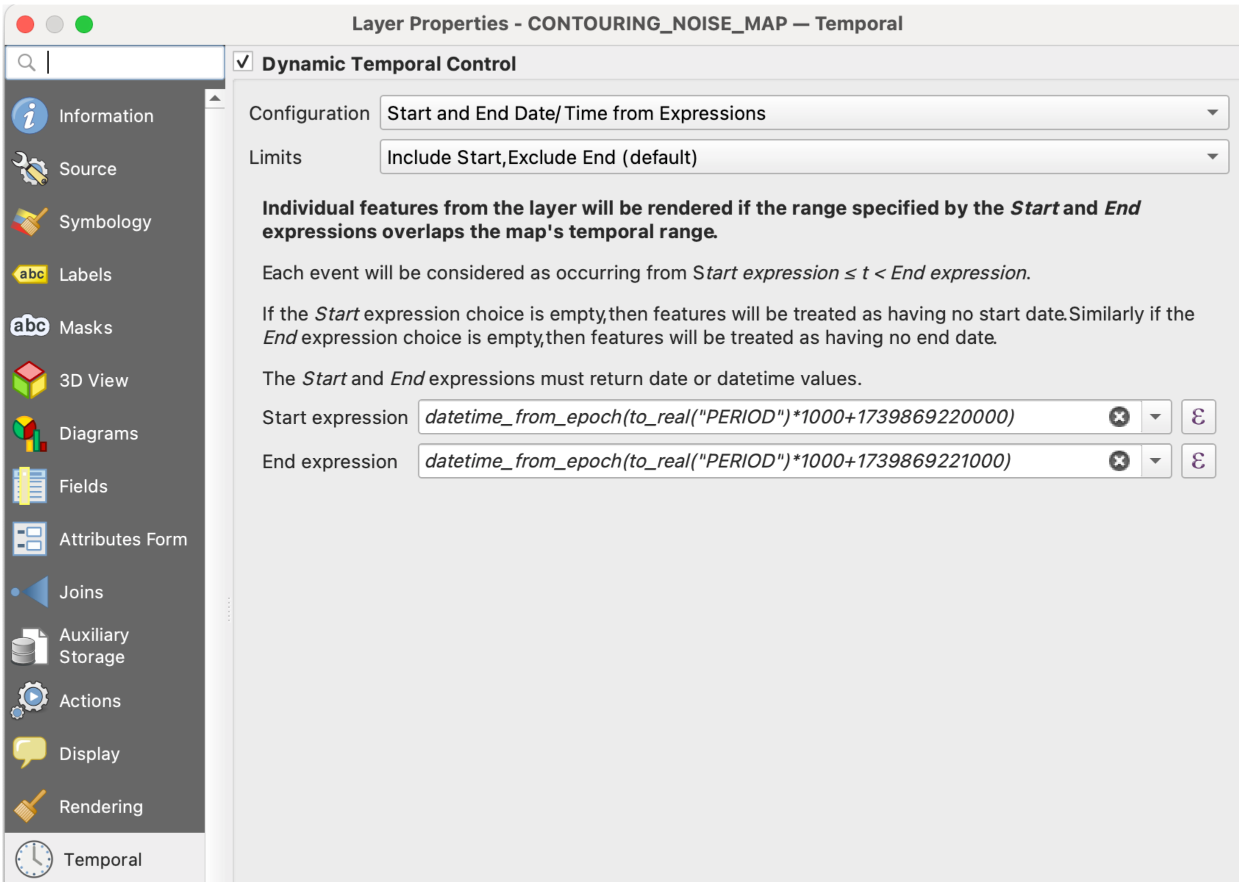
Load the style located in the NoiseModelling folder `Docs/styles/style_beate_tomio.sld`

In QGis, in time window, paste the following formulae:

Start expression: `datetime_from_epoch(to_real("PERIOD")*1000+1739869220000)` End expression: `datetime_from_epoch(to_real("PERIOD")*1000+1739869221000)`

Epoch is in millisecond, so we multiply by 1000 and add any base epoch time. The step end 1000 milliseconds after the start period.

With the navigation bar of QGis you can select the period to display.


Layer Properties - CONTOURING_NOISE_MAP — Temporal
Information
Source
Symbology
Labels
Masks
3D View
Diagrams
Fields
Attributes Form
Joins
Auxiliary Storage
Actions
Display
Rendering
Temporal
Dynamic Temporal Control
Configuration
Start and End Date/ Time from Expressions
Limits
Include Start,Exclude End (default)
Individual features from the layer will be rendered if the range specified by the Start and End expressions overlaps the map's temporal range.
Each event will be considered as occurring from Start expression ≤ t < End expression.
If the Start expression choice is empty,then features will be treated as having no start date.Similarly if the End expression choice is empty,then features will be treated as having no end date.
The Start and End expressions must return date or datetime values.
Start expression
datetime_from_epoch(to_real("PERIOD")*1000+1739869220000)
End expression
datetime_from_epoch(to_real("PERIOD")*1000+1739869221000)

5.20.3 Case 2: A Road Network with Traffic Flows at Regular Intervals

This case is similar to the **MATSim** use case (here), but this tutorial generalizes the approach to fit other datasets.

This sample dataset used in this example was kindly provided by Valentin Le Bescond from Université Gustave Eiffel.

Import Buildings for your study area

Use `Import File` Block

1. `Path of the input File`: Enter the path of building (can be relative to NoiseModelling): `resources/Dynamic/Z_EXPORT_TEST_BUILDINGS.geojson`
2. `Projection identifier`: Enter SRID `2154`
3. `Output table name`: Enter `buildings`

Import the road network

Use `Import File` Block

1. `Path of the input File`: Enter the path of building (can be relative to NoiseModelling): `resources/Dynamic/Z_EXPORT_TEST_TRAFFIC.geojson`
2. `Projection identifier`: Enter SRID `2154`
3. `Output table name`: Enter `roads`

Create a receiver grid using 25 meters step in a grid pattern

Use `Regular_Grid` Block

1. `Table bounding box name`: Enter `ROADS` The receivers will use the envelope of the `ROADS` table.
2. `Offset`: Enter `25` for 25 meters distance
3. `height`: Enter `1.5`

Split geometry and traffic periods

In the table `ROADS`, the traffic information is given for each period in the `PERIOD` column.

The following Block aggregates roads by the geometry and places the associated pair `IDSOURCE/PERIOD` with the corresponding road traffic into the `SOURCES_EMISSION` table.

Use the Block `Dynamic::Split_Sources_Period` :

1. `Source table name`: Enter `ROADS`
2. `Source index field name`: Enter `LINK_ID`
3. `Source period field name`: Enter `PERIOD`.

Two output tables are created `SOURCES_GEOM` and `SOURCES_EMISSION`

Compute noise level at receiver points for each receiver-period combination

Use the `NoiseModelling:Noise_level_from_source` Block

1. `Buildings table name`: Enter `BUILDINGS`
2. `Source geometry table name`: Enter `SOURCES_GEOM` Contains only the geometries of the sources (points)
3. `Source emission table name`: Enter `SOURCES_EMISSION` Contains for each source index and period the noise emission
4. `Receivers table name`: Enter `RECEIVERS`
5. `Diffraction on horizontal edges`: Check it
6. `Order of reflexion`: Enter `0`

The `RECEIVERS_LEVEL` output table is created.

Compute noise indicators

This step is optional, it computes the LA10, LA50 and LA90 time-aggregated indicators at each receiver from the table `RECEIVERS_LEVEL`.

Use the `Acoustic_Tools:DynamicIndicators` WPS Block

1. `tableName`: Enter `RECEIVERS_LEVEL`
2. `columnName`: Enter `LAEQ`

Visualizing results

The result table `RECEIVERS_LEVEL` can be displayed in QGis, if you filter by `PERIOD`.

5.20.4 Case 3: Spatio-Temporal Data of Moving Sources

This sample dataset was kindly provided by Sacha Baclet from KTH (See his ORCID).

Import Buildings for your study area

Use `Import File` Block

1. `Path of the input File`: Enter the path of building (can be relative to NoiseModelling): `resources/Dynamic/buildings_nm_ready_pop_heights.shp`
2. `Projection identifier`: Enter SRID `32635`
3. `Output table name`: Enter `buildings`

Import the receivers (or generate your set of receivers using Regular_Grid script for example)

Use `Import File` Block

1. `Path of the input File`: Enter the path of building (can be relative to NoiseModelling): `resources/Dynamic/receivers_python_method0_50m_pop.shp`
2. `Projection identifier`: Enter SRID 32635
3. `Output table name`: Enter `receivers`

Import the road network

Use `Import File` Block

1. `Path of the input File`: Enter `resources/Dynamic/network_tartu_32635_.geojson`
2. `Projection identifier`: Enter SRID 32635
3. `Output table name`: Enter `network_tartu`

Add primary key column to the road network (Optional)

Use `Add_Primary_Key` Block

1. `Name of the column`: Enter PK
2. `Name of the table`: Enter SRID `network_tartu`

Import the vehicle trajectories

Use `Import File` Block

1. `Path of the input File`: Enter `resources/Dynamic/SUMO.geojson`
2. `Projection identifier`: Enter SRID 32635
3. `Output table name`: Enter `vehicle`

Create point sources from the network every 10 meters

This point source will be used to compute the noise attenuation level from them to each receiver. The created table will be named `SOURCES_GEOM`.

Use `Point_Source_From_Network` Block

1. `Input table name`: Enter `network_tartu`
2. `gridStep`: Enter SRID `10`

Create a table with the noise level from the vehicles and snap the vehicles to the point sources

Use `Ind_Vehicles_2_Noisy_Vehicles` Block

1. `Source geometry table`: Enter `SOURCES_GEOM`
2. `Individual Vehicles table`: Enter `vehicle`
3. `Snap distance`: Enter `30` This is the maximal distance (m) to reattach individual vehicle positions to the source points
4. `Vehicles table format`: Enter `SUMO`

Compute noise attenuation for each receiver-source pairs

Unlike the previous tutorial we will use an alternative approach here by storing the attenuation between all sources and receivers first.

This attenuation will be applied later to the emission levels for each period.

Use the `NoiseModelling:Noise_level_from_source` Block

1. `Buildings table name`: Enter `BUILDINGS`
2. `Source geometry table name`: Enter `SOURCES_GEOM` Contains only the geometries of the sources (points)
3. `Receivers table name`: Enter `RECEIVERS`
4. `Maximum source receiver distance`: Enter `300`
5. `Diffraction on horizontal edges`: Check it
6. `Order of reflexion`: Enter `0`
7. `Separate receiver level by source identifier`: Check it to have the SOURCEID column on the output

Apply attenuation on emission levels

Compute the noise level from the moving vehicles to the receivers. The output table is called here `LT_GEOM` and contains the time series of the noise level at each receiver.

Use the `Dynamic:Noise_From_Attenuation_Matrix` Block

1. `LW(PERIOD)`: Enter `SOURCES_EMISSION`
2. `Attenuation Matrix Table name`: Enter `RECEIVERS_LEVEL`
3. `outputTable Matrix Table name`: Enter `LT_GEOM`

Compute noise indicators

This step is optional, it computes the LA10, LA50 and LA90 time-aggregated indicators at each receiver from the table `LT_GEOM`

Use the `Acoustic_Tools:DynamicIndicators` WPS Block

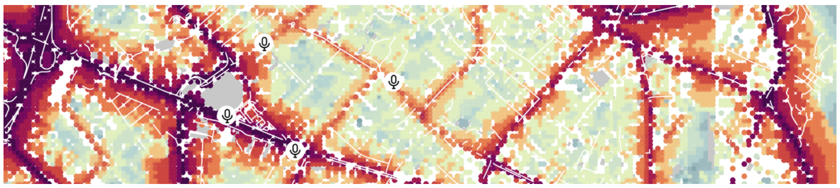
1. `tableName`: Enter `LT_GEOM`
2. `columnName`: Enter `LAEQ`

Visualizing results

The result table LT_GEOM can be displayed into QGis, if you filter by PERIOD.

Note: All this tutorial done with Groovy is written on this unit test source code: Github source

5.21 Data Assimilation



5.21.1 Introduction

Data assimilation is a technique that combines observations with a numerical model to improve the accuracy of forecasts or analyses. Applied to acoustics and noise maps, data assimilation makes it possible to integrate real noise measurements (*e.g* coming from sensors) into noise propagation models to produce more accurate and reliable noise maps.

In this tutorial, we will see how to model and simulate noise in Geneva using static (road network, buildings) and dynamic (traffic, temperature) data. The main objective is to **combine measurements from sensors** located in Geneva **with acoustic simulations** to **identify traffic configurations that best match measurements**.

5.21.2 Requirements

To play with this tutorial, you will need:

- a working installation of NoiseModelling (NM) with at least version 5. If needed, get the last release on the official GitHub repository,
- the tutorial's datasets, stored in the folder `.../NoiseModelling/resources/dataAssimilation/`,
- the dedicated Groovy scripts (Blocks), stored in the folder `.../NoiseModelling/scripts/DataAssimilation/`.

5.21.3 The data

In the the tutorial's folder, you have the following data:

1. sensor measurement `.csv` files, stored in the `devices_data/` folder (One file per sensor),
2. `device_mapping_sf.geojson` containing two columns : the point geometry (`THE_GEOM`) and a sensor's unique identifier (`DEVEUI``). This file is also provided in `.csv` format,
3. the OpenStreetMap (OSM) dataset of the studied area : `geneva.osm.pbf`.

Sensor measurements

Each `.csv` files in the folder `devices_data/` contains environmental noise measurements recorded by individual sensors. The columns are:

- `deveui` : Unique identifier of the sensor,
- `epoch` : Time of measurement (Epoch format - Unix time, ex:1724567400),
- `Leq` : Equivalent continuous sound level in dB(A), calculated over a period (15 min),
- `Temp` : Temperature (°C) recorded by the sensor at the time of measurement,
- `timestamp` : Time of measurement (timestamp format `"YYYY-MM-DD HH:MM:SS"`).

Below is an illustration with the file `4a6.csv`. Here we can see that 5 measurements were taken, every 15 minutes, between 6:30 and 7:30 am. During this period, noise levels rose slightly, while temperatures remained stable overall (except for the first measurement).

Table 3: Informations stored in the sensor's file `4a6.csv`

deveui	epoch	Leq	Temp	timestamp
4a6	1724567400	40.6	10.00	2024-08-25 06:30:00
4a6	1724568300	41.9	18.125	2024-08-25 06:45:00
4a6	1724569200	43.9	18.125	2024-08-25 07:00:00
4a6	1724570100	42.3	18.125	2024-08-25 07:15:00
4a6	1724571000	45.5	18.125	2024-08-25 07:30:00

Warning: In this tutorial all the values, coming from the 7 sensors, are sampled approximately every 15 minutes, but the exact spacing may vary slightly.

We could also have had data with a time step of 5 minutes, 10 minutes, 1 hour, etc.

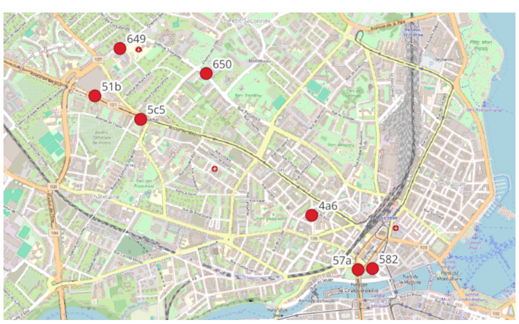
In all cases, it is important that all the measurements from the sensors are set to the same time step AND that they are phased (i.e. set to the same time intervals).

Sensor localization

The `device_mapping_sf.geojson` columns are:

- `deveui` : Unique Identifier of the sensor,
- `the_geom` : 3D point geometry in WKT (Well-Known Text) format — includes coordinates (X, Y) and altitude (Z) in the projected coordinate system.

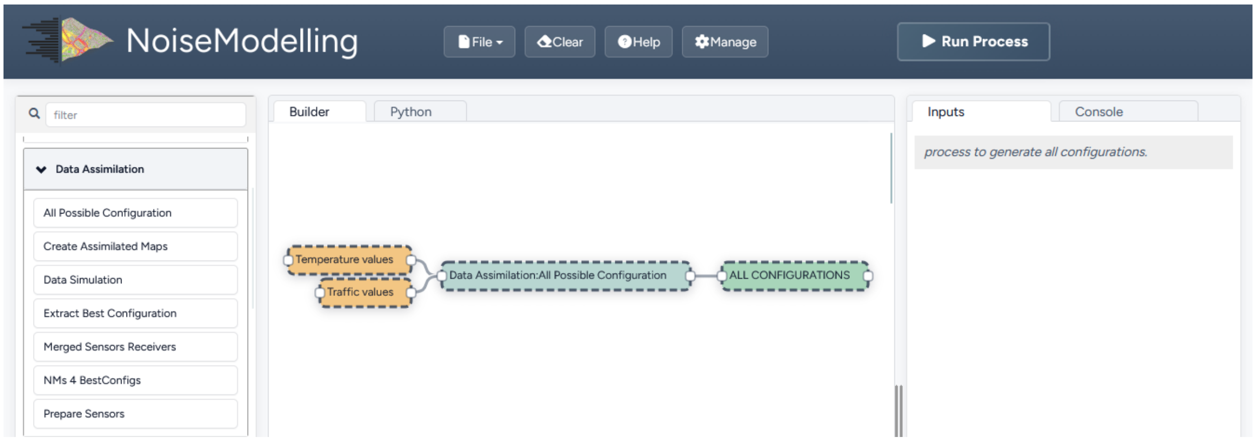
Below is a map, showing the seven sensors (red points), with their identifier `deveui`.



5.21.4 How to compute the assimilation?

To compute the data assimilation, you will have to execute several Groovy scripts (Blocks). You can play with them in two ways:

- with the NoiseModelling's GUI (Graphic User Interface). In this case, the Blocks are listed in the `Data_Assimilation` tab *(see screenshot below)*,
- in command line (see how to *NoiseModelling client line interface (CLI)*). In this case, just note that they are stored in the folder `.../NoiseModelling_x.x.x/scripts/DataAssimilation/`,
- in a Groovy script (Block), calling one or various Groovy scripts (Blocks).



5.21.5 Data Simulation

This process prepares the training dataset from sensor measurements, importing it into NoiseModelling (in a spatial database) and performing various calculations to determine noise levels.

Maps are generated with all possible combinations, in order to identify the best road configuration in terms of noise levels generated.

Step 1 : Generate all possible combinations

Generates all possible combinations of values from two given lists and inserts them into a table named `ALL_CONFIGURATIONS`.

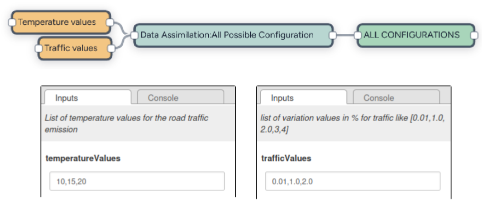
To calculate the combinations, you have to execute the script `All_Possible_Configuration`.

Two input parameters are needed:

- **Traffic values** (`trafficValues`): variation around standard values for the 4 types of roads: primary, secondary, tertiary and others. When generating a variation of the default map (the set of maps), the values taken by the primary, the secondary, … sections, will be calculated from these parameters (between 1 and *n*). For example, if `"trafficValues": "0.01, 1"`, then the primary sections will take 1% or 100% of their default value. The same applies to the remaining road types.
- **Temperature values** (`temperatureValues`) : the temperatures (in °C - Double). The 1 to *n* possible temperatures to play (in the example below, 3 temperatures are defined: 10, 15 and 20°C)

Execute `All_Possible_Configuration` Block

With the **NoiseModelling GUI**



With **command lines**

```
./bin/wps_scripts -w ./ -s scripts/Data_Assimilation/All_Possible_Configuration.groovy -trafficValues "0.01,1,2" -temperatureValues "10,15,20"
```

With **Groovy script (Block)**

```
new All_Possible_Configuration().exec(connection,[
               "trafficValues": "0.01,1.0,2.0",
               "temperatureValues": "10,15,20"
])
```

Result

The generated combinations include values for type of roads primary, secondary, tertiary, others, and temperature. The resulting table `ALL_CONFIGURATIONS` has the following columns:

- `IT`: Unique identifier of the combination (Primary Key - Integer)
- `PRIMARY_VAL`: percentage of primary roads traffic, given by trafficValues (Double)
- `SECONDARY_VAL`: … of secondary roads … (Double)
- `TERTIARY_VAL`: … of tertiary roads … (Double)
- `OTHERS_VAL`: … of other roads … (Double)
- `TEMP_VAL`: temperature (Double)

The first 10 lines of this table are shown below:

IT	PRIMARY_VAL	SECONDARY_VAL	TERTIARY_VAL	OTHERS_VAL	TEMP_VAL
1	0.01	0.01	0.01	0.01	10
2	0.01	0.01	0.01	0.01	15
3	0.01	0.01	0.01	0.01	20
4	1.0	0.01	0.01	0.01	10
5	1.0	0.01	0.01	0.01	15
6	1.0	0.01	0.01	0.01	20
7	1.0	1.0	0.01	0.01	10
8	1.0	1.0	0.01	0.01	15
9	1.0	1.0	0.01	0.01	20
10	1.0	1.0	1.0	0.01	10

Warning: The total number of combinations can be huge. This value is defined as: (number of `trafficValues` elements) ^ 4 * (number of `temperatureValues` elements).

In our example, we have `"trafficValues": "0.01, 1.0, 2.0"` and `"temperatureValues": "10,15, 20"`, so the number of combinations = 3 ^ 4 * 3 = 243 *(before filtration - see 'note' tab just below)*

Even though this table is very important, only part of it will be used for all the maps to be simulated (see *Step 5*)

Note: When the combinations are calculated, **a filter is applied to remove inconsistent pairs** (e.g. an 'other' type road with much more traffic than a 'primary' road).

As a result, only pairs meeting the following rule are retained: `traffic on a lower type of road / traffic on a higher type of road <= 20`

This rule eliminates cases where a road of a lower type would have 20 times more traffic than another of a higher type.

Step 2 : Import sensor positions

Using the `Import_File` Block*, import the location of the sensors into the NoiseModelling's database from the .geojson file `device_mapping_sf.geojson`. This file will be stored in a table called `SENSORS_LOCATION`.

* in the `scripts/Import_and_Export/` folder

Since we are in the Geneva area, we are using the `CH1903+` metric coordinate system (identified as EPSG:2056).

- `pathFile` : `./resources/dataAssimilation/device_mapping_sf.geojson`
- `inputSRID` : `2056`
- `tableName` : `SENSORS_LOCATION`

If you are using the Groovy script (Block)

```
new Import_File().exec(connection,[
                "pathFile" : workingFolder+"device_mapping_sf.geojson",
                "inputSRID" : 2056,
                "tableName": "SENSORS_LOCATION"
])
```

Result

Once done, you have the table `SENSORS_LOCATION`, presented below.

THE_GEOM	DEVEUI
POINT Z(2499748.01396418 1118023.64763599 4)	57a
POINT Z(2497926.50847343 1119715.69600241 4)	649
POINT Z(2499855.07784477 1118032.40228679 4)	582
POINT Z(2497737.00785445 1119351.14658559 4)	51b
POINT Z(2499392.11210506 1118441.15233318 4)	4a6
POINT Z(2498584.3443027 1119522.320612 4)	650
POINT Z(2498084.77671566 1119172.90847018 4)	5c5

Step 3 : Prepare sensor data

Now we can extract and prepare the sensors, for a given period. To do so, we are using the `Prepare_Sensors` Groovy script (Block) stored in the folder `.../scripts/DataAssimilation/`.

This script has the following parameters:

- `startDate` : the start timestamp for the dataset extraction (in format `YYYY-MM-DD HH:MM:SS`)
- `endDate` : the final timestamp for the dataset extraction (in format `YYYY-MM-DD HH:MM:SS`)
- `trainingRatio` : define the percentage of data to be used for training (e.g a value of 0.7 means that 70% of the data will be used for training. The remaining 30% will be used for validation) (Double)
- `workingFolder` : folder containing the file `device_mapping_sf.csv`, the OSM file and the `devices_data` folder.
- `targetSRID`: Target projection identifier (also called SRID) of your project (Integer)

Execution

For this tutorial, you can fill with these informations:

- Start time stamp (`startDate`) : `2024-08-25 06:30:00`
- End time stamp (`endDate`) : `2024-08-25 07:30:00`
- Training data percentage(`trainingRatio`) : `0.8`
- Working directory path with input files (`workingFolder`) : `./resources/dataAssimilation/` (enter the full URL e.g `/home/myUserName/Documents/NoiseModelling/resources/dataAssimilation/`)
- Target projection identifier (`targetSRID`) : `2056`

If you are using the Groovy script (Block)

```
new Prepare_Sensors().exec(connection,[
                "startDate":"2024-08-25 06:30:00",
                "endDate": "2024-08-25 07:30:00",
                "trainingRatio": 0.8,
                "workingFolder": "./resources/dataAssimilation/",
                "targetSRID": 2056
])
```

Result

Once executed, two tables are created:

- `SENSORS_MEASUREMENTS` : representing all the data for this period, with sensor's position (POINT geometry)
- `SENSORS_MEASUREMENTS_TRAINING` : filtering `SENSORS_MEASUREMENTS` to keep only the training dataset. This table (described below) will be used as the RECEIVER table for the following steps.
  - `IDSENSOR` : Unique identifier of the sensor,
  - `THE_GEOM` : The sensor's position (POINT geometry),
  - `IDRECEIVER` : The receiver's unique Id
  - `EPOCH` : Time of measurement (Epoch format - Unix time, ex:1724567400),

- LAEQ : Equivalent continuous sound level in dB(A), calculated over a period (15 min),
- TEMP : Temperature (°C) recorded by the sensor at the time of measurement.

Step 4: Import buildings and roads

Now, using the Import_OSM Block, you can import buildings and road network (with predicted traffic flows) from the geneva.osm.pbf OSM file.

Execution

- Path of the OSM file (pathFile): ./resources/dataAssimilation/geneva.osm.pbf
- Target projection identifier (targetSRID): 2056
- Do not import Surface acoustic absorption (ignoreGround): true
- Remove tunnels from OSM data (removeTunnels): true

The other parameters are left as default

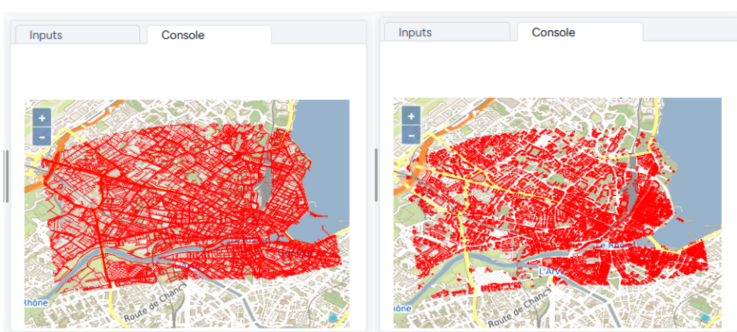
If you are using the Groovy script (Block)

```
new Import_OSM().exec(connection, [
                "pathFile"      : "./resources/dataAssimilation/geneva.osm.pbf",
                "targetSRID"    : 2056,
                "ignoreGround"  : true,
                "removeTunnels" : true
])
```

Result

The tables BUILDINGS and ROADS are created.

Note: If you are using NoiseModelling with the GUI and wish to visualize the data in a map, you can use the Table_Visualization_Map Block. Below is the result with the two table BUILDINGS *(left)* and ROADS *(right)*.



Step 5 : Generate all traffic emissions and maps

This step consists in generating all the traffic emissions by modifying traffic data according to the road type, using data from `ALL_CONFIGURATIONS` (see *Step 1*).

1. Generate emissions

To do so, users have first to execute the `Data_Simulation` Block, which has only one **optional** parameter :

- `noiseMapLimit` : final number of maps to be generated (%). If a value is filled, a random selection is applied to keep the percentage(%) of expected maps, based on the LHS (Latin Hypercube Sampling) method.

Execution

For this tutorial, you can fill with this information:

- `noiseMapLimit` : `80`

If you are using the Groovy script (Block)

```
new DataSimulation().exec(connection,[
                "noiseMapLimit": 80
])
```

Result

Two tables are created:

- `LW_ROADS` : table containing all traffic emissions
- `ROADS_GEOM` : table containing the geometry of roads

2. Calculate noise levels

Now, users can calculate the noise levels, emitted from the road sources. To do so, execute the `Noise_level_from_source` Block.

Execution

For this tutorial, you can fill with this information:

- Source geometry table name (`tableSources`): `ROADS_GEOM`
- Source emission table name (`tableSourcesEmission`): `LW_ROADS`
- Buildings table name (`tableBuilding`): `BUILDINGS`
- Receivers table name (`tableReceivers`): `SENSORS_LOCATION`,
- Separate receiver level by source id (`confExportSourceId`): `false`
- Maximum source-receiver distance (`confMaxSrcDist`): `250`
- Diffraction on vertical edges (`confDiffVertical`): `false`

- Diffraction on horizontal edges (`confDiffHorizontal`): `false`

If you are using the Groovy script (Block)

```
new Noise_level_from_source().exec(connection, [
                "tableSources": "ROADS_GEOM",
                "tableSourcesEmission" : "LW_ROADS",
                "tableBuilding": "BUILDINGS",
                "tableReceivers": "SENSORS_LOCATION",
                "confExportSourceId": false,
                "confMaxSrcDist": 250,
                "confDiffVertical": false,
                "confDiffHorizontal": false
])
```

Result

The table `RECEIVERS_LEVEL` is created and stores all the generated maps *(see the three first lines below - you can scroll the table to the right)*.

IDRECEIVER	PERIOD	THE_GEOM	HZ63	...	HZ8000	LAEQ	LEQ
1433	344	SRID=2056;POINT Z (2496912.2844730876 1116884.1525357629 4)	44.601295		14.591279	38.107693	46.40716
1433	448	SRID=2056;POINT Z (2496912.2844730876 1116884.1525357629 4)	46.617485		15.979266	40.203373	48.37574
1433	648	SRID=2056;POINT Z (2496912.2844730876 1116884.1525357629 4)	46.617485		15.979266	40.203373	48.37574

Step 6 : Extract best configuration

Many maps have been generated. So now, the best map, **minimizing the difference between the measurements and the simulation**, must be chosen.

To do so, users have to execute the `Extract_Best_Configuration` script. There are 3 parameters to enter here:

- Measurement table (`observationTable`): name of the table where observed data are stored,
- Noise map table (`noiseMapTable`): name of the table where simulated data are stored,
- Temperature tolerance threshold (`tempToleranceThreshold`): tolerance threshold (expressed in °C) for the temperature that allows to extract the map that have a temperature value close to the real temperature.

This process will:

- first determine the LAEQ difference between simulated (`noiseMapTable`) and observed (`observationTable`) values,
- then calculate, for each time steps and for all maps, the median value of the difference between the two values. **The map having the smallest median value will be the best one**.

Execution

For this tutorial, you can fill with this information:

- `observationTable` : `SENSORS_MEASUREMENTS_TRAINING`
- `noiseMapTable` : `RECEIVERS_LEVEL`
- `tempToleranceThreshold` : `5`

If you are using the Groovy script (Block)

```
new Extract_Best_Configuration().exec(connection,[
                "observationTable": "SENSORS_MEASUREMENTS_TRAINING",
                "noiseMapTable": "RECEIVERS_LEVEL",
                "tempToleranceThreshold"  : 5
])
```

Result

Note: The best configuration is found for each time step (here 15 minutes)

As a result, the `BEST_CONFIGURATION_FULL` table is created (see table below). This table is made of the following columns:

- `EPOCH` : Time of measurement (Epoch format - Unix time, ex:1724567400),
- `MIN_MEDIAN_DIFF` : the minimum median of the difference between simulated and measured LAEQs (this is how the best maps are chosen)
- `IT` : Unique identifier of the combination (Primary Key - Integer)
- `PRIMARY_VAL` : percentage of primary roads traffic, given by `trafficValues` (Double)
- `SECONDARY_VAL` : … secondary roads …
- `TERTIARY_VAL` : … tertiary roads …
- `OTHERS_VAL` : … other roads …
- `TEMP_VAL` : temperature (Integer)

Table 4: Table `BEST_CONFIGURATION_FULL` showing one best configuration for each of the time steps

EPOCH	MIN_MEDIAN_DIFF	IT	PRIMARY_VAL	SECONDARY_VAL	TERTIARY_VAL	OTHERS_VAL	TEMP_VAL
1724567400	9.3827	366	2.8	2.8	0.01	0.2	10
1724569200	11.2629	577	3.0	0.2	3.0	0.2	15
1724570100	11.2	577	3.0	0.2	3.0	0.2	15
1724571000	11.625	577	3.0	0.2	3.0	0.2	15
1724568300	11.1953	679	3.0	3.0	0.2	0.2	25
1724569201	7.8047	679	3.0	3.0	0.2	0.2	25
1724569202	10.3047	679	3.0	3.0	0.2	0.2	25

5.21.6 Execute Simulation: Generate the Dynamic Map

This part is made to execute a dynamic traffic calibration process, using the best configuration.

Step 7 : Generate new receivers

Create a regular grid of receivers between the buildings.

To do so, use the `Regular_Grid` script (in the `/Receivers/` scripts folder).

Execution

For this tutorial, you can fill with this information:

- Buildings table name (`buildingTableName`): `BUILDINGS`
- Table bounding box name (`fenceTableName`): `BUILDINGS`
- Source table name (`sourcesTableName`): `ROADS`
- Offset (`delta`): `200` (1 receiver every 200m)

If you are using the Groovy script (Block)

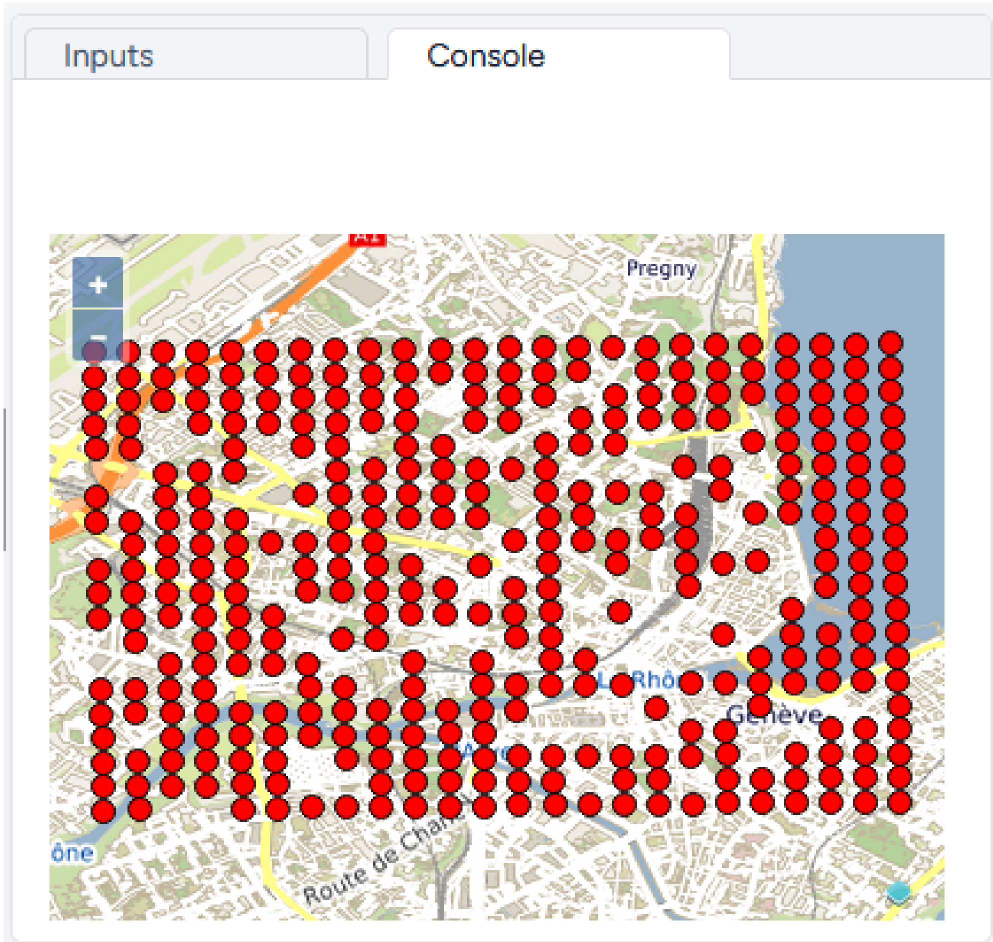
```
new Regular_Grid().exec(connection,[
              "buildingTableName": "BUILDINGS",
              "fenceTableName": "BUILDINGS",
              "sourcesTableName":"ROADS",
              "delta": 200
])
```

Result

The table `RECEIVERS` is created.

Table 5: Three first lines of the table `RECEIVERS`

THE_GEOM	ID_COL	ID_ROW	PK
POINT Z(2496682.2844730876 1116854.1525357629 4)	1	1	1
POINT Z(2496882.2844730876 1116854.1525357629 4)	2	1	2
POINT Z(2497482.2844730876 1116854.1525357629 4)	5	1	5



Step 8 : Adding sensors as receivers

Optionally, add the sensors location into the RECEIVERS table (just created before).

Execution

In the Merged_Sensors_Receivers Block, fill these information:

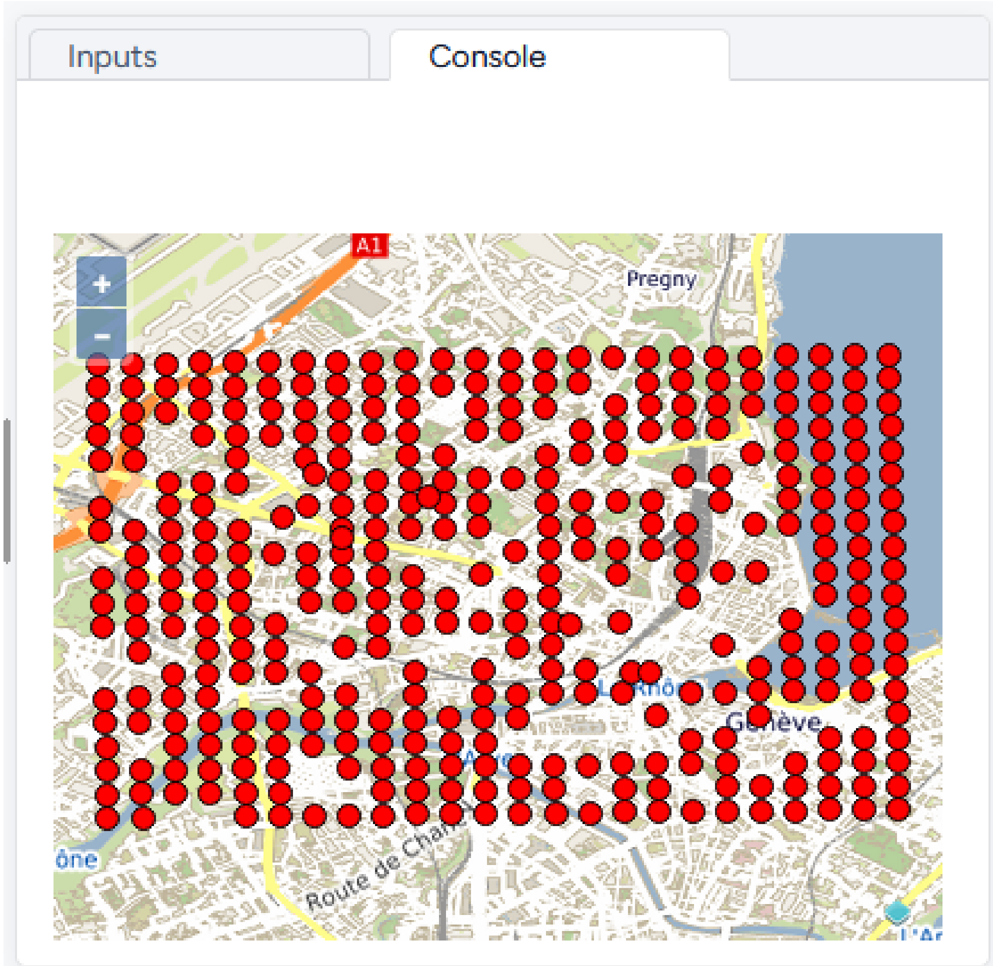
- The receiver table (tableReceivers): RECEIVERS
- The sensors table (tableSensors): SENSORS_LOCATION

If you are using the Groovy script (Block)

```
new Merged_Sensors_Receivers().exec(connection,[
                "tableReceivers": "RECEIVERS",
                "tableSensors"  : "SENSORS_LOCATION"
])
```

Result

The RECEIVERS table has now new points, the sensor's one



Step 9 : Generate dynamic road emissions

For each time step (here 15 min), generate an emissions map for all the receivers, corresponding to the best configuration (for this time step).

To do so, use the `NMs_4_BestConfigs` Block. This Block has two inputs:

- The best configuration table (`bestConfig`)
- The road emission table (`roadEmission`)

Execution

- `bestConfig` : `BEST_CONFIGURATION_FULL`
- `roadEmission` : `LW_ROADS`

If you are using the Groovy script (Block)

```
new NMs_4_BestConfigs().exec(connection)
                "bestConfig": "BEST_CONFIGURATION_FULL",
                "roadEmission" : "LW_ROADS"
```

Result

The table `LW_ROADS_best` is created, with the following structure:

- `PK` : Unique Identifier
- `IDSOURCE` : Source id
- `PERIOD` : time
- `HZ63` : noise level for the 63 HZ frequency band
- `HZ125` : … for 125 HZ …
- `HZ250` : … for 250 HZ …
- `HZ500` : … for 500 HZ …
- `HZ1000` : … for 1000 HZ …
- `HZ2000` : … for 2000 HZ …
- `HZ4000` : … for 4000 HZ …
- `HZ8000` : … for 8000 HZ …

Step 10 : Generate the noise levels

Using the `Noise_level_from_source` Block, we can finally compute the noise level from the network sources emission (`LW_ROADS_best`) based on all the receivers.

Execution

- Source geometry table name (`tableSources`): `ROADS_GEOM`
- Source emission table name (`tableSourcesEmission`): `LW_ROADS_best`
- Buildings table name (`tableBuilding`): `BUILDINGS`
- Receivers table name (`tableReceivers`): `RECEIVERS`
- Separate receiver level by source id (`confExportSourceId`): `false`
- Maximum source-receiver distance (`confMaxSrcDist`): `250`
- Diffraction on vertical edges (`confDiffVertical`): `false`
- Diffraction on horizontal edges (`confDiffHorizontal`): `false`

If you are using the Groovy script (Block)

```
new Noise_level_from_source().exec(connection, [
                    "tableSources": "ROADS_GEOM",
                    "tableSourcesEmission" : "LW_ROADS_best",
                    "tableBuilding": "BUILDINGS",
                    "tableReceivers": "RECEIVERS",
                    "confExportSourceId": false,
                    "confMaxSrcDist": 250,
                    "confDiffVertical": false,
                    "confDiffHorizontal": false
])
```

Result

We obtain the table RECEIVERS_LEVEL.

Step 11 : Create & visualize the resulting table

Create a table, called ASSIMILATED_MAPS, containing both sound levels and configuration parameters.

To do so, execute the Create_Assimilated_Maps Block, with the 3 following parameters:

- bestConfigTable : the best configuration table name
- receiverLevel : the receivers level table name
- outputTable : the output table name

Execution

For this tutorial, you can fill with this information:

- bestConfigTable : BEST_CONFIGURATION_FULL
- receiverLevel : RECEIVERS_LEVEL
- outputTable : ASSIMILATED_MAPS

```
new Create_Assimilated_Maps().exec(connection,[
                "bestConfigTable" : "BEST_CONFIGURATION_FULL",
                "receiverLevel" : "RECEIVERS_LEVEL",
                "outputTable": "ASSIMILATED_MAPS"
])
```

Result

The resulting `ASSIMILATED_MAPS` table has the following columns:

- `TIMESTAMP` : time of the measure
- `LAEQ` : noise level (dB(A))
- `THE_GEOM` : receiver's geometry (POINT)
- `IDRECEIVER` : receiver's unique identifier

Table 6: Three first lines of the table `RECEIVERS`

TIMESTAMP	LAEQ	THE_GEOM	IDRECEIVER
1724567400	38.16436	SRID=2056;POINT Z (2496912.2844730876 1116884.1525357629 4)	1433
1724567400	39.29134	SRID=2056;POINT Z (2496912.2844730876 1116864.1525357629 4)	1197
1724567400	42.9573	SRID=2056;POINT Z (2496912.2844730876 1116844.1525357629 4)	961

Visualize the map

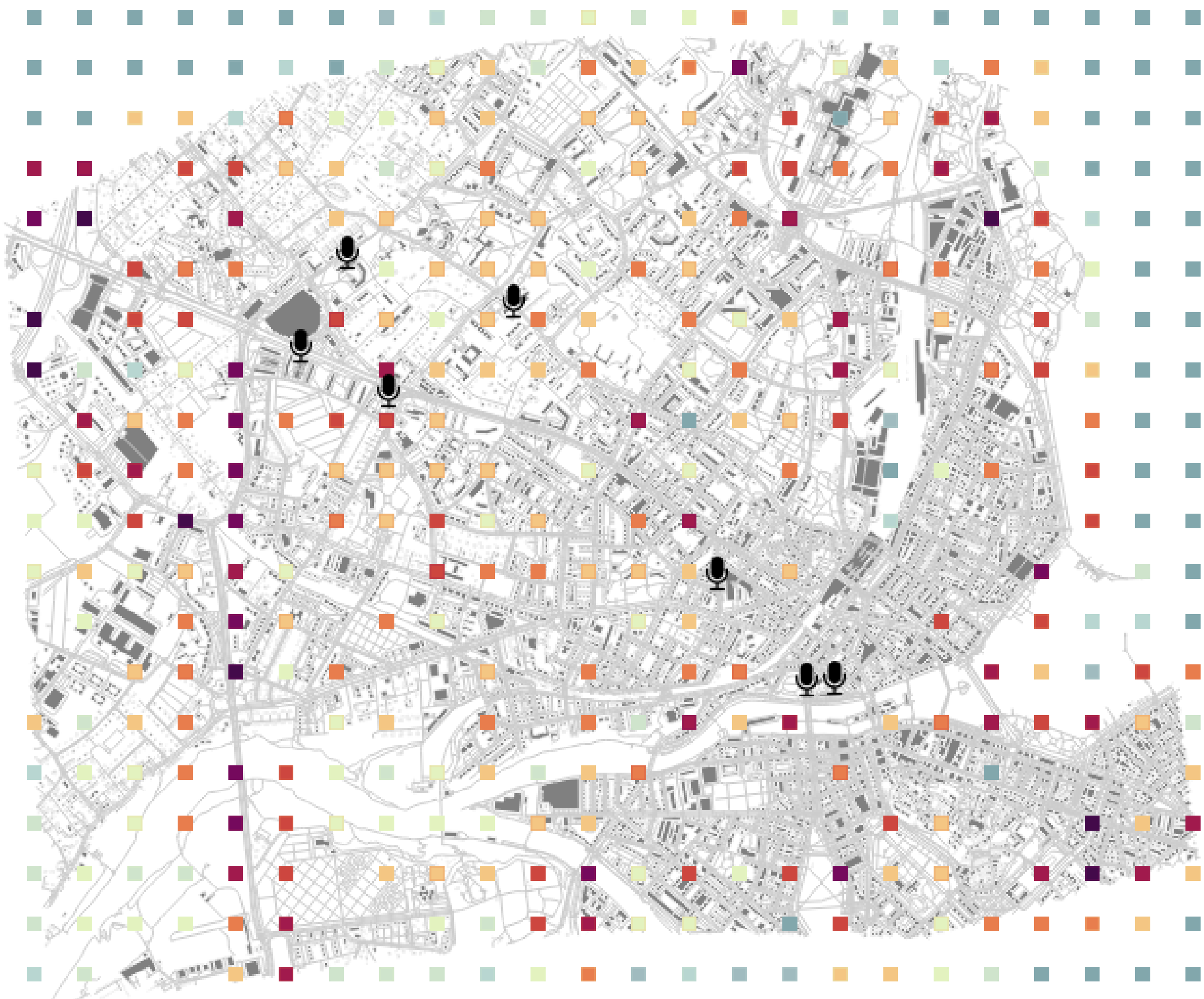
You can now export the `ASSIMILATED_MAPS` table, for example as a Shapefile, using the `Export_Table` Block and then import it into your favorite GIS app (such as QGIS) to visualize the results.

- `exportPath` : `results/ASSIMILATED_MAPS.shp`
- `tableToExport`: `ASSIMILATED_MAPS`

or

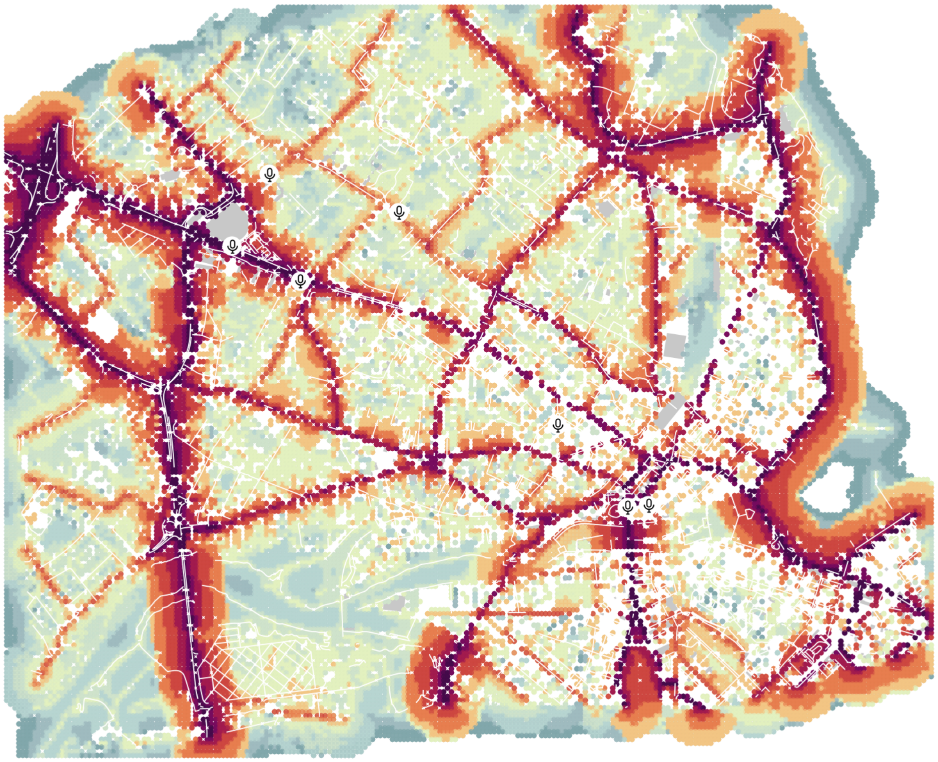
```
new Export_Table().exec(connection,
                ["exportPath": "results/ASSIMILATED_MAPS.shp",
                 "tableToExport": "ASSIMILATED_MAPS"
])
```

The resulting map may looks like this:



Legend: Here, we applied the color scheme proposed by Beate Tomio (see *Noise Map Color Scheme*) on the column LAEQ. We also added the input sensors, the buildings (medium grey) and the roads (light grey).

Remark: In this example, we have chosen to have a receiver every 200m (see *Step 7*). To achieve a denser rendering, we could have made the receiver grid denser. Below is an example, on the same area, but with a 20m grid.



5.22 NoiseModelling client line interface (CLI)

In this tutorial, we describe the different method to pilot NoiseModelling thanks to scripts. To do so, we will use a separate command-line interface, called `ScriptRunner`, in which the GUI has been removed (no more *Builder*).

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

1. with simple command lines
2. with Bash script
3. with Groovy script (Block)

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

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 `\` in the folders paths.

5.22.1 Requirements

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

5.22.2 1. Simple command line

Using the terminal of your operating system

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

```
cd /home/user/NoiseModelling/

./bin/ScriptRunner -w ./ -s scripts/Import_and_Export/Import_File.groovy --pathFile resources/ground_type.shp
```

> **Warning:** Adapt `/home/user/NoiseModelling` address with the real installation folder of NoiseModelling. Use the appropriate `./bin/ScriptRunner` or `./bin/ScriptRunner.bat` *(depending on if you are on Linux / Mac or Windows)* file, which is located in the `bin/` directory.

5.22.3 2. Bash script

Below is an example of a sequence of simple .groovy scripts (Blocks), using bash instructions and launching the 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/ScriptRunner -w ./ -s scripts/Import_and_Export/Import_File.groovy --pathFile resources/ground_type.shp
./bin/ScriptRunner -w ./ -s scripts/Import_and_Export/Import_File.groovy --pathFile resources/buildings.shp
./bin/ScriptRunner -w ./ -s scripts/Import_and_Export/Import_File.groovy --pathFile resources/receivers.shp
./bin/ScriptRunner -w ./ -s scripts/Import_and_Export/Import_File.groovy --pathFile resources/ROADS2.shp
./bin/ScriptRunner -w ./ -s scripts/Import_and_Export/Import_File.groovy --pathFile resources/dem.geojson


# Step 5: Run Calculation
./bin/ScriptRunner -w ./ -s scripts/NoiseModelling/Noise_level_from_sources.groovy --tableBuilding BUILDINGS --tableSources ROADS2 --tableReceivers RECEIVERS --tableDEM DEM --tableGroundAbs GROUND_TYPE
```

(continues on next page)

(continued from previous page)

```
# Step 6: Export (& see) the results
./bin/ScriptRunner -w ./ -s scripts/Import_and_Export/Export_Table.groovy --exportPath␣
↪RECEIVERS_LEVEL.shp --tableToExport RECEIVERS_LEVEL
```

5.22.4 3. Groovy script (Block)

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

```
package org.noise_planet.noisemodelling.scripts

import org.h2gis.api.ProgressVisitor
import org.noise_planet.noisemodelling.scripts.Import_and_Export.Import_File
import org.noise_planet.noisemodelling.scripts.NoiseModelling.Noise_level_from_source
import org.noise_planet.noisemodelling.scripts.NoiseModelling.Road_Emission_from_Traffic
import org.noise_planet.noisemodelling.webserver.utilities.Logging
import org.slf4j.Logger
import org.slf4j.LoggerFactory
import groovy.sql.Sql
import java.io.File

import java.sql.Connection

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

inputs = [ resourcesFolder : [
        description: "Path of the resource folder for input data",
        title: "Resource folder",
        default: 'resources',
        type: String.class
]]

outputs = [result: [name: 'Result output string', title: 'Result output string',␣
↪description: 'Result table name. Can be used as input for another WPS process', type:␣
↪String.class]]

def exec(Connection connection, Map input, ProgressVisitor progress) {
    Logger logger = LoggerFactory.getLogger("tutorial")
    ProgressVisitor tutorialProgress = progress.subProcess(8)
    // 8 steps in this task

    def resourceFolder = input.resourcesFolder as String
    // Upload files to database
    def groundTable = new Import_File().exec(connection, ["pathFile": new␣
↪File(resourceFolder, "ground_type.shp")], tutorialProgress)["outputTable"]

    def buildingTable = new Import_File().exec(connection, ["pathFile": new␣
↪File(resourceFolder, "buildings.shp")], tutorialProgress)["outputTable"]

```

(continues on next page)

(continued from previous page)

```
    def receiversTable = new Import_File().exec(connection, ["pathFile": new File(resourceFolder, "receivers.shp")], tutorialProgress)["outputTable"]

    def roadsTable = new Import_File().exec(connection, ["pathFile": new File(resourceFolder, "ROADS2.shp")], tutorialProgress)["outputTable"]

    def demTable = new Import_File().exec(connection, ["pathFile": new File(resourceFolder, "dem.geojson")], tutorialProgress)["outputTable"]

    def roadEmissionTable = new Road_Emission_from_Traffic().exec(connection, [tableRoads : roadsTable], tutorialProgress).result

    // print some lines of road emission
    Logging.formatSqlQueryResult(new Sql(connection), "SELECT * FROM $roadEmissionTable LIMIT 10" as String)

    // Run Calculation
    def resultTable = new Noise_level_from_source().exec(connection, ["tableBuilding": buildingTable, "tableSources": roadEmissionTable, "tableReceivers": receiversTable,
                                                                      "tableDEM"     : demTable, "tableGroundAbs": groundTable], tutorialProgress)

    // Return results
    return Logging.formatSqlQueryResult(new Sql(connection), "SELECT * FROM $resultTable.result ORDER BY IDRECEIVER, PERIOD LIMIT 10" as String, 120)
}
```

You can find this script `get_started_tutorial_complex.groovy` on the installation folder of NoiseModelling

To run it use this bash command.

```
./bin/ScriptRunner -w ./ -s get_started_tutorial_complex.groovy
```

5.23 Tutorials - FAQ

5.23.1 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.

5.23.2 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.

5.23.3 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).

5.23.4 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.

5.23.5 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)

5.24 List of functions

Below is a list of all the functions that can be run in NoiseModelling. These functions, written as `.groovy` scripts, are available in the `/scripts/` folder.

5.24.1 Acoustic Tools

Add Laeq Leq columns

Add LAeq and Leq columns

Overview

Add the columns LAeq and Leq to a table with octave band values from 63 Hz to 8000 Hz. The columns of the table should be named HZ63, HZ125,. . . , HZ8000 with an HZ prefix that can be changed.

Arguments

Mandatory inputs

`prefix` — ***Prefix of the frequency bands column***
Prefix of the columns containing the octave bands. (STRING) For example: HZ

Type: `String`

`tableName` — ***Name of the table***
Name of the table on which LAeq and Leq columns will be added.

Type: `String`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Create Isolines

Overview

Generate isolines (isophones) by linear interpolation on triangle edges (marching-triangles). One multilines map per PERIOD and per LEVEL is created. Main output table : ISOLINES_NOISE_MAP with : - PERIOD : receivers period label (VARCHAR). - LEVEL : isoline value (DOUBLE). - THE_GEOM : MULTILINESTRING/LINESTRING geometry. Additional output tables : one table per PERIOD, named L<PERIOD>_ISOLINES_NOISE_MAP, containing only the isolines of that PERIOD (same structure as above).

Arguments

Mandatory inputs

`receiversTable` — ***Receivers level table***
Name of the receivers level table.Shall contain : IDRECEIVER, PERIOD, THE_GEOM, LAEQ (or any field to contour).

Type: `String`

`trianglesTable` — ***Triangles table***
Name of the triangles table.Shall contain : PK, THE_GEOM, PK_1, PK_2, PK_3, CELL_ID.

Type: `String`

Optional inputs

`fieldName` — ***Field to contour***
Receivers numeric field to contour (e.g. LAEQ).

Type: `String`

Default: `LAEQ`

`isoClasses` — ***Iso levels (dB)***
Comma-separated levels.

Type: `String`

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

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

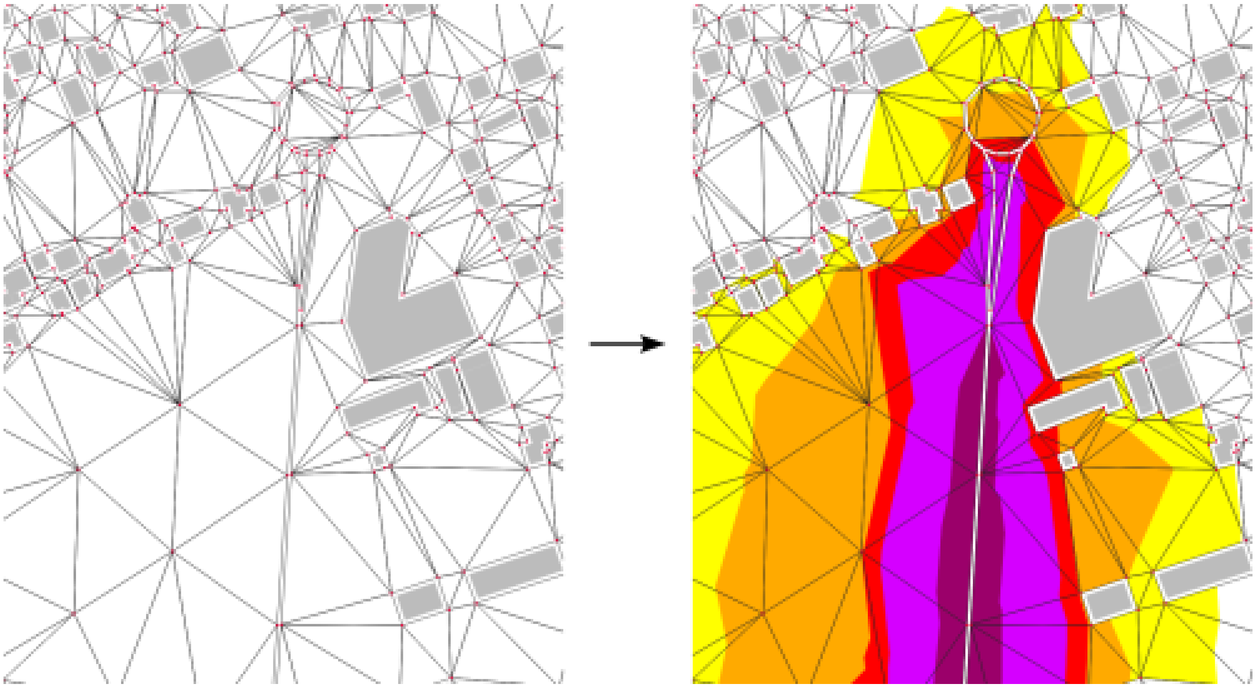
Type: `String`

Create Isosurface

Create isosurfaces from a NoiseModelling resulting table and its associated TRIANGLES table.

Overview

Create isosurfaces from a NoiseModelling resulting table and its associated TRIANGLES table. The triangle table must have been created using the "Receivers/Delaunay_Grid" WPS block. The output table is called CONTOURING_NOISE_MAP



Arguments

Mandatory inputs

`resultTable` — *Sound levels table*
: Name of the sound levels table, generated from the "Noise_level_from_source" WPS block. (STRING) Example : RECEIVERS_LEVEL

  Type: `String`

Optional inputs

`isoClass` — *Iso levels in dB*
: Separation of sound levels for isosurfaces. The first range is from -∞ to the first value (excluded). The next range is from the first value (included) to the next value (excluded). Read this documentation for more information about sound levels classes.

  Type: `String`

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

`keepTriangles` — *Keep triangles*
: Point inside areas with the same iso levels are kept so elevation variation into same iso level areas will be preserved but the output data size will be higher. Keeping triangles will reduce significantly the computation time.

  Type: `Boolean`

  Default: `false`

`resultTableField` — *Field of result table*
: Field to read in the result table to make the iso surface.

  Type: `String`

Default: `LAEQ`

`smoothCoefficient` — *Polygon smoothing coefficient*
: This coefficient (Bezier curve coefficient) will smooth the generated isosurfaces. If equal to 0, it disables the smoothing step and will keep the altitude of final polygons (3D geojson can be viewed on https://kepler.gl).Use this option with keepTriangles to keep the altitude variation into same iso level areas.

  Type: `Double`

  Default: `0`

Output

`result` — *Output table*
: Name of the output table containing the isosurfaces. The table is created in the same schema as the input result table. (STRING)

  Type: `String`

DynamicIndicators

Compute dynamic indicators

Overview

Computes dynamic percentile indicators (L10, L50, L90) for each row in the table

Arguments

Mandatory inputs

`columnName` — *Column name*
: Column name on which to perform the calculation. (STRING) For example : LEAQ

  Type: `String`

`tableName` — *Name of the table*
: Name of the table on which to perform the calculation. The table must contain multiple sound level values for a single receiver. The columns of the table should be named HZ63, HZ125,. . . , HZ8000 with an HZ prefix that can be changed. (STRING) For example : RECEIVERS_LEVEL

  Type: `String`

Optional inputs

`outputTableName` — *Name of the output table*
: Name of the output table

  Type: `String`

  Default: `tableName_DYN_IND`

Output

`result` — ***Result output string***

This type of result does not allow the blocks to be linked together.

Type: `String`

5.24.2 Data Assimilation

All Possible Configuration

All configurations

Overview

Process to generate all configurations.

Arguments

Mandatory inputs

`temperatureValues` — ***Temperature values***

List of temperature values for the road traffic emission

Type: `String`

`trafficValues` — ***Traffic values***

List of variation values in % for traffic like [0.01,1.0, 2.0,3,4]

Type: `String`

Output

`result` — ***ALL_CONFIGURATIONS***

A sql table named ALL_CONFIGURATIONS

Type: `String`

Create Assimilated Maps

Creation of the result table

Overview

Creates the ASSIMILATED_MAPS table by joining the best configuration table with the receivers noise levels.

Arguments

Mandatory inputs

`bestConfigTable` — ***The best configuration table***
: The best configuration table

  Type: `String`

`outputTable` — ***The output table name***
: The output table name

  Type: `String`

`receiverLevel` — ***The receivers Level table***
: The receivers Level table

  Type: `String`

Output

`result` — ***The result table***
: The result table

  Type: `String`

Data Simulation

Overview

Method for performing a series of operations to generate noise maps

Arguments

Optional inputs

`noiseMapLimit` — ***Number of map***
: The optional parameter between 1 and 100 corresponding to the percentage of number of maps relative to the maximal number of combinations

  Type: `Integer`

  Default: `100`

Output

`result` — *Noise map table*
: LW_ROADS and ROADS_GEOM tables output

  Type: `String`

Extract Best Configuration

Extraction of the best configurations

Overview

Extraction of the best maps, i.e. those that minimise the difference between the measured and simulated values, by calculating the minimum median values.

Arguments

Mandatory inputs

`noiseMapTable` — *Noise map table*
: Table of "noiseMapTable" containing the noise maps after simulation

  Type: `String`

`observationTable` — *Measurement table*
: Table of "observationSensor" containing the training data Set

  Type: `String`

`tempToleranceThreshold` — *Temperature tolerance threshold*
: Temperature tolerance threshold used to filter and extract the best configurations

  Type: `Double`

Output

`result` — *Best Configuration Table*
: BEST_CONFIGURATION_FULL table created

  Type: `String`

Merged Sensors Receivers

Merged Sensors and Receivers

Overview

Merges sensor locations into an existing RECEIVERS table previously created with a regular grid.

Arguments

Mandatory inputs

`tableReceivers` — ***The receiver table***
: The receiver table

  Type: `String`

`tableSensors` — ***The Sensors table***
: The Sensors table

  Type: `String`

Output

`result` — ***Merged table***
: Receiver table containing all sensors

  Type: `String`

NMs 4 BestConfigs

Dynamic Road Traffic Emission

Overview

Creation of the dynamic road using best configurations

Arguments

Mandatory inputs

`bestConfig` — ***The best configuration table***
: The best configuration table BEST_CONFIGURATION_FULL

  Type: `String`

`roadEmission` — ***The Road Emission table***
: The Road Emission table LW_ROADS

  Type: `String`

Output

results — *Dynamic Road Emission Table*
Dynamic Road Emission Table using best configuration LW_ROADS_best

Type: `String`

Prepare Sensors

Preparation of Sensor data

Overview

Extracts sensor data for a given period and creates SQL tables

Arguments

Mandatory inputs

endDate — *End Time Stamp*
The end timestamp to extract the dataset in format "%Y-%m-%d %H:%M:%S"

Type: `String`

startDate — *Start Time Stamp*
The start timestamp to extract the dataset in format "%Y-%m-%d %H:%M:%S"

Type: `String`

targetSRID — *Target projection identifier*
Target projection identifier (also called SRID) of your table. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Web Mercator projection).

The target SRID must be in metric coordinates (e.g 2056 for Geneva).

Type: `Integer`

trainingRatio — *Training data percentage*
Training data as a percentage of total data

Type: `Float`

workingFolder — *Working directory path with input files*
Folder containing .csv files "device_mapping_sf", the .osm file and the folder "devices_data"

Type: `String`

Output

`result` — *Sql tables output*
: Sql tables "SENSORS_MEASUREMENTS", "SENSORS_LOCATION", "SENSORS_MEASUREMENTS_TRAINING"

  Type: `String`

5.24.3 Database Manager

Add Primary Key

Add primary key column or constraint

Overview

Adds a Primary Key () column, or adds a Primary Key constraint to an existing column. It is necessary to add a Primary Key on one of the columns for the source and receiver tables before doing a calculation. If the table already has a Primary Key, it will remove the constraint before the operation.

Arguments

Mandatory inputs

`pkName` — *Name of the column*
: Name of the column to be added, or for which the main key constraint will be added. Primary keys must contain UNIQUE values, and cannot contain NULL values

  Type: `String`

`tableName` — *Name of the table*
: Name of the table on which a primary key will be added

  Type: `String`

Output

`result` — *Result output string*
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Clean Database

Delete all database tables

Overview

Delete all non-system tables of the database. Use with caution

Arguments

Mandatory inputs

`areYouSure` — *Are you sure?*
: Are you sure you want to delete all the tables in the database?

  Type: `Boolean`

Optional inputs

`schema` — *Schema*
: Database schema to clean. A schema is a container that organizes tables, views, and other database objects. If you are unsure, leave this set to the default schema, "public"

  Type: `String`

  Default: `public`

Output

`result` — *Result output string*
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Display Database

Display the list of tables (and their attributes).

Overview

Displays the list of tables that are in the database. Optionally it is also possible to display their attributes ("showColumns" parameter). To visualize the content of (a part of) a table, you can use "Table Visualization Data" script.

Arguments

Optional inputs

`showColumns` — ***Display columns of the tables***
: Would you also like to display the column name in the tables? Note : A small yellow key symbol () will appear if the column as a Primary Key constraint.

  Type: `Boolean`

  Default: `true`

Output

`result` — ***Result output string***
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Drop a Table

Remove a table from the database.

Overview

Remove a table from the database. Use with caution

Arguments

Mandatory inputs

`tableToDrop` — ***Name of the table to drop***
: Name of the table to drop

  Type: `String`

Output

`result` — ***Result output string***
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Execute Query

Run SQL queries and display the results

Overview

Run multiple SQL queries and display the results.

Arguments

Mandatory inputs

`sqlQueries` — ***SQL queries***
: SQL queries (e.g., CREATE TABLE mytable AS SELECT * FROM othertable; SELECT * FROM mytable;)

  Type: `String`

Optional inputs

`outputFormat` — ***Output format***
: Choose the output format for the result.

  Type: `String`

  Default: `HTML`

  Allowed values: `HTML`, `JSON`

Output

`result` — ***Result output string***
: The output of the SQL query execution, formatted according to the selected output format.

  Type: `String`

Table Visualization Data

Display first rows of a query result.

Overview

Display the content of a SQL query result. You can provide either a table name or a complete SELECT SQL query. Using "linesNumber" parameter, you can choose the number of lines to display Be careful, this treatment can be very long if the query returns many rows.

Arguments

Mandatory inputs

`tableName` — *Table name*

Table name or SQL SELECT query (e.g., mytable or SELECT * FROM mytable)

Type: `String`

Optional inputs

`linesNumber` — *Number of rows*

Number of rows you want to display. This parameter is ignored if your SQL query already contains a LIMIT clause.

Type: `Integer`

Default: `10`

Output

`result` — *Result output string*

This type of result does not allow the blocks to be linked together.

Type: `String`

Table Visualization Map

Display a table on a map.

Overview

Display a table containing a geometric column on a map Technically, it groups all the geometries of a table and returns them in WKT OGC format. Be careful, this treatment can be blocked if the table is too large.

Arguments

Mandatory inputs

`tableName` — *Name of the table*

Name of the table you want to display.

Type: `String`

Optional inputs

`inputSRID` — *Projection identifier*

Original projection identifier (also called SRID) of your table. It should be an EPSG code, a integer with 4 or 5 digits (ex: 3857 is Web Mercator projection). (INTEGER) All coordinates will be projected from the specified EPSG to WGS84 coordinates. This entry is optional because many formats already include the projection and you can also import files without geometry attributes.

Type: `Integer`

Output

`result` — *Result output geometry*

This is the output geometry in WKT OGC format

Type: `Geometry`

5.24.4 Deprecated

Noise level from traffic

Compute noise level directly from road traffic data

Overview

Computes Noise map from each period from the 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 table is RECEIVERS_LEVEL The output tables contain:

- IDRECEIVER: an identifier (INTEGER, PRIMARY KEY)
- IDSOURCE: an identifier of the source (INTEGER) if keepSource is true
- THE_GEOM : the 3D geometry of the receivers (POINT)
- PERIOD : time period ex. D, E, N, DEN (Varchar)
- Lw63, Lw125, Lw250, Lw500, Lw1000, Lw2000, Lw4000, Lw8000, Laeq, Leq: noise level at receiver (REAL)

Arguments

Mandatory inputs

`tableBuilding` — *Buildings table name*

Name of the Buildings table The table must contain:

- THE_GEOM : the 2D or 3D geometry of the building (POLYGON or MULTIPOLYGON). If the geometry is 2D then the *HEIGHT* column is mandatory
- HEIGHT : Optional, the height of the building above the ground (mandatory if geometry is 2D, ignored if geometry is 3D) (FLOAT)

Type: `String`

`tableReceivers` — ***Receivers table name***

Name of the Receivers table The table must contain:

- PK : an identifier. It shall be a primary key (INTEGER, PRIMARY KEY)
- THE_GEOM : the 3D geometry of the sources (POINT, MULTIPOINT)

This table can be generated from the WPS Blocks in the "Receivers" folder

Type: `String`

`tableRoads` — ***Roads table name***

Name of the Roads table, traffic can be provided here but are limited to DAY EVENING NIGHT periods This function recognize the following columns (* mandatory):

- PK * : an identifier. It shall be a primary key (INTEGER, PRIMARY KEY)
- LV_D TV_E TV_N : Hourly average light vehicle count (6-18h)(18-22h)(22-6h) (DOUBLE)
- MV_D MV_E MV_N : 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 (6-18h)(18-22h)(22-6h) (DOUBLE)
- HGV_D HGV_E HGV_N : Hourly average heavy duty vehicles, touring cars, buses, with three or more axles (6-18h)(18-22h)(22-6h) (DOUBLE)
- WAV_D WAV_E WAV_N : Hourly average mopeds, tricycles or quads 50 cc count (6-18h)(18-22h)(22-6h) (DOUBLE)
- WBV_D WBV_E WBV_N : Hourly average motorcycles, tricycles or quads > 50 cc count (6-18h)(18-22h)(22-6h) (DOUBLE)
- LV_SPD_D LV_SPD_E LV_SPD_N : Hourly average light vehicle speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- MV_SPD_D MV_SPD_E MV_SPD_N : Hourly average medium heavy vehicles speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- HGV_SPD_D HGV_SPD_E HGV_SPD_N : Hourly average heavy duty vehicles speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- WAV_SPD_D WAV_SPD_E WAV_SPD_N : Hourly average mopeds, tricycles or quads 50 cc speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- WBV_SPD_D WBV_SPD_E WBV_SPD_N : Hourly average motorcycles, tricycles or quads > 50 cc speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- PVMT : CNOSSOS road pavement identifier (ex: NL05)(default NL08) (VARCHAR)
- TEMP_D TEMP_E TEMP_N : Average day, evening, night temperature (default 20°C) (6-18h)(18-22h)(22-6h)(DOUBLE)
- TS_STUD : A limited period Ts (in months) over the year where a average proportion pm of light vehicles are equipped with studded tyres (0-12) (DOUBLE)
- PM_STUD : Average proportion of vehicles equipped with studded tyres during TS_STUD period (0-1) (DOUBLE)
- JUNC_DIST : Distance to junction in meters (DOUBLE)
- JUNC_TYPE : Type of junction (k=0 none, k = 1 for a crossing with traffic lights ; k = 2 for a roundabout) (INTEGER)
- SLOPE : Slope (in %) of the road section. If the field is not filled in, the LINESTRING z-values will be used to calculate the slope and the traffic direction (way field) will be force to 3 (bidirectional). (DOUBLE)

- WAY : Define 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 inverse 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 (INTEGER)

This table can be generated from the WPS Block "Import_OSM"

Type: `String`

Optional inputs

`coefficientVersion` — ***Coefficient version***
Cnossos coefficient version (1 = 2015, 2 = 2020)

Type: `Integer`

Default: `2`

`confDiffHorizontal` — ***Diffraction on horizontal edges***
Compute or not the diffraction on horizontal edges

Type: `Boolean`

Default: `false`

`confDiffVertical` — ***Diffraction on vertical edges***
Compute or not the diffraction on vertical edges. Following Directive 2015/996, enable this option for rail and industrial sources only

Type: `Boolean`

Default: `false`

`confExportSourceId` — ***Separate receiver level by source identifier***
Keep source identifier in output in order to get noise contribution of each noise source. When only the source geometry is given, the attenuation between each pair of "source-receiver" points is specified (commonly referred to as the "attenuation matrix")

Type: `Boolean`

Default: `false`

`confFavourableOccurrencesDefault` — ***Probability of occurrences***
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: `String`

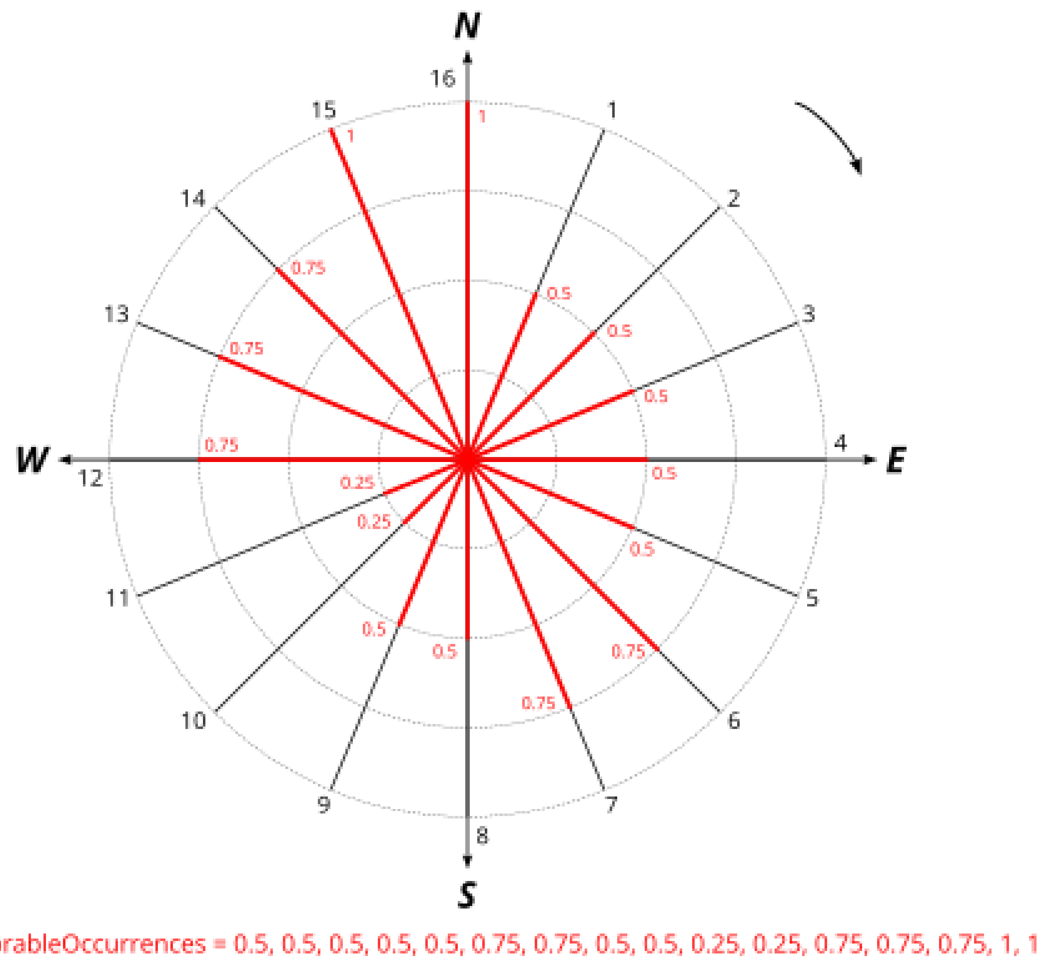
Default: `0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5`

`confHumidity` — ***Relative humidity***
Humidity for noise propagation (%) [0,100]

Type: `Double`

Default: `70`

`confMaxError` — ***Max Error (dB)***
Threshold for excluding negligible sound sources in calculations.This parameter is ignored if no emission level is specified or if you set it to 0 dB. This parameter have a great impact on computation time.

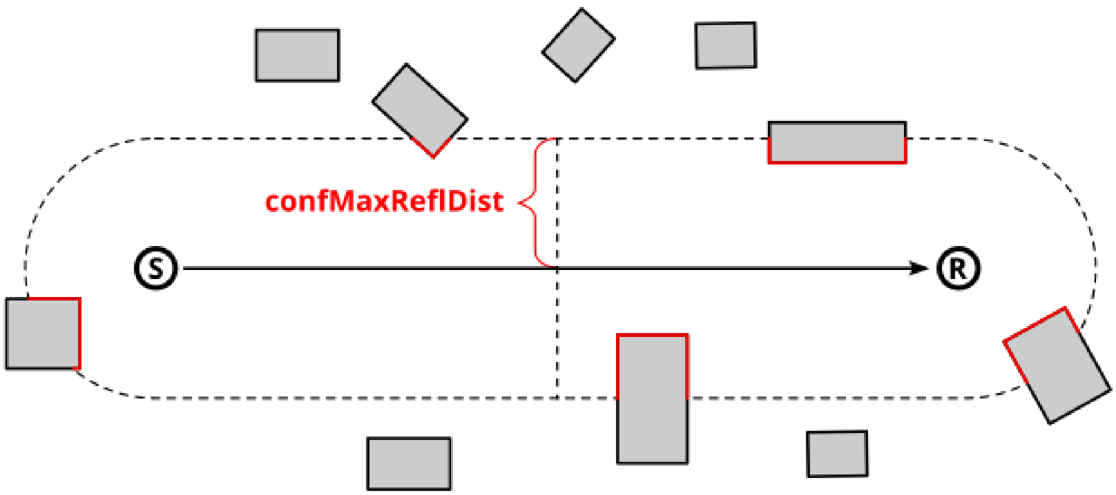

N
E
S
W
16
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
confFavorableOccurrences = 0.5, 0.5, 0.5, 0.5, 0.5, 0.75, 0.75, 0.5, 0.5, 0.25, 0.25, 0.75, 0.75, 0.75, 1, 1

Type: `Double`

Default: `0.1`

`confMaxReflDist` — *Maximum source-reflexion distance*

Maximum search distance of walls / facades from the "Source-Receiver" segment, for the calculation of specular reflections (meters).

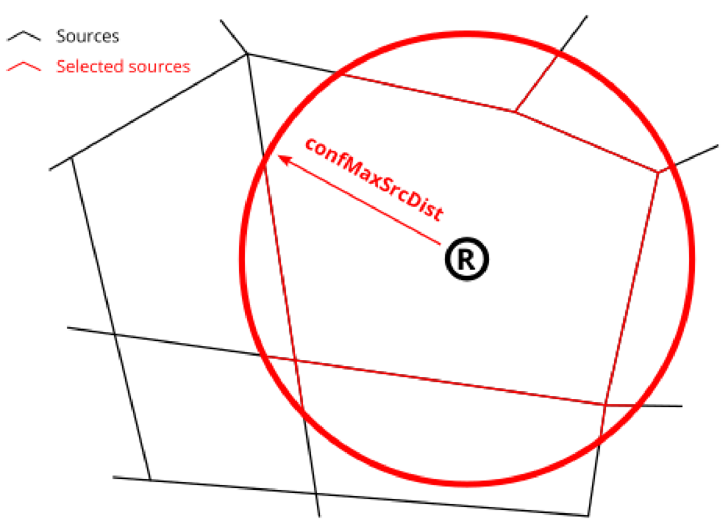


Type: `Double`

Default: `50`

`confMaxSrcDist` — *Maximum source-receiver distance*

Maximum distance between source and receiver (FLOAT, in meters).



Type: `Double`

Default: `150`

`confMinWallReflDist` — *Ignore close reflections*

Optional maximum receiver-to-wall distance (meters) below which reflection cut profiles are ignored. With regard to the population's exposure to noise, it is recommended that the contribution due to reflection off the façade wall of the building where the resident lives should be disregarded. If you have placed the receivers 0.1 m from the façades, you can set this parameter to 0.2 m. This offset is set to ensure that the contribution from the nearby wall is ignored. Use 0 to keep all reflections.

Type: `Double`

Default: `0`

`confRaysName` — *Export scene*
Save each mnt, buildings and propagation rays into the specified table (ex:RAYS) or file URL (ex: file:///Z:/dir/map.kml) You can set a table name here in order to save all the rays computed by NoiseModelling. The number of rays has been limited in this script in order to avoid memory exception.

Type: `String`

`confReflOrder` — *Order of reflexion*
Maximum number of reflections to be taken into account (INTEGER). Adding 1 order of reflexion can significantly increase the processing time

Type: `Integer`

Default: `1`

`confTemperature` — *Air temperature*
Air temperature (°C)

Type: `Double`

Default: `15`

`confThreadNumber` — *Thread number*
Number of thread to use on the computer (INTEGER).

Type: `Integer`

Default: `0`

`frequencyFieldPrepend` — *Frequency field name*
Frequency field name prepend. Ex. for 1000 Hz frequency the default column name is HZ1000

Type: `String`

Default: `HZ`

`paramWallAlpha` — *Wall absorption coefficient*
Wall absorption coefficient [0,1] (between `0` : "fully reflective" and `1` : "fully absorbent")

Type: `Double`

`tableDEM` — *DEM table name*
Name of the Digital Elevation Model (DEM) table The table must contain:

- THE_GEOM : the 3D geometry of the sources (POINT, MULTIPOINT).

This table can be generated from the WPS Block "Import_Asc_File"

Type: `String`

`tableGroundAbs` — *Ground absorption table name*
Name of the surface/ground acoustic absorption table The table must contain:

- THE_GEOM : the 2D geometry of the sources (POLYGON or MULTIPOLYGON)
- G : the acoustic absorption of a ground (FLOAT between 0 : very hard and 1 : very soft)

Type: `String`

`tablePeriodAtmosphericSettings` — *Atmospheric settings table name for each time period*

Name of the Atmospheric settings table The table must contain the following columns:

- PERIOD : time period (VARCHAR PRIMARY KEY)
- WINDROSE : probability of occurrences of favourable propagation conditions (ARRAY(16))
- TEMPERATURE : Temperature in celsius (FLOAT)
- PRESSURE : air pressure in pascal (FLOAT)
- HUMIDITY : air humidity in percentage (FLOAT)
- GDISC : choose between accept G discontinuity or not (BOOLEAN) default true
- PRIME2520 : choose to use prime values to compute eq. 2.5.20 (BOOLEAN) default false

Type: `String`

`tableRoadsTraffic` — *Roads traffic table name*

Name of the Roads traffic table per period This function recognize the following columns (* mandatory):

- IDSOURCE * : an identifier. It shall be linked to the primary key of tableRoads (INTEGER)
- PERIOD * : Time period, you will find this column on the output (VARCHAR)
- LV : Hourly average light vehicle count (DOUBLE)
- MV : 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 (DOUBLE)
- HGV : Hourly average heavy duty vehicles, touring cars, buses, with three or more axles (DOUBLE)
- WAV : Hourly average mopeds, tricycles or quads 50 cc count (DOUBLE)
- WBV : Hourly average motorcycles, tricycles or quads > 50 cc count (DOUBLE)
- LV_SPD : Hourly average light vehicle speed (DOUBLE)
- MV_SPD : Hourly average medium heavy vehicles speed (DOUBLE)
- HGV_SPD : Hourly average heavy duty vehicles speed (DOUBLE)
- WAV_SPD : Hourly average mopeds, tricycles or quads 50 cc speed (DOUBLE)
- WBV_SPD : Hourly average motorcycles, tricycles or quads > 50 cc speed (DOUBLE)
- PVMT : CNOSSOS road pavement identifier (ex: NL05)(default NL08) (VARCHAR)
- TS_STUD : A limited period Ts (in months) over the year where a average proportion pm of light vehicles are equipped with studded tyres (0-12) (DOUBLE)
- PM_STUD : Average proportion of vehicles equipped with studded tyres during TS_STUD period (0-1) (DOUBLE)
- JUNC_DIST : Distance to junction in meters (DOUBLE)
- JUNC_TYPE : Type of junction (k=0 none, k = 1 for a crossing with traffic lights ; k = 2 for a roundabout) (INTEGER)
- SLOPE : Slope (in %) of the road section. If the field is not filled in, the LINESTRING z-values will be used to calculate the slope and the traffic direction (way field) will be force to 3 (bidirectional). (DOUBLE)
- WAY : Define 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 inverse 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 (INTEGER)

Type: `String`

`tableSourceDirectivity` — ***Source directivity table name***

Name of the emission directivity table If not specified the default is train directivity of CNOSSOS-EU The table must contain the following columns:

- DIR_ID : identifier of the directivity sphere (INTEGER)
- THETA : [-90;90] Vertical angle in degree. 0° front 90° top -90° bottom (FLOAT)
- PHI : [0;360] Horizontal angle in degree. 0° front 90° right (FLOAT)
- LW63, LW125, LW250, LW500, LW1000, LW2000, LW4000, LW8000 : attenuation levels in dB for each octave or third octave (FLOAT)

Type: `String`

Output

`result` — ***Created table***

Name of the table containing the results of the computation. Can be used as input for another process.

Type: `String`

5.24.5 Dynamic

Flow 2 Noisy Vehicles

From Road traffic flows to noisy individual vehicles

Overview

Calculates individual vehicle position and noise level based on average traffic flows. A first output table is called : SOURCES_GEOM which is needed to compute the Noise Attenuation Matrixand contain : - IDSOURCE : an identifier (INTEGER, PRIMARY KEY). - ROAD_ID : id link to the road segment (INTEGER). - THE_GEOM : the 3D geometry of the sources (POINT). The output table is called : SOURCES_EMISSION and contain : - PK : an identifier (INTEGER, PRIMARY KEY). - IDSOURCE : link to the source point (INTEGER). - PERIOD : The TIMESTAMP iteration (VARCHAR).- HZ63, HZ125, HZ250, HZ500, HZ1000,HZ2000, HZ4000, HZ8000 : 8 columns giving the instantaneous emission sound level for each octave band (FLOAT).

Arguments

Mandatory inputs

`method` — ***Selected Method***

Two methods are available : - PROBA : Probabilistic representation of vehicle appearances for each time step (quicker, but sacrifices temporal coherence) Aumond, P., Jacquesson, L., & Can, A. (2018). Probabilistic modeling framework for multisource sound mapping. Applied Acoustics, 139, 34-43. . - TNP : Simplified vehicle movements (slower, but maintaining temporal coherence) De Coensel, B.; Brown, A.L.; Tomerini, D. A road traffic noise pattern simulation model that includes distributions of vehicle sound power levels. Appl. Acoust. 2016, 111, 170–178. .

Type: `String`

Allowed values: `PROBA`, `TNP`

`tableRoads` — *Roads table name*

Name of the Roads table. The table shall contain : - PK : an identifier. It shall be a primary key (INTEGER, PRIMARY KEY) - LV_D : Hourly average light and heavy vehicle count (DOUBLE) - HGV_D : Hourly average heavy vehicle count (DOUBLE) - LV_SPD_D : Hourly average light vehicle speed (DOUBLE) - HGV_SPD_D : Hourly average heavy vehicle speed (DOUBLE) - PVMT : CNOSSOS road pavement identifier (ex: NL05) (VARCHAR) This table can be generated from the WPS Block 'Import_OSM'. .

Type: `String`

Optional inputs

`duration` — *duration in sec.*

Number of seconds to compute (INTEGER).

Type: `Integer`

Default: `60`

`gridStep`

Distance between location of vehicle along the network in meters.

Type: `Integer`

Default: `10`

`timestep`

Number of iterations. Timestep in sec.

Type: `Integer`

Default: `1`

Output

`result` — *Generated table name*

Name of the generated table. Can be used as the input of other process.

Type: `String`

Ind Vehicles 2 Noisy Vehicles

Convert Individual Vehicles traffic to emission noise level and Snap them to the network point sources.

Overview

Calculating dynamic road emissions based on vehicles trajectories and snap them to the network The output table is called : SOURCES_EMISSION and contain : - IDSOURCE : an identifier (INTEGER). - PERIOD : The TIMESTAMP iteration (STRING).- HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000 : 8 columns giving the emission sound level for each octave band (FLOAT).

Arguments

Mandatory inputs

`tableFormat` — *Format of the individual Vehicles table*
: Format of the individual Vehicles table. Can be for the moment SUMO or Matsim. See in the code to understand the different format.

  Type: `String`

`tableSourceGeom` — *table of the source geometry*
: table of points source geometry, the output emission will be reattached to the index of this table according to the snap distance. Should be SOURCES_GEOM See Point_Source_From_Network to convert lines to points

  Type: `String`

`tableVehicles` — *table of the individual Vehicles*
: it should contain timestep, geometry (POINT), speed, acceleration, veh_type…

  Type: `String`

Optional inputs

`distance2snap` — *Maximum distance to snap on the network point sources*
: Maximum distance to snap on the network point sources

  Type: `Double`

`keepNoEmissionGeoms` — *Keep source geometries without emission value*
: Do not delete source geometries that does not contain any emission value. Default to true, it reduce the computation time when evaluating the attenuation

  Type: `Boolean`

  Default: `true`

Output

`result` — *Result output string*
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Noise From Attenuation Matrix

Noise Map From Attenuation Matrix

Overview

Arguments

Mandatory inputs

`attenuationTable` — ***Attenuation Matrix Table name***
: Attenuation Matrix Table name, Obtained from the Noise_level_from_source script with "confExportSourceId" enabled. Should be RECEIVERS_LEVEL The table must contain the following fields : IDRECEIVER, IDSOURCE, THE_GEOM, HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000

  Type: `String`

`lwTable` — ***LW(PERIOD)***
: LW(PERIOD) ex. SOURCES_EMISSION The table must contain the following fields : IDSOURCE, PERIOD, HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000 IDSOURCE link to primary key of attenuation table and PERIOD a varchar

  Type: `String`

`outputTable` — ***outputTable Matrix Table name***
: outputTable

  Type: `String`

Optional inputs

`lwTable_sourceId` — ***LW(PERIOD) source index field***
: LW(PERIOD) source index field. Default is IDSOURCE

  Type: `String`

Output

`result` — ***Result output string***
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Point Source From Network

Create Point Source From a network

Overview

Creates a SOURCES_GEOM point source table from a network linestring table. This table is useful to compute the attenuation matrix between sources and receivers.Create point sources from the network every "gridStep" meters. This point source will be used to compute the noise attenuation level from them to each receiver.

Arguments

Mandatory inputs

`tableNetwork` — ***Input table name***
Name of the network table.

Must contain at least:- PK: identifier with a Primary Key constraint- THE_GEOM: geometric column

Type: `String`

Optional inputs

`gridStep`
Distance between location of possible sources along the network in meters.

Type: `Integer`

`height` — ***Source height***
Height of the source in meters.

Type: `Double`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Split Sources Period

Aggregate by source index

Overview

Split a single table with duplicated geometry and source identifier into SOURCES_GEOM and SOURCES_EMISSION tables

Arguments

Mandatory inputs

`sourceIndexFieldName` — ***Source index field name***
The field name of the source index, will be translated into IDSOURCE

Type: `String`

`sourcePeriodFieldName` — ***Source period field name***
The field name of the source period (ex. T), will be translated into PERIOD

Type: `String`

`tableSourceDynamic` — *Source table name*
: Name of the Source table. The source table have for the same index multiple periods, other columns can be any supported columns of noise level from emission or noise level from traffic

  Type: `String`

Optional inputs

`sourceEmissionTableName` — *Source emission table name*
: The output table that contain for each source index, the period and other attributes of the source. Default is SOURCES_EMISSION. Can be used directly on noise_level_from_source or Noise_From_Attenuation_Matrix

  Type: `String`

`sourceGeomTableName` — *Source geometry table name*
: The output table that contain the distinct source index with the appropriate geometry. Default is SOURCES_GEOM

  Type: `String`

Output

`result` — *Result output string*
: This type of result does not allow the blocks to be linked together.

  Type: `String`

5.24.6 Experimental

Noise Map Difference

Map Difference

Overview

Computes the difference between two noise maps

Arguments

Mandatory inputs

`mainMapTable` — *Primary map table name*
: Name of the table containing the primary noise map data.

  The table must contain the following columns: PK, THE_GEOM, HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000, LAEQ, LEQ

  Type: `String`

`outTable` — *Name of created table*
: Name of the table you want to create

The table will contain the following columns: PK, THE_GEOM, HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000, LAEQ, LEQ

Type: `String`

`secondMapTable` — ***Secondary map table name***
Name of the table containing the second noise map data.

The table must contain the following columns: PK, THE_GEOM, HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000, LAEQ, LEQ

Type: `String`

Optional inputs

`invert` — ***Invert the substraction ?***
Invert the substraction?

- False (default) : Primary map - Second map
- True : Second map - Primary map

Type: `Boolean`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Road Emission From AADF

Compute Road Emission

Overview

Compute Road Emission Noise Map from Estimated Annual average daily flows (AADF) estimates. This block allows to calculate a road traffic noise emission map from the AADF estimates given in the ROADS.shp file of the tutorial. The average traffic is first converted to hourly traffic before the calculation of Lday, Levening and Lnight using distribution in Berengier et al., 2019 : "DEUFRABASE: A Simple Tool for the Evaluation of the Noise Impact of Pavements in Typical Road Geometries".

Arguments

Mandatory inputs

`sourcesTableName` — *Sources table name*

Type: `String`

Optional inputs

databaseName — *Name of the database*
Name of the database (default : first found db)

Type: `String`

Output

`result` — *Result*

Type: `String`

Road Emission From TMJA

Compute Road Emission

Overview

Compute Road Emission Noise Map from Estimated Annual average daily flows (TMJA) estimates.

Arguments

Mandatory inputs

`sourcesTableName` — *Sources table name*

Type: `String`

Optional inputs

databaseName — *Name of the database*
Name of the database (default : first found db)

Type: `String`

Output

`result` — *Result*

Type: `String`

5.24.7 Experimental Matsim

Agent Exposure

Calculate Matsim agents exposure

Overview

Loads a Matsim plans.xml file and calculate agents noise exposure, based on previously calculated timesliced noisemap at receiver positions, linked with matsim activities (facilities)

Arguments

Mandatory inputs

`dataTable` — ***Table containing the noise data***
: Table containing the noise data The table must contain the following fields : PK, IDRECEIVER, THE_GEOM, HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ000, HZ8000, PERIOD

  Type: `String`

`experiencedPlansFile` — ***Path of the Matsim output_experienced_plans file***
: Path of the Matsim output_plans file For example : /home/matsim/simulation_output/output_experienced_plans.xml.gz

  Type: `String`

`outTableName` — ***Name of created table***
: Name of the table you want to create from the file. The table will contain the following fields : PK, PERSON_ID, HOME_FACILITY, HOME_GEOM, WORK_FACILITY, WORK_GEOM, LAEQ, HOME_LAEQ, DIFF_LAEQ

  Type: `String`

`plansFile` — ***Path of the Matsim output_plans file***
: Path of the Matsim output_plans file For example : /home/matsim/simulation_output/output_plans.xml.gz

  Type: `String`

`receiversTable` — ***Table containing the receivers position***
: Table containing the receivers position The table must contain the following fields : PK, FACILITY, ORIGIN_GEOM, THE_GEOM, TYPES

  Type: `String`

`timeBinSize` — ***The size of time bins in seconds.***
: This parameter dictates the time resolution of the resulting data The time information stored will be the starting time of the time bins For exemple with a timeBinSize of 3600, the data will be analysed using the following timeBins: 0, 3600, 7200, . . . , 79200, 82800

  Type: `Integer`

Optional inputs

`SRID` — *Projection identifier*
: Original projection identifier (also called SRID) of your tables.It should be an EPSG code; an integer with 4 or 5 digits (ex: 3857 is Web Mercator projection).

  Type: `Integer`

`personsCsvFile`
: Path of the Matsim output_persons csv file For example : /home/matsim/simulation_output/output_persons.csv.gz

  Type: `String`

Output

`result` — *Result output string*
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Import Activities

Import Matsim "facilities" file

Overview

containing agents activities location.

Arguments

Mandatory inputs

`facilitiesPath` — *Path of the Matsim facilities file*
: Path of the Matsim facilities file

  Type: `String`

`outTableName` — *Name of created table*
: Name of the table you want to create The table will contain the following fields : PK, FACILITY, THE_GEOM, TYPES, BUILDING_ID

  Type: `String`

Optional inputs

`SRID` — *Projection identifier*
Original projection identifier (also called SRID) of your table.It should be an EPSG code, a integer with 4 or 5 digits (ex: 3857 is Web Mercator projection).

Type: `Integer`

Output

`result` — *Result output string*
This type of result does not allow the blocks to be linked together.

Type: `String`

Plot Exposition Distribution

Overview

Plot a graph displaying the distribution of a chosen field in a previously calculated Matsim agents noise exposition table. Will display a Graph Window on the server

Arguments

Mandatory inputs

`expositionField` — *Field containing noise exposition*
Field containing noise exposition

Type: `String`

`expositionsTableName` — *Name of the table containing the expositions*
Name of the table containing the expositions The table must contain the following fields : PK, PERSON_ID, HOME_FACILITY, HOME_GEOM, WORK_FACILITY, WORK_GEOM, LAEQ, HOME_LAEQ, DIFF_LAEQ

Type: `String`

Optional inputs

`otherExpositionField` — *Other field containing noise exposition*
Other field containing noise exposition

Type: `String`

Output

result — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Receivers From Activities Closest

Choose closest receivers for Matsim activities

Overview

Choose the closest receiver in a RECEIVERS table for every Mastim activity in an ACTIVITIES table

Arguments

Mandatory inputs

activitiesTable — ***Name of the table containing the activities***
Name of the table containing the activities The table must contain the following fields : PK, FACILITY, THE_GEOM, TYPES

Type: `String`

outTableName — ***Name of created table***
Name of the table you want to create The table will contain the following fields : PK, FACILITY, ORIGIN_GEOM, THE_GEOM, TYPES

Type: `String`

receiversTable — ***Name of the table containing the receivers***
Name of the table containing the receivers The table must contain the following fields : PK, THE_GEOM

Type: `String`

Output

result — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Receivers From Activities Random

Choose a random receivers for Matsim activities

Overview

Choose the closest building for every Mastim activity in an ACTIVITIES table, and then chose a random receiver previously generated around this building.

Arguments

Mandatory inputs

`activitiesTable` — ***Name of the table containing the activities***

Name of the table containing the activities The table must contain the following fields : PK, FACILITY, THE_GEOM, TYPES

Type: `String`

`buildingsTable` — ***Name of the table containing the buildings***

Name of the table containing the buildings The table must contain the following fields : PK, THE_GEOM

Type: `String`

`outTableName` — ***Name of created table***

Name of the table you want to create The table will contain the following fields : PK, FACILITY, ORIGIN_GEOM, THE_GEOM, TYPES, BUILD_PK

Type: `String`

`receiversTable` — ***Name of the table containing the receivers***

Name of the table containing the receivers The table must contain the following fields : PK, THE_GEOM, BUILD_PK

Type: `String`

Optional inputs

`randomSeed` — ***Random seed***

Random seed

Type: `Integer`

Default: `1234`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Traffic From Events

Import traffic data from Matsim simulation output folder

Overview

Read Matsim events output file in order to get traffic NoiseModelling input

Arguments

Mandatory inputs

`folder` — ***Path of the Matsim output folder***
Path of the Matsim output folder For example : /home/matsim/simulation_output The folder must contain at least the following files:

- output_network.xml.gz
- output_allVehicles.xml.gz
- output_events.xml.gz

Type: `String`

Optional inputs

`SRID` — ***Projection identifier***
Projection identifier (also called SRID) of the geometric data.It should be an EPSG code, a integer with 4 or 5 digits (ex: 3857 is Web Mercator projection).

Type: `Integer`

`exportTraffic` — ***Export additionnal traffic data ?***
Define if you want to output average speed and flow per vehicle category in an additional table

Type: `Boolean`

Default: `false`

`link2GeometryFile` — ***Network CSV file***
The path of the pt2matsim CSV file generated when importing OSM network. Ignored if not set. The file must contain at least two columns :

- The link ID
- The WKT geometry

Type: `String`

`outTableName` — ***Output table name***

Name of the table you want to create. A table with this name will be created plus another with a "_LW" suffix For exemple if set to "MATSIM_ROADS (default value)":

- the table MATSIM_ROADS, with the link ID and the geometry field
- the table MATSIM_ROADS_LW, with the link ID and the traffic data

Type: `String`

`populationFactor` — ***Population Factor***

Set the population factor of the MATSim simulation Must be a decimal number between 0 and 1

Type: `Double`

Default: `1.0`

`skipUnused` — ***Skip unused links ?***

Define if links with unused traffic should be omitted in the output table.

Type: `Boolean`

Default: `true`

`timeBinMax` — ***The maximum of time bins in seconds***

The maximum of time bins in seconds,

Type: `Integer`

Default: `86400`

`timeBinMin` — ***The minimum of time bins in seconds***

The minimum of time bins in seconds

Type: `Integer`

Default: `0`

`timeBinSize` — ***The size of time bins in seconds.***

This parameter dictates the time resolution of the resulting data The time information stored will be the starting time of the time bins For exemple with a timeBinSize of 3600, the data will be analysed using the following timeBins: 0, 3600, 7200, . . . , 79200, 82800

Type: `Integer`

Default: `3600`

Output

`result` — ***Result output string***

This type of result does not allow the blocks to be linked together.

Type: `String`

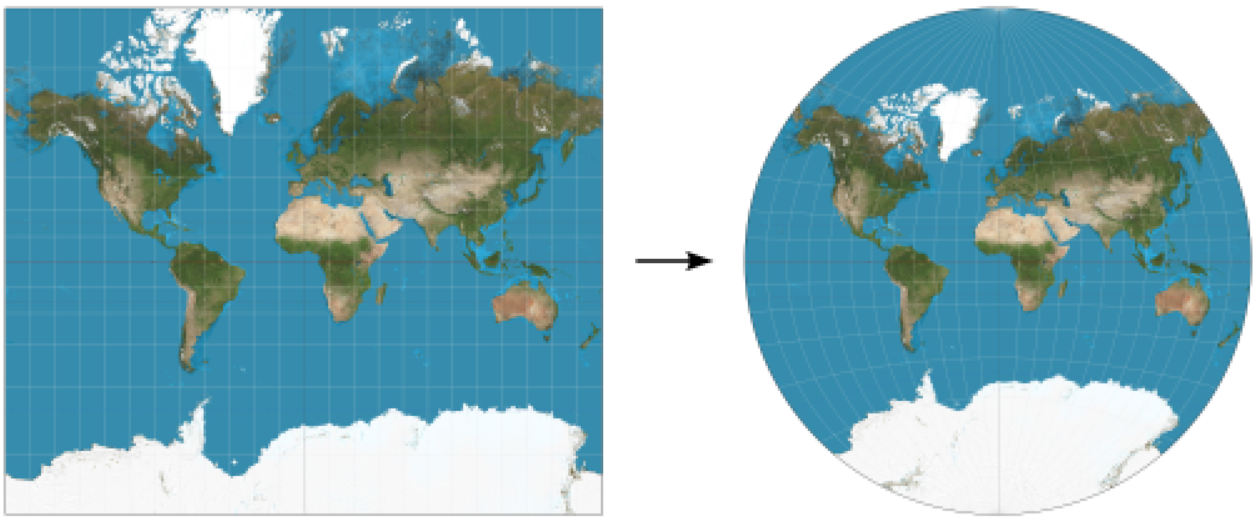
5.24.8 Geometric Tools

Change SRID

Change or set SRID

Overview

Assign a new Spatial Reference Identifier (SRID) to the specified table If the table: - has already an associated SRID: the new SRID is applied to the table and a reprojection of geometries is done, - has no associated SRID: the new SRID is applied to the table but without doing a reprojection of geometries.



Arguments

Mandatory inputs

newSRID — ***Projection identifier***
New projection identifier (also called SRID) of your table. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Pseudo-Mercator projection)

Type: `Integer`

tableName — ***Name of the table***
Name of the table you want to change the SRID (and reproject if the table has already a SRID)

Type: `String`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Clean Buildings Table

Overview

Clean the BUILDINGS table, avoiding overlapping areas and unclosed polygons. NoiseModelling propagation code does not support intersecting polygons well The input table will be erased and replaced by the cleaned one.

Arguments

Mandatory inputs

`tableName` — ***Buildings table name***
Name of the Buildings table. The table must be projected in a metric coordinate system (SRID). Use "Change_SRID" WPS Block if needed. The table shall contain: - THE_GEOM : the 2D geometry of the building (POLYGON or MULTIPOLYGON).- HEIGHT : the height of the building (FLOAT)

Type: `String`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Enrich DEM

Overview

Insert altimetric points coming from input layers into the input DEM. This script works with five input layers:

- Digital Elevation Model (DEM) to be enriched
- Orographic lines
- Hydrograpic network
- Roads
- Railways

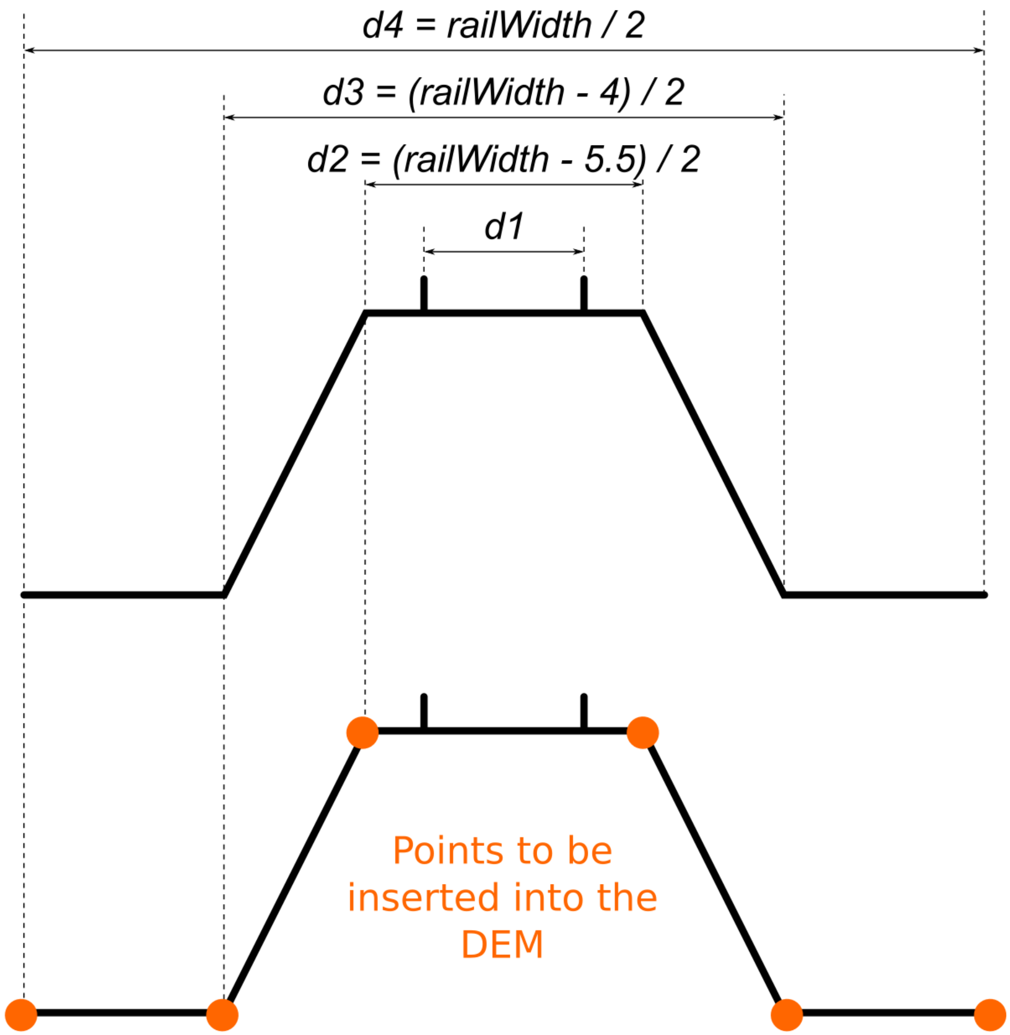
And six parameters:

- Road width (roadWidth): Name of column where the road width is stored (Mandatory)
- Roads platform height (hRoad) (Optional). Default value = 0m
- Railroads right-of-way (railWidth): Name of column where the railroad right-of-way is stored (Mandatory)

- Rail platform height (hRail) (Optional). Default value = 0.5m
- Input SRID (inputSRID): SRID of the input tables (Optional)
- Output suffixe (outputSuffixe): Suffixe applied at the end of the resuling table name (Optional). If not specified, "ENRICHED" is applied

In the schema below, orange points will be inserted into the DEM. d2, d3 and d4 are deduced from the information provided in the parameter railWidth, using the following formula:

- d2 = (railWidth - 5.5)/2
- d3 = (railWidth - 4)/2
- d4 = (railWidth)/2



Arguments

Mandatory inputs

`inputDEM` — *Input DEM table*
Name of the input DEM table to be enriched

Type: `String`

`inputHydro` — *Input hydrographic table*
Name of the input hydrographic network table

Type: `String`

`inputOro` — *Input orography table*
Name of the input orography table

Type: `String`

`inputRail` — *Input railways table*
Name of the input railways table

Type: `String`

`inputRoad` — *Input roads table*
Name of the input roads table

Type: `String`

`railWidth` — *Railways width*
Name of column where the railways width is stored

Type: `String`

`roadWidth` — *Road width*
Name of column where the road width is stored

Type: `String`

Optional inputs

`hRail` — *Railways platform height*
Railways platform height (in meters) (Optional)

Type: `double`

Default: `0.5`

`hRoad` — *Roads platform height*
Roads platform height (in meters) (Optional)

Type: `Double`

Default: `0`

`inputSRID` — *Input SRID*
SRID of the input tables. If not specified, the SRID from DEM layer is applied. If DEM has no SRID, 0 is applied

Type: `Integer`

`outputSuffixe` — ***Output suffix***
: Suffix applied at the end of the resulting table name

  Type: `String`

  Default: `ENRICHED`

Output

`result` — ***Result output string***
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Enrich DEM with lines

Overview

Insert altimetric points coming from linestring input layers into the input DEM. This script works with two input layers:

- Digital Elevation Model (DEM) to be enriched
- A linestring layer (e.g: hydrographic network, …) in which coordinates have a Z dimension

And three optional parameters:

- Input SRID (inputSRID): SRID of the input tables
- Source (source): Text indicating the source of the linestring layer. Can be useful to distinguish the points in the resulting DEM . If not specified, "LINESTRING" is applied
- Output suffix (outputsuffix): suffix applied at the end of the resuling table name. If not specified, "ENRICHED" is applied

Arguments

Mandatory inputs

`inputDEM` — ***Input DEM table***
: Name of the input DEM table to be enriched

  Type: `String`

`inputLine` — ***Input Linestring table***
: Name of the input Linestring table

  Type: `String`

Optional inputs

`inputSRID` — *Input SRID*
SRID of the input tables. If not specified, the SRID from DEM layer is applied. If DEM has no SRID, 0 is applied

Type: `Integer`

`outputsuffix` — *Output suffix*
Suffix applied at the end of the resulting table name

Type: `String`

Default: `ENRICHED`

`source` — *Source*
Text indicating the source of the linestring layer (Optional)

Type: `String`

Default: `LINESTRING`

Output

`result` — *Result output string*
This type of result does not allow the blocks to be linked together.

Type: `String`

Enrich DEM with rail

Enrich DEM with railways

Overview

Insert altimetric points coming from railways into the input DEM. This script works with two input layers:

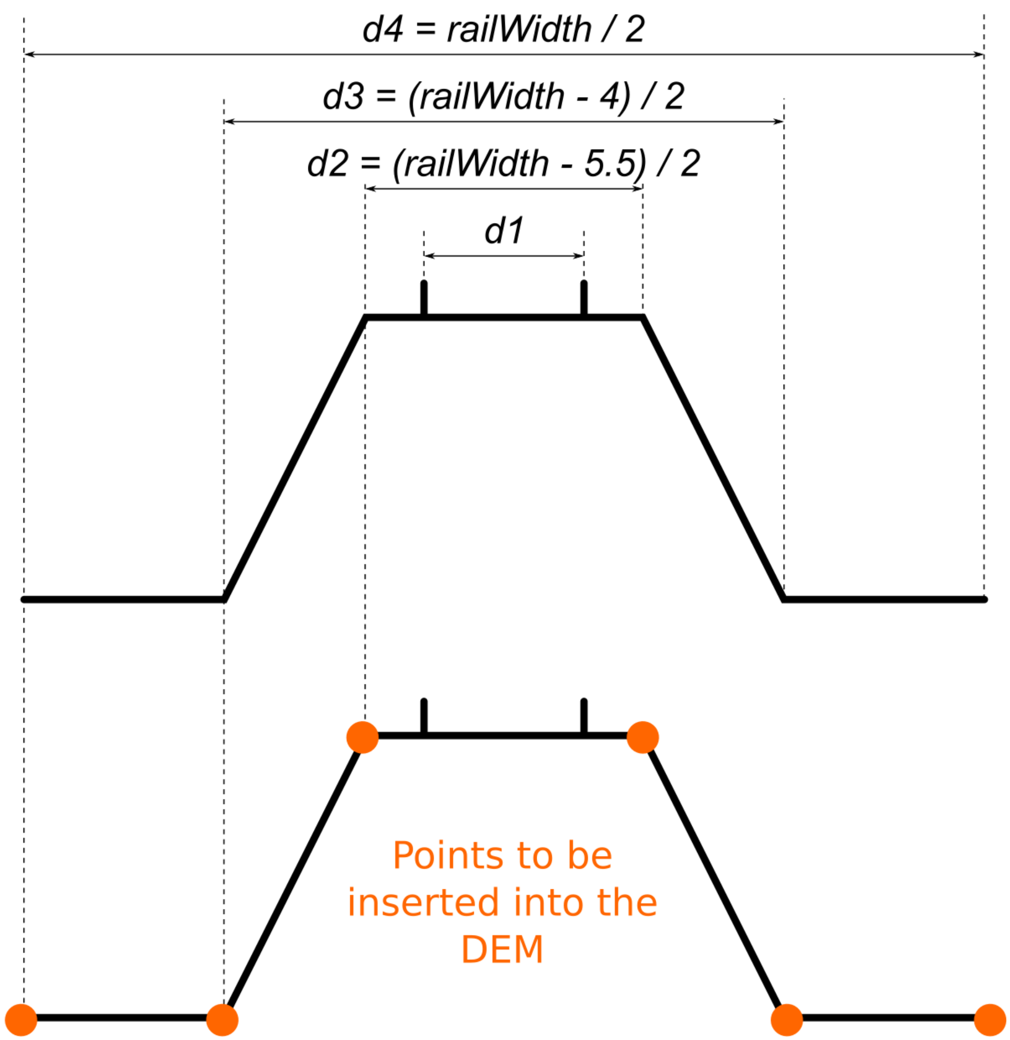
- Digital Elevation Model (DEM) to be enriched
- Railways

And four parameters:

- Railroads right-of-way (railWidth): Name of column where the railroad right-of-way is stored (Mandatory)
- Rail platform height (hRail): Railways platform height (Optional). Default value = 0.5m
- Input SRID (inputSRID): SRID of the input tables (Optional)
- Output suffix (outputsuffix): suffix applied at the end of the resuling table name (Optional). If not specified, "ENRICHED" is applied

In the schema below, orange points will be inserted into the DEM. d2, d3 and d4 are deduced from the information provided in the parameter railWidth, using the following formula:

- d2 = (railWidth - 5.5)/2
- d3 = (railWidth - 4)/2
- d4 = (railWidth)/2


d4 = railWidth / 2
d3 = (railWidth - 4) / 2
d2 = (railWidth - 5.5) / 2
d1
Points to be inserted into the DEM

Arguments

Mandatory inputs

`inputDEM` — *Input DEM table*
: Name of the input DEM table to be enriched

  Type: `String`

`inputRail` — *Input railways table*
: Name of the input railways table

  Type: `String`

`railWidth` — *Railways width*
: Name of column where the railways width is stored

  Type: `String`

Optional inputs

`hRail` — *Railways platform height*
: Railways platform height (in meters) (Optional)

  Type: `double`

  Default: `0.5`

`inputSRID` — *Input SRID*
: SRID of the input tables. If not specified, the SRID from DEM layer is applied. If DEM has no SRID, 0 is applied

  Type: `Integer`

`outputsuffix` — *Output suffix*
: Suffix applied at the end of the resulting table name If not specified, "ENRICHED" is applied

  Type: `String`

  Default: `ENRICHED`

Output

`result` — *Result output string*
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Enrich DEM with road

Enrich DEM with roads

Overview

Insert altimetric points coming from roads into the input DEM. This script works with two input layers:

- Digital Elevation Model (DEM) to be enriched
- Roads

And four parameters:

- Roads right-of-way (roadWidth): Name of column where the road right-of-way is stored (Mandatory)
- Road platform height (hRoad): Roads platform height (Optional). Default value = 0.0m
- Input SRID (inputSRID): SRID of the input tables (Optional)
- Output suffix (outputSuffix): Suffix applied at the end of the resulting table name (Optional). If not specified, "ENRICHED" is applied

Arguments

Mandatory inputs

`inputDEM` — ***Input DEM table***
Name of the input DEM table to be enriched

Type: `String`

`inputRoad` — ***Input roads table***
Name of the input roads table

Type: `String`

`roadWidth` — ***Road width***
Name of column where the road width is stored

Type: `String`

Optional inputs

`hRoad` — ***Roads platform height***
Roads platform height (in meters) (Optional)

Type: `Double`

Default: `0`

`inputSRID` — ***Input SRID***
SRID of the input tables. If not specified, the SRID from DEM layer is applied. If DEM has no SRID, 0 is applied

Type: `Integer`

outputSuffix — *Output suffix*
Suffix applied at the end of the resulting table name

Type: `String`

Default: `ENRICHED`

Output

result — *Result output string*
This type of result does not allow the blocks to be linked together.

Type: `String`

Enrich Landcover with rail

Enrich Landcover with railways

Overview

Insert rail ground surfaces into the input LANDCOVER. This script works with two input layers:

- Landcover to be enriched
- Railways

And four parameters:

- Railroads right-of-way (railWidth): Name of column where the railroad right-of-way is stored (Mandatory)
- Rail platform height (hRail): Railways platform height (Optional). Default value = 0.5m
- Input SRID (inputSRID): SRID of the input tables (Optional)
- Output suffix (outputsuffix): suffix applied at the end of the resuling table name (Optional). If not specified, "ENRICHED" is applied

In the schema below, orange points will be inserted into the DEM. d2, d3 and d4 are deduced from the information provided in the parameter railWidth, using the following formula:

- d2 = (railWidth - 5.5)/2
- d3 = (railWidth - 4)/2
- d4 = (railWidth)/2

Arguments

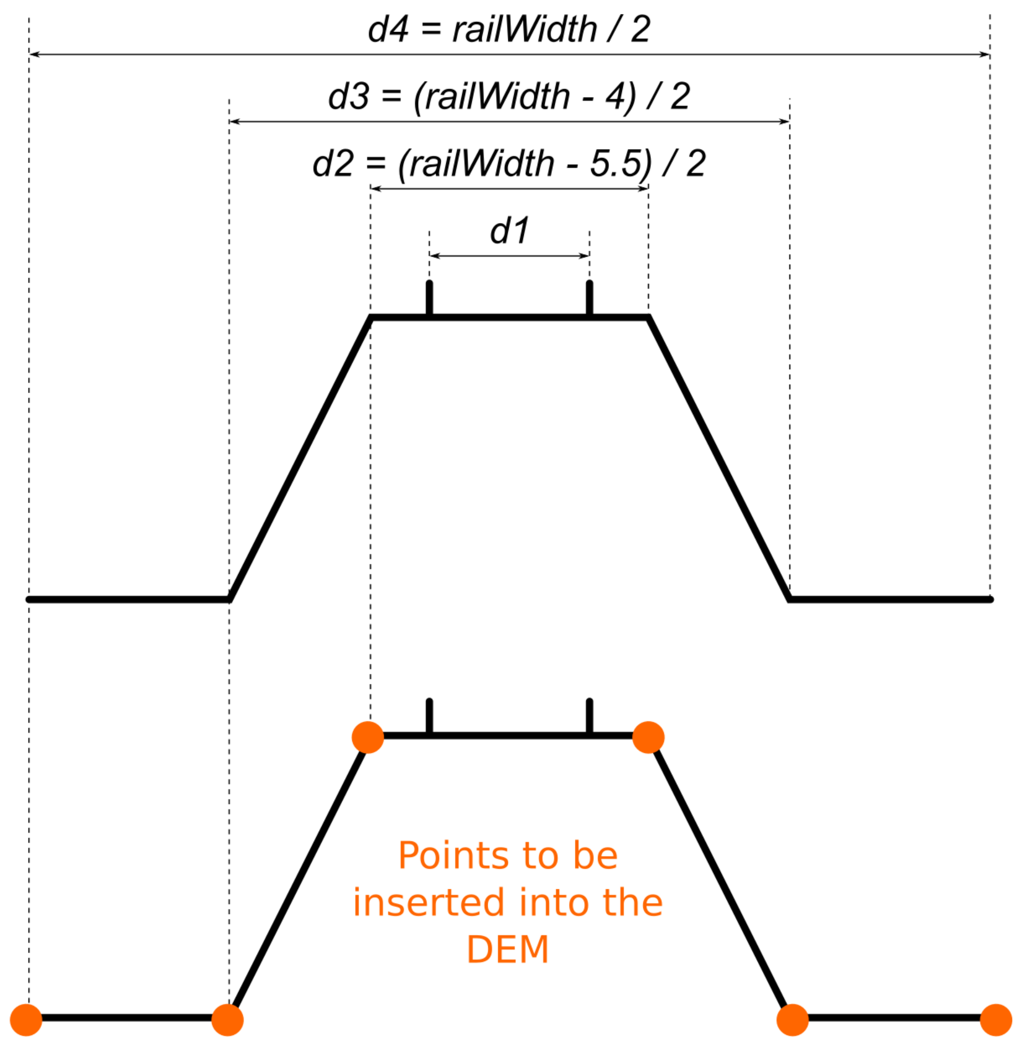
Mandatory inputs

gColumn — *G column*
Ground absorption coefficient (G) column name

Type: `String`

inputLandcover — *Input landcover table*
Name of the input landcover table

Type: `String`


d4 = railWidth / 2
d3 = (railWidth - 4) / 2
d2 = (railWidth - 5.5) / 2
d1
Points to be inserted into the DEM

inputRail — ***Input railways table***
: Name of the input railways table

  Type: `String`

railWidth — ***Railways width***
: Name of column where the railways width is stored

  Type: `String`

Optional inputs

inputSRID — ***Input SRID***
: SRID of the input tables. If not specified, the SRID from DEM layer is applied. If DEM has no SRID, 0 is applied

  Type: `Integer`

outputsuffix — ***Output suffix***
: Suffix applied at the end of the resulting table name

  Type: `String`

  Default: `ENRICHED`

Output

result — ***Result output string***
: This type of result does not allow the blocks to be linked together.

  Type: `String`

Point Source 0dB From Network

Create 0 dB Source From Roads

Overview

Creates a SOURCE table from a ROAD table. The SOURCE table can then be used in the Noise_level_from_source WPS block with the "confExportSourceId" set to true. The Noise_level_from_source output will contain a list of "source-receiver" attenuation matrix independent of the source absolute noise power levels.

Arguments

Mandatory inputs

tableRoads — ***Input table name***
: Name of the Roads table.

  Must contain at least:- PK: identifier with a Primary Key constraint- THE_GEOM: geometric column

  Type: `String`

Optional inputs

`gridStep`
Distance between location of vehicle along the network in meters.

Type: `Integer`

Default: `10`

Output

`result` — *Result output string*
This type of result does not allow the blocks to be linked together.

Type: `String`

Screen to building

Convert screens to building format

Overview

Convert the screens to the building format. A width of 10 cm will be defined. If you also give a building table, this WPS script allows you to merge the two layers together. Tables must be projected in a same metric coordinate system (SRID). Use "Change_SRID" WPS Block if needed. The output table is called : BUILDINGS_SCREENS and contain: - THE_GEOM : the 2D geometry of the created table (POLYGON or MULTIPOLYGON). - HEIGHT : the height of the created polygons (FLOAT)

Arguments

Mandatory inputs

`tableScreens` — *Screens table name*
Name of the Screens table. The table must contain: - THE_GEOM : the 2D geometry of the screens (POLYGON or MULTIPOLYGON). - HEIGHT : the height of the screens (FLOAT)

Type: `String`

Optional inputs

`tableBuilding` — *Buildings table name*
Name of the Buildings table. The table must contain: - THE_GEOM : the 2D geometry of the building (POLYGON or MULTIPOLYGON). - HEIGHT : the height of the building (FLOAT)

Type: `String`

Output

`result` — ***Result output string***

This type of result does not allow the blocks to be linked together.

Type: `String`

Set Height

Overview

Update the geometry by adding a height from the column in the input table that contains the heights or elevations or from a static value.

Arguments

Mandatory inputs

`tableName` — ***Name of the table***

Name of the table on which the height will be modified.

Type: `String`

Optional inputs

`height` — ***New height***

New height for the input table (in meters) (FLOAT)

Type: `Double`

`heightColumn`

The column name in the input table that contains the heights

Type: `String`

Output

`result` — ***Result output string***

This type of result does not allow the blocks to be linked together.

Type: `String`

Simplify Geometries

Overview

Use Douglas-Peucker algorithm to simplify geometries in the selected table. The input table geometries will be updated.

Arguments

Mandatory inputs

`tableName` — *Name of the table*
Name of the table on which geometries will be simplified

Type: `String`

Optional inputs

`distanceTolerance` — *Distance tolerance*
Sets the tolerance distance for the simplification (FLOAT)

Type: `Double`

Default: `1`

`preserveTopology` — *Preserve topology ?*
Do you want to preserve topology?

Type: `Boolean`

Default: `false`

Output

`result` — *Result output string*
This type of result does not allow the blocks to be linked together.

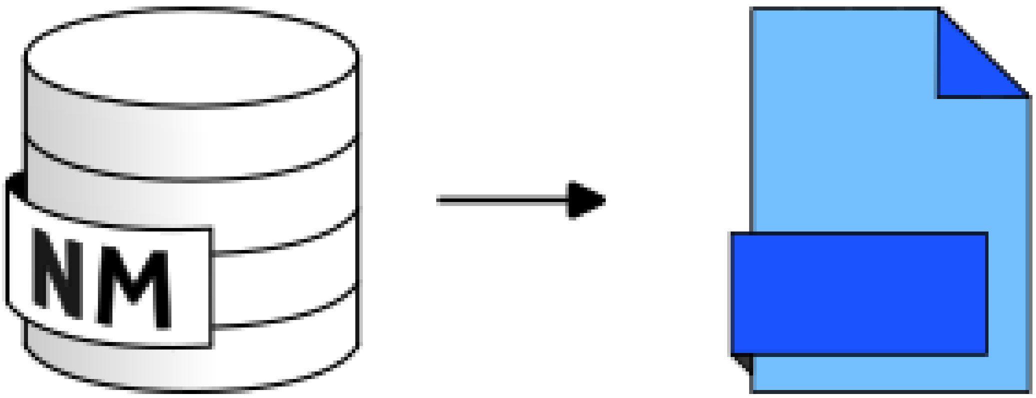
Type: `String`

5.24.9 Import and Export

Export Table

Overview

Export table from the database into a local file. Valid file extensions: csv, dbf, geojson, gpx, bz2, gz, osm, shp, tsv, fgb



Arguments

Mandatory inputs

`exportPath` — *Path of the file you want to export*
: Path of the file, including its extension. For example: c:/home/receivers.geojson

  Type: `String`

`tableToExport` — *Name of the table*
: Table Name or SQL Query Option 1: Simple table name Enter the name of an existing table, e.g.: mytable Option 2: SQL query with parenthesis Wrap your SELECT query in parenthesis to export filtered or joined data Example: (SELECT * FROM mytable WHERE field = 1)

  Type: `String`

Output

`result` — *Exported table name*
: The name of the exported table, can be used as input for another process

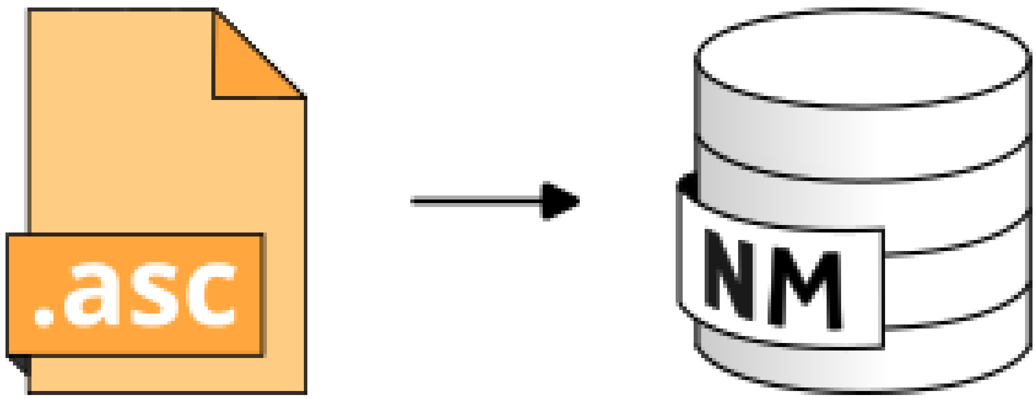
  Type: `String`

Import Asc File

Import Asc File.

Overview

Import ESRI Ascii Raster file and convert into a Digital Elevation Model (DEM) compatible with NoiseModelling (X,Y,Z). Valid file extensions : asc and asc.gz . The output table is called: DEM and contain: - THE_GEOM: the 3D point cloud of the DEM (POINT)



Arguments

Mandatory inputs

`pathFile` — *Path of the ESRI Ascii Raster file*
: Path of the ESRI Ascii Raster file you want to import, including its extension. Files can be gzip compressed. For example: c:/home/receivers.asc or c:/home/receivers.asc.gz

  Type: `String`

Optional inputs

`downscale` — *Skip pixels on each axis*
: Divide the number of rows and columns read by the following coefficient (FLOAT)

  Type: `Integer`

  Default: `1.0`

`fence` — *Fence geometry*
: Create DEM table only in the provided polygon

  Type: `Geometry`

`inputSRID` — *Projection identifier*
: Original projection identifier (also called SRID) of the .asc files. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Pseudo-Mercator projection)

  Type: `Integer`

Output

result — *Result output string*
This type of result does not allow the blocks to be linked together.

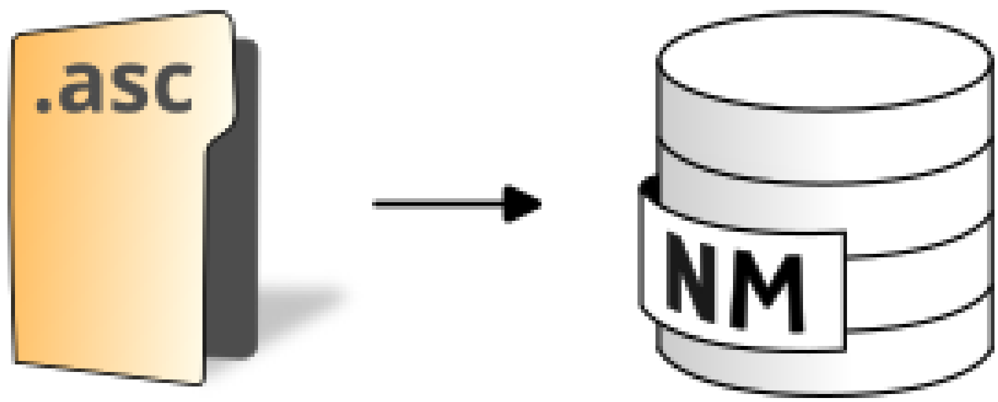
Type: `String`

Import Asc Folder

Import all .asc files from a folder

Overview

Import all files with .asc extension from a folder to the database The resulting tables will have the same name as the input files



Arguments

Mandatory inputs

pathFolder — *Path of the folder*
Path of the folder For example: c:/home/inputdata/

Type: `String`

Optional inputs

`downscale` — *Skip pixels on each axis*

Divide the number of rows and columns read by the following coefficient (FLOAT)

Type: `Integer`

Default: `1.0`

`inputSRID` — *Projection identifier*

Original projection identifier (also called SRID) of the .asc file. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Pseudo-Mercator projection)

Type: `Integer`

Output

`result` — *Result output string*

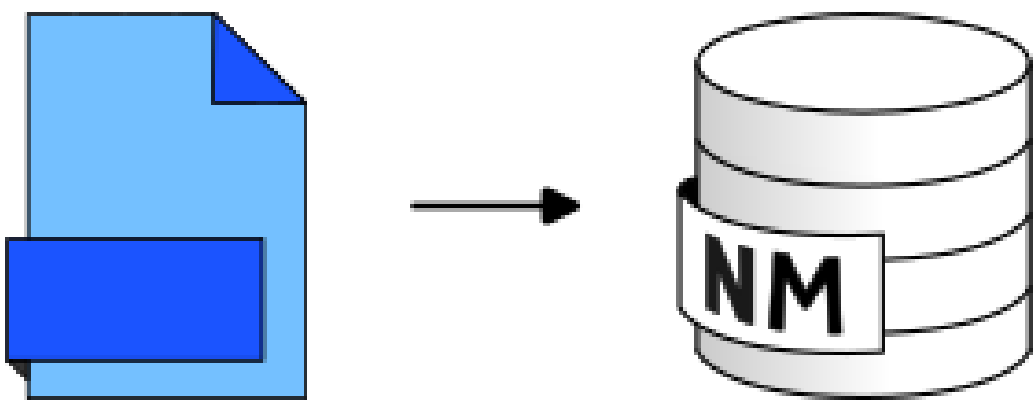
This type of result does not allow the blocks to be linked together.

Type: `String`

Import File

Overview

Import file into the database. Valid file extensions: csv, dbf, geojson, json, geojson.gz, gpx, osm.bz2, osm.gz, osm, shp, tsv



Arguments

Mandatory inputs

`pathFile` — *Path of the input File*
  Path of the file you want to import, including its extension. For example: c:/home/buildings.geojson

  Type: `String`

Optional inputs

`ifTableExists` — *Table exists operation*
  What to do if a table with the same name already exists ?

  Type: `String`

  Default: `Overwrite`

  Allowed values: `Skip import`, `Overwrite`, `Raise error`

`inputSRID` — *Projection identifier*
  Original projection identifier (also called SRID) of your table. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Pseudo-Mercator projection). This entry is optional because many formats already include the projection and you can also import files without geometry attributes. If the table is geometric and if this parameter is not filled and:- the file has a .prj file associated: the SRID is deduced from the .prj - the file has no .prj file associated: we apply the WGS84 (EPSG:4326) code

  Type: `Integer`

`tableName` — *Name of created table*
  Name of the table you want to create from the file.

  Type: `String`

Output

`outputTable` — *Name of the created table*
  Name of the created table

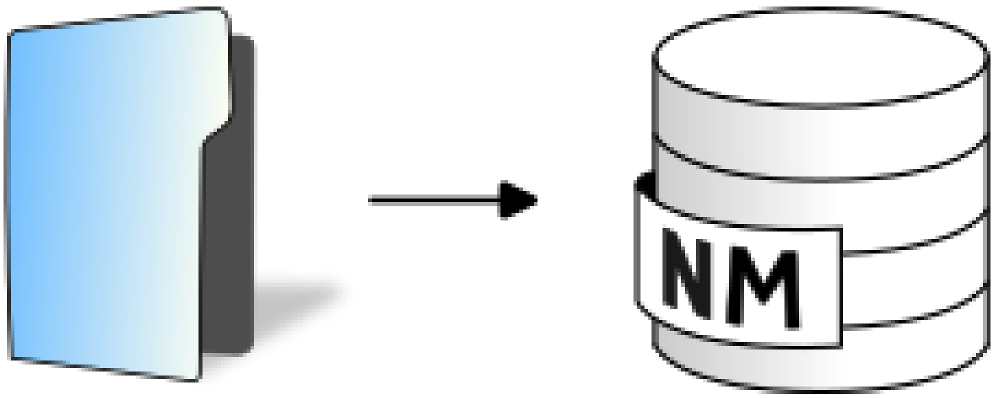
  Type: `String`

Import Folder

Import all files from a folder

Overview

Import all files with a specified extension from a folder to the database. Valid file extensions: csv, dbf, geojson, gpx, bz2, gz, osm, shp, tsv. The resulting tables will have the same name as the input files



Arguments

Mandatory inputs

`importExt` — *Extension to import*
  Extension to import. For example: shp

  Type: `String`

`pathFolder` — *Path of the folder*
  Path of the folder For example : c:/home/inputdata/

  Type: `String`

Optional inputs

`inputSRID` — *Projection identifier*
  Original projection identifier (also called SRID) of your table. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Pseudo-Mercator projection). This entry is optional because many formats already include the projection and you can also import files without geometry attributes. If the table is geometric and if this parameter is not filled and:- the file has a .prj file associated: the SRID is deduced from the .prj - the file has no .prj file associated: we apply the WGS84 (EPSG:4326) code

  Type: `Integer`

Output

`result` — *Result output string*

This type of result does not allow the blocks to be linked together.

Type: `String`

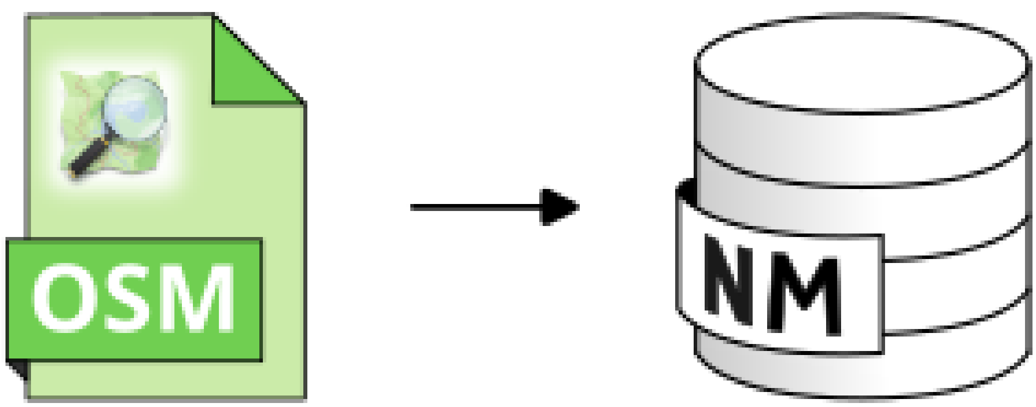
Import OSM

Import BUILDINGS, GROUND and ROADS tables from OSM

Overview

Convert .osm, .osm.gz or .osm.pbf file into NoiseModelling input tables. We recommend using OSMBBBike : https://extract.bbbike.org/ The following output tables will be created: - BUILDINGS : a table containing the buildings - GROUND : a table containing ground acoustic absorption, based on OSM landcover surfaces - ROADS : a table containing the roads. 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

The user can choose to avoid creating some of these tables by checking the dedicated boxes



Arguments

Mandatory inputs

`pathFile` — *Path of the OSM file*

Path of the OSM file, including its extension (.osm, .osm.gz or .osm.pbf). For example: c:/home/area.osm.pbf

Type: `String`

Optional inputs

`eliminateNoTrafficRoads` — ***Eliminate no traffic roads***

If checked, only roads with these "TYPE" values will remain: - bus_guideway: Dedicated lanes or tracks for buses - busway: Bus-only lanes - living_street: Residential streets with pedestrian priority - motorway: High-speed, restricted-access highways - motorway_link: Connector ramps for motorways - primary: Major roads linking large cities - primary_link: Connector ramps for primary roads - raceway: Racing tracks - residential: Roads in residential areas - road: Generic roads - secondary: Roads connecting smaller towns - secondary_link: Connector ramps for secondary roads - service: Service lanes (access to parking lots, etc.) - tertiary: Roads connecting villages and hamlets - tertiary_link: Connector ramps for tertiary roads - trunk: Important roads that are not motorways - trunk_link: Connector ramps for trunk roads - unclassified: Minor roads not fitting higher classifications - rest_area: Areas for rest along roads - traffic_calming: Traffic calming features (speed bumps, etc.) - traffic_island: Traffic islands

If not checked, all roads are processed as before.

Type: `Boolean`

Default: `false`

`ignoreBuilding` — ***Do not import Buildings***

If the box is checked → the table BUILDINGS will NOT be created.

If the box is NOT checked → the table BUILDINGS will be created and will contain: - PK : An identifier. It shall be a primary key (INTEGER, PRIMARY KEY) - THE_GEOM : The 2D geometry of the building (POLYGON or MULTIPOLYGON). - HEIGHT : The height of the building (FLOAT). If this information is not available then it is deduced from the number of floors (if available) with the addition of a small random variation from one building to another. Finally, if no information is available, a height of 5m is set by default.

Type: `Boolean`

Default: `false`

`ignoreGround` — ***Do not import Surface acoustic absorption***

If the box is checked → the table GROUND will NOT be created.

If the box is NOT checked → the table GROUND will be created and will contain: - PK : An identifier. It shall be a primary key (INTEGER, PRIMARY KEY) - ID_WAY : OSM identifier (INTEGER) - THE_GEOM : The 2D geometry of the sources (POLYGON or MULTIPOLYGON) - PRIORITY : Since NoiseModelling does not allowed overlapping geometries, if this is the case, this column is used to prioritize the geometry that will win over the other one when cutting. The order is given according to the type of land use - G : The acoustic absorption of a ground (FLOAT) (between 0 : very hard and 1 : very soft)

Type: `Boolean`

Default: `false`

`ignoreRoads` — ***Do not import Roads***

If the box is checked → the table ROADS will NOT be created.

If the box is NOT checked → the table ROADS will be created and will contain: - PK : An identifier. It shall be a primary key (INTEGER, PRIMARY KEY) - ID_WAY : OSM identifier (INTEGER) - THE_GEOM : The 2D geometry of the sources (LINESTRING or MULTILINESTRING) - LV_D : Hourly average light and heavy vehicle count (6-18h) (DOUBLE) - LV_E : Hourly average light and heavy vehicle count (18-22h) (DOUBLE) - LV_N : Hourly average light and heavy vehicle count (22-6h) (DOUBLE) - HGV_D : Hourly average heavy vehicle count (6-18h) (DOUBLE) - HGV_E : Hourly average heavy vehicle count (18-22h) (DOUBLE) - HGV_N : Hourly average heavy vehicle count (22-6h) (DOUBLE) - LV_SPD_D : Hourly average light vehicle speed (6-18h) (DOUBLE) - LV_SPD_E : Hourly average light vehicle speed (18-22h) (DOUBLE) - LV_SPD_N : Hourly average light vehicle speed (22-6h) (DOUBLE) - HGV_SPD_D : Hourly average heavy vehicle speed (6-18h) (DOUBLE) - HGV_SPD_E : Hourly average heavy vehicle speed (18-22h) (DOUBLE) - HGV_SPD_N : Hourly average heavy vehicle speed (22-6h) (DOUBLE) - PVMT : CNOSSOS road pavement identifier (ex: NL05) (VARCHAR)

These information are deduced from the roads importance in OSM..

Type: `Boolean`

Default: `false`

`removeTunnels` — ***Remove tunnels from OSM data***
If checked, remove roads from OSM data that contain OSM tag tunnel=yes.

Type: `Boolean`

Default: `false`

`targetSRID` — ***Target projection identifier***
Target projection identifier (also called SRID) of your table. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Web Mercator projection).

The target SRID must be in metric coordinates.

Type: `Integer`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Import OSM Pedestrian

Import Pedestrian tables from OSM

Overview

Convert .osm, .osm.gz or .osm.pbf file into NoiseModelling input tables.

The following output tables will be created: - BUILDINGS : a table containing the buildings The user can choose to avoid creating some of these tables by checking the dedicated boxes

Arguments

Mandatory inputs

`pathFile` — ***Path of the OSM file***
Path of the OSM file, including its extension (.osm, .osm.gz or .osm.pbf). For example: c:/home/area.osm.pbf

Type: `String`

`targetSRID` — ***Target projection identifier***
Target projection identifier (also called SRID) of your table. It should be an EPSG code, an integer with 4 or 5 digits (ex: 3857 is Web Mercator projection).

The target SRID must be in metric coordinates.

Type: `Integer`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Import Symuvia

Import Symuvia File

Overview

Import Symuvia outputs (as .xml) into the database

Arguments

Mandatory inputs

`pathFile` — ***Path of the input File***
Path of the input File (including extension .xml) For example: c:/home/mysymuviafile.xml

Type: `String`

Optional inputs

`inputSRID` — ***Symuvia output file SRID***
Symuvia output file SRID

Type: `Integer`

`tableName` — ***Name of created table***
Do not write the name of a table that contains a space

Type: `String`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Linked Table

Overview

Create a table into the database linked to an external database. The data is not stored into the database

Arguments

Mandatory inputs

`databaseUrl` — ***Database URL***
: Connection url of the database. For PostGIS jdbc:postgresql_h2://hostname:5432/databaseName. For H2 jdbc:h2:tcp://localhost/D:/data/test

  Type: `String`

`localTableName` — ***Name of created table***
: Name of the local linked table.

  Type: `String`

`lt_password` — ***User password***
: User password when connecting to the external database

  Type: `String`

`lt_username` — ***User name***
: User name when connecting to the external database

  Type: `String`

`remoteTableName` — ***External table name***
: External Table name or query. If a query is used instead of the original table name, then the table is read only. Queries must be enclosed in parenthesis: (SELECT * FROM ORDERS).

  Type: `String`

Optional inputs

`driverClass` — ***Driver name***
: Name of the class to connect to the external database.

  Type: `String`

  Default: `org.h2gis.postgis_jts.Driver`

  Allowed values: `org.h2gis.postgis_jts.Driver`, `org.h2.Driver`

`fetchSize` — ***Fetch size***
: the number of rows fetched, a hint with non-negative number of rows to fetch from the external table at once, may be ignored by the driver of external database. 0 is default and means no hint. The value is passed to java.sql.Statement.setFetchSize() method.

  Type: `Integer`

  Default: `0`

`force` — *Force*
Create the LINKED TABLE even if the remote database/table does not exist.

Type: `Boolean`

`remoteSchemaName` — *External table schema*
External Table Schema ex: public

Type: `String`

Default: `public`

Output

`result` — *Local table name*
The name of the local linked table, can be used as an input for another process

Type: `String`

5.24.10 NoiseModelling

Atmospheric Template

Generate default atmospheric settings from the PERIOD field of a noise emission table

Overview

Generate default atmospherics settings from the PERIOD field of a noise emission table. It is used to export the result table to be edited and reimported to be used into Noise_level_from_source. This table make you able to change the temperature and other settings for each time period of the simulation

Arguments

Mandatory inputs

`tableSourcesEmission` — *Sources emission table name*
Name of the Sources table (ex. SOURCES_EMISSION) The table must contain:

- IDSOURCE * : an identifier. It shall be linked to the primary key of tableRoads (INTEGER)
- PERIOD * : Time period, you will find this column on the output (VARCHAR)

Type: `String`

Optional inputs

`tablePeriodAtmosphericSettings` — ***Atmospheric settings table name output for each time period***
Name of the Atmospheric settings table The table will contain the following columns:

- PERIOD : time period (VARCHAR PRIMARY KEY)
- WINDROSE : probability of occurrences of favourable propagation conditions (ARRAY(16))
- TEMPERATURE : Temperature in celsius (FLOAT)
- PRESSURE : air pressure in pascal (FLOAT)
- HUMIDITY : air humidity in percentage (FLOAT)
- GDISC : choose between accept G discontinuity or not (BOOLEAN) default true
- PRIME2520 : choose to use prime values to compute eq. 2.5.20 (BOOLEAN) default false

Type: `String`

Default: `SOURCES_ATMOSPHERIC`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Noise level from source

Computes the propagation from the sounds sources to the receivers

Overview

Computes the propagation from the sounds sources to the receivers location using the noise emission table. Tables must be projected in a metric coordinate system (SRID). Use "Change_SRID" WPS Block if needed. The output table are called: RECEIVERS_LEVEL The output table contain:

- IDRECEIVER: receiver an identifier (INTEGER) linked to RECEIVERS table primary key
- IDSOURCE: source identifier (INTEGER) linked to SOURCES_GEOM primary key. Only if Keep source id is checked.
- PERIOD : Time period (VARCHAR) ex. L D E and DEN. Only if you provide emission power to sources or the atmospheric settings table.
- THE_GEOM : the 3D geometry of the receivers with the Z as the altitude (POINTZ)
- Hz63, Hz125, Hz250, Hz500, Hz1000,Hz2000, Hz4000, Hz8000 : 8 columns giving the sound level for each octave band (FLOAT)

Arguments

Mandatory inputs

`tableBuilding` — ***Buildings table name***

Name of the Buildings table The table must contain:

- THE_GEOM : the 2D or 3D geometry of the building (POLYGON, MULTIPOLYGON or LINESTRING). If the geometry is 2D then the *HEIGHT* column is mandatory. LineString is used to add noise barriers
- HEIGHT : Optional, the height of the building above the ground (mandatory if geometry is 2D, ignored if geometry is 3D) (FLOAT)
- G : Optional, Wall absorption value if g is [0, 1] or wall surface impedance ([N.s.m-4] static air flow resistivity of material) if G is [20, 20000] (default is 0.1 if the column G does not exists) (FLOAT)

Type: `String`

`tableReceivers` — ***Receivers table name***

Name of the Receivers table The table must contain:

- PK : an identifier. It shall be a primary key (INTEGER, PRIMARY KEY)
- THE_GEOM : the 3D geometry of the sources (POINT, MULTIPOINT)

This table can be generated from the WPS Blocks in the "Receivers" folder

Type: `String`

`tableSources` — ***Sources geometry table name***

Name of the Sources table (if only geometry is specified) The table must contain (* mandatory):

- PK * : an identifier. It shall be a primary key (INTEGER, PRIMARY KEY)
- THE_GEOM * : the 3D geometry of the sources (POINT, MULTIPOINT, LINESTRING, MULTILINESTRING). According to CNOSSOS-EU, you need to set a height of 0.05 m for a road traffic emission
- HZD63, HZD125, HZD250, HZD500, HZD1000, HZD2000, HZD4000, HZD8000 : 8 columns giving the day emission sound level for each octave band (FLOAT)
- HZE : 8 columns giving the evening emission sound level for each octave band (FLOAT)
- HZN : 8 columns giving the night emission sound level for each octave band (FLOAT)
- YAW : Source horizontal orientation in degrees. For points 0° North, 90° East. For lines 0° line direction, 90° right of the line direction. (FLOAT)
- PITCH : Source vertical orientation in degrees. 0° front, 90° top, -90° bottom. (FLOAT)
- ROLL : Source roll in degrees (FLOAT)
- DIR_ID : 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) (INTEGER)

This table can be generated from the WPS Block "Road_Emission_from_Traffic"

Type: `String`

Optional inputs

`confDiffHorizontal` — *Diffraction on horizontal edges*
Compute or not the diffraction on horizontal edges

Type: `Boolean`

Default: `false`

`confDiffVertical` — *Diffraction on vertical edges*
Compute or not the diffraction on vertical edges. Following Directive 2015/996, enable this option for rail and industrial sources only

Type: `Boolean`

Default: `false`

`confExportSourceId` — *Separate receiver level by source identifier*
Keep source identifier in output in order to get noise contribution of each noise source. When only the source geometry is given, the attenuation between each pair of "source-receiver" points is specified (commonly referred to as the "attenuation matrix")

Type: `Boolean`

Default: `false`

`confFavourableOccurrencesDefault` — *Probability of occurrences*
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: `String`

Default: `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`

`confHumidity` — *Relative humidity*
Humidity for noise propagation (%) [0,100]

Type: `Double`

Default: `70`

`confLineSourceSpacingRatio` — *Line source spacing ratio*
Dictates the density of source points created from a line sound source. A higher value means more points and finer discretization : DistanceBetweenPoints = DistanceSourceToReceiver / LineSourceSpacingRatio (this parameter)

Type: `Double`

Default: `2.0`

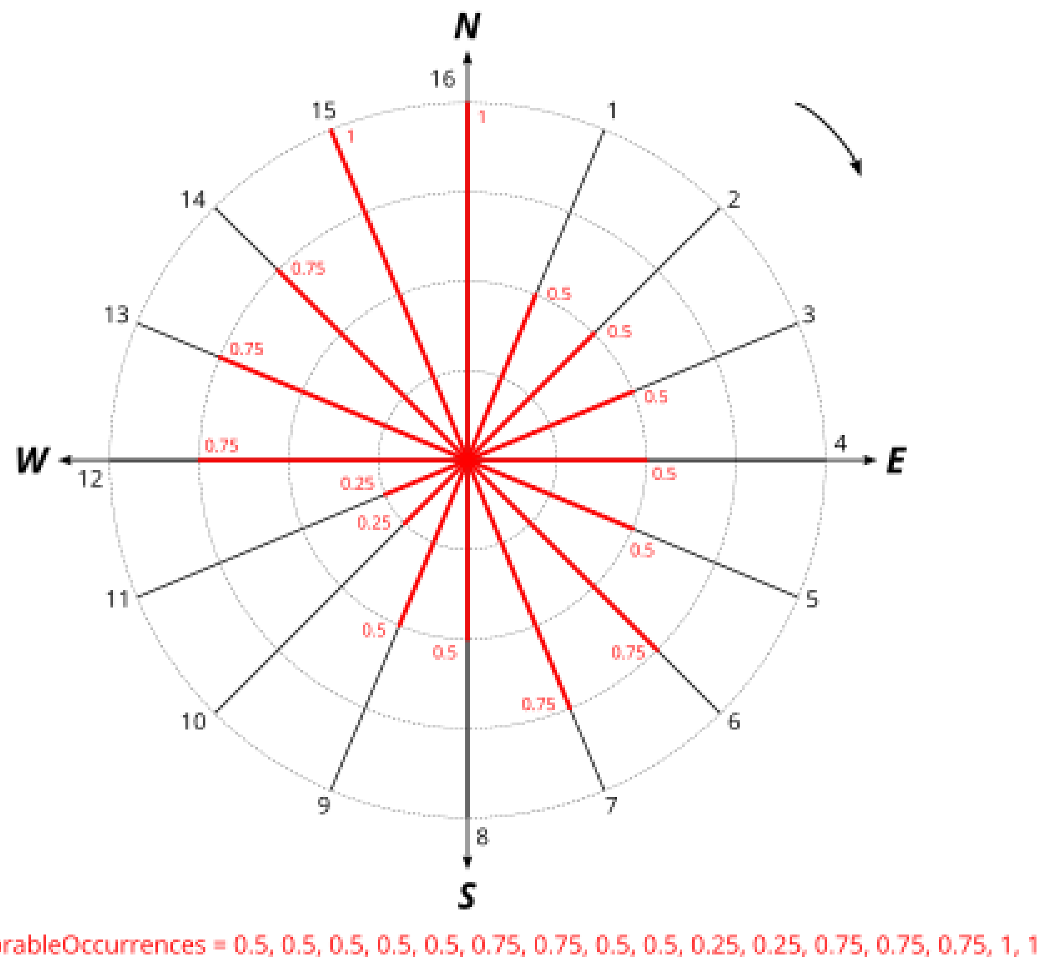
`confMaxError` — *Max Error (dB)*
Threshold for excluding negligible sound sources in calculations.This parameter is ignored if no emission level is specified or if you set it to 0 dB. This parameter have a great impact on computation time.

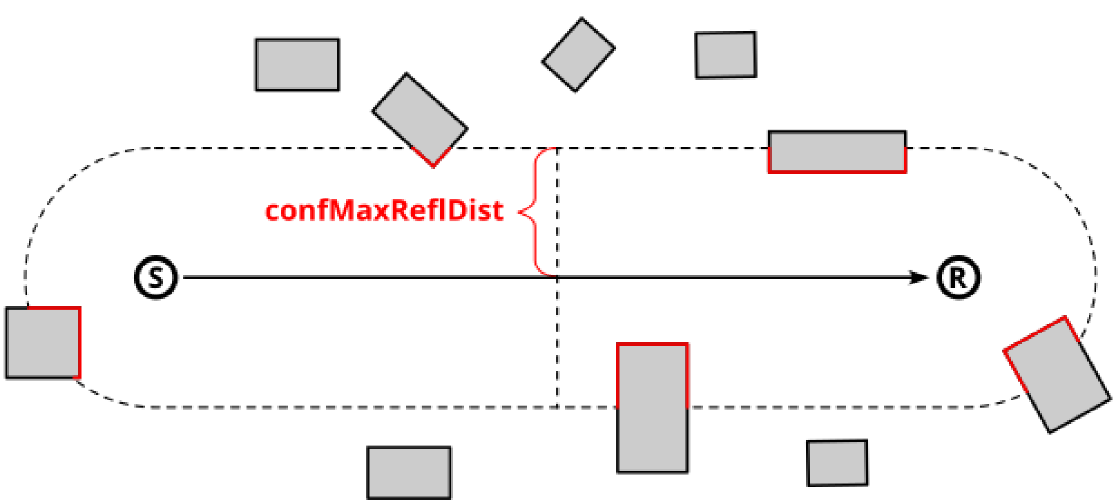
Type: `Double`

Default: `0.1`

`confMaxReflDist` — *Maximum source-reflexion distance*
Maximum search distance of walls / facades from the "Source-Receiver" segment, for the calculation of specular reflections (meters).

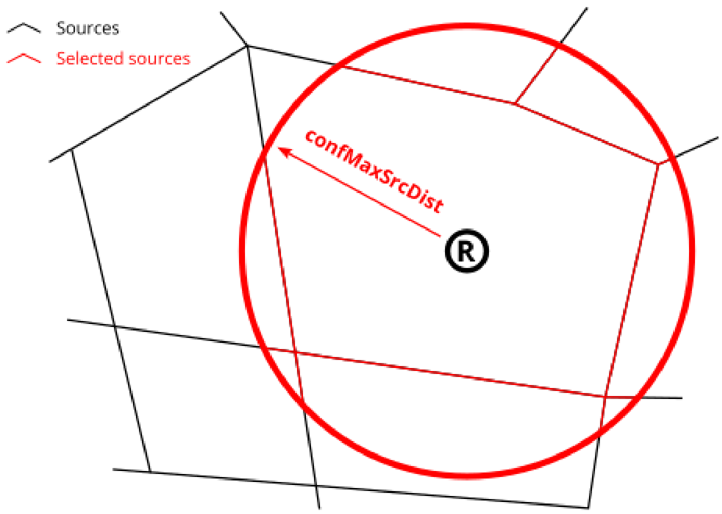
Type: `Double`


N
S
E
W
16
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
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


confMaxReflDist
S
R

Default: `50`

`confMaxSrcDist` — ***Maximum source-receiver distance***
Maximum distance between source and receiver (FLOAT, in meters).



Type: `Double`

Default: `150`

`confMinWallReflDist` — ***Ignore close reflections***
Optional maximum receiver-to-wall distance (meters) below which reflection cut profiles are ignored. With regard to the population's exposure to noise, it is recommended that the contribution due to reflection off the façade wall of the building where the resident lives should be disregarded. If you have placed the receivers 0.1 m from the façades, you can set this parameter to 0.2 m. This offset is set to ensure that the contribution from the nearby wall is ignored. Use 0 to keep all reflections.

Type: `Double`

Default: `0`

`confRaysName` — ***Export scene***
You can provide a table name to export the propagation rays with the attenuation computation details into the specified table (ex:RAYS).You can also provide a folder path URI (ex: file:///C:/Users/joe/My%20Documents/3D%20Scene/ or file:/home/user/scene3d/; you can paste the path in the browser address to convert it to an URI) to export the 3D scene (DEM, Buildings, Sources) for each sub-domains. The export format is KML and can be viewed into earth.google.com . If not provided nothing is exported

Type: `String`

`confReflOrder` — ***Order of reflexion***
Maximum number of reflections to be taken into account (INTEGER). Adding 1 order of reflexion can significantly increase the processing time.

Type: `Integer`

Default: `1`

`confTemperature` — ***Air temperature***
Air temperature (°C)

Type: `Double`

Default: `15`

`confThreadNumber` — *Thread number*

Number of thread to use on the computer (INTEGER).

Type: `Integer`

Default: `0`

`frequencyFieldPrepend` — *Frequency field name*

Frequency field name prepend. Ex. for 1000 Hz frequency the default column name is HZ1000.

Type: `String`

Default: `HZ`

`paramWallAlpha` — *Wall absorption coefficient*

Wall absorption coefficient [0,1] (between `0` : "fully reflective" and `1` : "fully absorbent")

Type: `Double`

Default: `0.1`

`tableDEM` — *DEM table name*

Name of the Digital Elevation Model (DEM) table The table must contain:

- THE_GEOM : the 3D geometry of the sources (POINT, MULTIPOINT)

This table can be generated from the WPS Block "Import_Asc_File"

Type: `String`

`tableGroundAbs` — *Ground absorption table name*

Name of the surface/ground acoustic absorption table The table must contain:

- THE_GEOM : the 2D geometry of the sources (POLYGON or MULTIPOLYGON)
- G : the acoustic absorption of a ground (FLOAT between 0 : very hard and 1 : very soft)

Type: `String`

`tablePeriodAtmosphericSettings` — *Atmospheric settings table name for each time period*

Name of the Atmospheric settings table The table must contain the following columns:

- PERIOD : time period (VARCHAR PRIMARY KEY)
- WINDROSE : probability of occurrences of favourable propagation conditions (ARRAY(16))
- TEMPERATURE : Temperature in celsius (FLOAT)
- PRESSURE : air pressure in pascal (FLOAT)
- HUMIDITY : air humidity in percentage (FLOAT)
- GDISC : choose between accept G discontinuity or not (BOOLEAN) default true
- PRIME2520 : choose to use prime values to compute eq. 2.5.20 (BOOLEAN) default false

Type: `String`

`tableSourceDirectivity` — *Source directivity table name*

Name of the emission directivity table If not specified the default is train directivity of CNOSSOS-EU The table must contain the following columns:

- DIR_ID : identifier of the directivity sphere (INTEGER)
- THETA : [-90;90] Vertical angle in degree. 0° front 90° top -90° bottom (FLOAT)

- PHI : [0;360] Horizontal angle in degree. 0° front 90° right (FLOAT)
- HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000 : attenuation levels in dB for each octave or third octave (FLOAT)

Type: `String`

`tableSourcesEmission` — *Sources emission table name*
Name of the Sources table (ex. SOURCES_EMISSION) The table must contain:

- IDSOURCE * : an identifier. It shall be linked to the primary key of tableRoads (INTEGER)
- PERIOD * : Time period, you will find this column on the output (VARCHAR)
- HZ63, HZ125, HZ250, HZ500, HZ1000, HZ2000, HZ4000, HZ8000 : Emission noise level in dB can be third-octave 50Hz to 10000Hz (FLOAT)

Type: `String`

Output

`result` — *Created table*
Name of the table containing the results of the computation. Can be used as input for another process.

Type: `String`

PlotDirectivity

Plots the directivity graph of the specified DIR_ID

Overview

Plots the directivity graph of the specified "DIR_ID"

Arguments

Mandatory inputs

`confDirId` — *Directivity Index*
Identifier of the directivity sphere from "tableSourceDirectivity" parameter or train directivity if "tableSourceDirectivity" parameter is not filled (INTEGER) In case of train, you can use these values:

- 0 = OMNIDIRECTIONAL
- 1 = ROLLING
- 2 = TRACTIONA
- 3 = TRACTIONB
- 4 = AERODYNAMICA
- 5 = AERODYNAMICB
- 6 = BRIDGE

Type: `Integer`

`confFrequency` — ***Frequency***
Frequency to plot (INTEGER). 63, 125, 250, 500, 1000, 2000, 4000, 8000 (should match with the column of tableSourceDirectivity

Type: `Integer`

Optional inputs

`confScaleMaximum` — ***Maximum scale attenuation (dB)***
Maximum scale attenuation (in dB)

Type: `Double`

Default: `0`

`confScaleMinimum` — ***Minimum scale attenuation (dB)***
Minimum scale attenuation (in dB)

Type: `Double`

Default: `-35`

`tableSourceDirectivity` — ***Source directivity table name***
Name of the emission directivity table. If not specified the default is train directivity of CNOSSOS-EU The table must contain the following columns:

- DIR_ID : identifier of the directivity sphere (INTEGER)
- THETA : [-90;90] Vertical angle in degree. 0° front 90° top -90° bottom (FLOAT)
- PHI : [0;360] Horizontal angle in degree. 0° front 90° right (FLOAT)
- LW63, LW125, LW250, LW500, LW1000, LW2000, LW4000, LW8000 : attenuation levels in dB for each octave or third octave (FLOAT).

Type: `String`

Output

`result` — ***Result output string***
Svg/Html of the directivity chart

Type: `String`

Railway Emission from Traffic

Compute railway emission noise map from vehicule, traffic table AND section table.

Overview

Compute Rail Emission Noise Map from Day, Evening and Night traffic flow rate and speed estimates (specific format, see input details). The output table is called LW_RAILWAY

Arguments

Mandatory inputs

`tableRailwayTrack` — ***RailWay Track table name***

Name of the Railway Track table. This function recognize the following columns (* mandatory):

- IDSECTION* : A section identifier (PRIMARY KEY) (INTEGER)
- NTRACK* : Number of tracks (INTEGER)
- TRACKSPD* : Maximum speed on the section (in km/h) (DOUBLE)
- TRANSFER : Track transfer function identifier, e.g. “SNCF5” or “EU7” (VARCHAR)
- ROUGHNESS : Rail roughness identifier, e.g. “SNCF1” or “EU3” (VARCHAR)
- IMPACT : Impact noise coefficient identifier, e.g. “SNCF1” or “EU1”, empty for none (VARCHAR)
- CURVATURE : Listed code describing the curvature of the section (INTEGER)
- BRIDGE : Bridge transfer function identifier, e.g. “EU3”, empty for none (VARCHAR)
- TRACKSPD : Commercial speed on the section (in km/h) (DOUBLE)
- ISTUNNEL : Indicates whether the section is a tunnel or not (0 = no / 1 = yes) (BOOLEAN)

Type: `String`

`tableRailwayTraffic` — ***Railway traffic table name***

Name of the Rail traffic table. This function recognize the following columns (* mandatory):

- IDTRAFFIC* : A traffic identifier (PRIMARY KEY) (INTEGER)
- IDSECTION* : A section identifier, refering to RAIL_SECTIONS table (INTEGER)
- TRAINTYPE* : Type of vehicle, listed in the Rail_Train_SNCF_2021 file (mainly for french SNCF) (STRING)
- TRAINSPD* : Maximum Train speed (in km/h) (DOUBLE)
- TDAY, TEVENING and TNIGHT : Hourly average train count (6-18h)(18-22h)(22-6h) (INTEGER)

Type: `String`

Optional inputs

`railwayEmissionDataFile` — ***Railway emission data file***

URL of the railway emission data file in CNOSSOS format (json). By default, the file provided with NoiseModelling is used.

Type: `String`

`trainSetDataFile` — ***Railway train set data file***

URL of the railway train set data file in CNOSSOS format (json). By default, the file provided with NoiseModelling is used.

Type: `String`

`vehicleDataFile` — ***Railway vehicle data file***
URL of the railway vehicle data file in CNOSSOS format (json). By default, the file provided with NoiseModelling is used.

Type: `String`

Output

`result` — ***Result output string***
This type of result does not allow the blocks to be linked together.

Type: `String`

Road Emission from Traffic

Compute road emission noise map from road table.

Overview

Compute Road Emission Noise Map from Day Evening Night traffic flow rate and speed estimates (specific format, see input details). The output table is called: LW_ROADS

Arguments

Mandatory inputs

`tableRoads` — ***Roads table name***
Name of the Roads table. If you provide the PERIOD field you do not need to provide the fields with the extension _D _E _N. This function recognize the following columns (* mandatory) :

- PK : If there is a primary key defined, it will be copied with the same name and set as a primary for the output table
- IDSOURCE : an identifier, if present will be copied as is. It is expected if you will use LW_ROADS as SOURCES_EMISSION in the Noise_Level_From_Source script input (INTEGER)
- PERIOD : Any text that could be time period ex. D, E, N, DEN (Varchar), if present will be copied as is
- LV : Hourly average light vehicle count (DOUBLE)
- MV : 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 (DOUBLE)
- HGV : Hourly average heavy duty vehicles, touring cars, buses, with three or more axles (DOUBLE)
- WAV : Hourly average mopeds, tricycles or quads 50 cc count (DOUBLE)
- WBV : Hourly average motorcycles, tricycles or quads > 50 cc count (DOUBLE)
- LV_SPD : Hourly average light vehicle speed (DOUBLE)
- MV_SPD : Hourly average medium heavy vehicles speed (DOUBLE)
- HGV_SPD : Hourly average heavy duty vehicles speed (DOUBLE)

- WAV_SPD : Hourly average mopeds, tricycles or quads 50 cc speed (DOUBLE)
- WBV_SPD : Hourly average motorcycles, tricycles or quads > 50 cc speed (DOUBLE)
- LV_D LV_E LV_N : Hourly average light vehicle count (6-18h)(18-22h)(22-6h) (DOUBLE)
- MV_D MV_E MV_N : 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 (6-18h)(18-22h)(22-6h) (DOUBLE)
- HGV_D HGV_E HGV_N : Hourly average heavy duty vehicles, touring cars, buses, with three or more axles (6-18h)(18-22h)(22-6h) (DOUBLE)
- WAV_D WAV_E WAV_N : Hourly average mopeds, tricycles or quads 50 cc count (6-18h)(18-22h)(22-6h) (DOUBLE)
- WBV_D WBV_E WBV_N : Hourly average motorcycles, tricycles or quads > 50 cc count (6-18h)(18-22h)(22-6h) (DOUBLE)
- LV_SPD_D LV_SPD_E LV_SPD_N : Hourly average light vehicle speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- MV_SPD_D MV_SPD_E MV_SPD_N : Hourly average medium heavy vehicles speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- HGV_SPD_D HGV_SPD_E HGV_SPD_N : Hourly average heavy duty vehicles speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- WAV_SPD_D WAV_SPD_E WAV_SPD_N : Hourly average mopeds, tricycles or quads 50 cc speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- WBV_SPD_D WBV_SPD_E WBV_SPD_N : Hourly average motorcycles, tricycles or quads > 50 cc speed (6-18h)(18-22h)(22-6h) (DOUBLE)
- TEMP : Hourly average air temperature (DOUBLE)
- TEMP_D TEMP_E TEMP_N : Hourly average air temperature (6-18h)(18-22h)(22-6h) (DOUBLE)
- PVMT : CNOSSOS road pavement identifier (ex: NL05)(default NL08) (VARCHAR)
- TS_STUD : A limited period Ts (in months) over the year where a average proportion pm of light vehicles are equipped with studded tyres (0-12) (DOUBLE)
- PM_STUD : Average proportion of vehicles equipped with studded tyres during TS_STUD period (0-1) (DOUBLE)
- JUNC_DIST : Distance to junction in meters (DOUBLE)
- JUNC_TYPE : Type of junction (k=0 none, k = 1 for a crossing with traffic lights ; k = 2 for a roundabout) (INTEGER)
- SLOPE : Slope (in %) of the road section. If the field is not filled in, the LINESTRING z-values will be used to calculate the slope and the traffic direction (way field) will be force to 3 (bidirectional). (DOUBLE)
- WAY : Define 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 inverse 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 (INTEGER)

This table can be generated from the WPS Block 'Import_OSM'. .

Type: `String`

Optional inputs

`coefficientVersion` — ***Coefficient version***

Cnossos coefficient version (1 = 2015, 2 = 2020)

Type: `Double`

Default: `2`

`outputTable` — ***Output table name***

Name of the output table. If the table already exists, it will be dropped and replaced by the new one.

Type: `String`

Default: `LW_ROADS`

Output

`result` — ***Result output string***

This type of result does not allow the blocks to be linked together.

Type: `String`

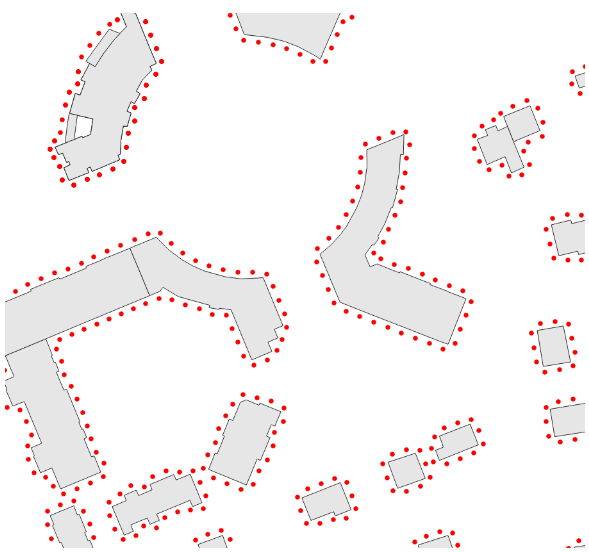
5.24.11 Receivers

Building Grid

Buildings Grid

Overview

Generates receivers, 2m around the building facades, at a given height. The output table is called RECEIVERS and contain a field build_pk corresponding to the primary key of the buildings table



Arguments

Mandatory inputs

`tableBuilding` — ***Buildings table name***
Name of the Buildings table. The table must contain:

- THE_GEOM : the 2D geometry of the building (POLYGON or MULTIPOLYGON)
- HEIGHT : the height of the building (in meter) (FLOAT)
- POP : (optional field) building population to add in the receiver attribute (FLOAT)

Type: `String`

Optional inputs

`delta` — ***Distance between receivers***
Distance between receivers (in the Cartesian plane - in meter) (FLOAT)

Type: `Double`

Default: `10`

`distance` — ***Distance from wall***
Distance between the receivers and the wall, in metres (FLOAT)

Type: `Double`

Default: `2`

`fence` — ***Extent filter***
Create receivers only in the provided polygon (fence)

Type: `Geometry`

`fenceTableName` — ***Filter using table bounding box***
Filter receivers, using the bounding box of the given table name:

- Extract the bounding box of the specified table,
- then create only receivers on the table bounding box.

The given table must contain:

- THE_GEOM : any geometry type.

Type: `String`

`height` — ***Height***
Height of receivers (in meter) (FLOAT)

Type: `Double`

Default: `4`

`sourcesTableName` — ***Sources table name***
Keep only receivers that are at least 1 meter from the provided source geometries.The source geometries table must contain:

- THE_GEOM : any geometry type.

Type: `String`

Output

`result` — *Created table*

Name of the table containing the results of the computation. Can be used as input for another process.

Type: `String`

Building Grid3D

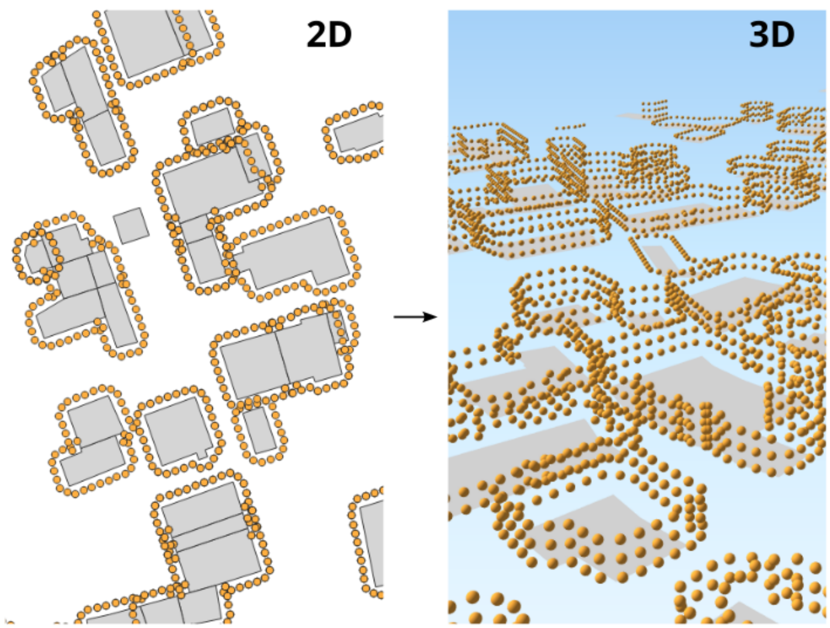
Buildings Grid

Overview

Generates 3D receivers around the buildings and at different levels. Main parameters:

- "Height between levels": coupled with the building height, allows to determine the number of levels,
- "Distance from wall": set the distance between the receivers and the building facades,
- "Distance between receivers": set the number of receivers around the buildings.

The output table is called RECEIVERS



Arguments

Mandatory inputs

`tableBuilding` — ***Buildings table name***
Name of the Buildings table. The table must contain:

- THE_GEOM : the 2D geometry of the building (POLYGON or MULTIPOLYGON)
- HEIGHT : the height of the building (in meter) (FLOAT)
- POP : building population to add in the receiver attribute (FLOAT) (Optional)

Type: `String`

Optional inputs

`delta` — ***Distance between receivers***
Distance between receivers (in the Cartesian plane - in meters) (FLOAT)

Type: `Double`

Default: `10`

`distance` — ***Distance from wall***
Distance between the receivers and the wall, in metres (FLOAT)

Type: `Double`

Default: `2`

`fence` — ***Extent filter***
Create receivers only in the provided polygon (fence)

Type: `Geometry`

`fenceTableName` — ***Filter using table bounding box***
Filter receivers, using the bounding box of the given table name:

- Extract the bounding box of the specified table,
- then create only receivers on the table bounding box.

The given table must contain:

- THE_GEOM : any geometry type.

Type: `String`

`heightLevels` — ***Height between levels***
Height between each level of receivers, in meters (FLOAT)

Type: `Double`

Default: `2.5`

`sourcesTableName` — ***Sources table name***
Keep only receivers that are at least 1 meter from the provided source geometries.The source geometries table must contain:

- THE_GEOM : any geometry type

Type: `String`

Output

`result` — *Created table*
: Name of the table containing the results of the computation. Can be used as input for another process.

  Type: `String`

Delaunay Grid

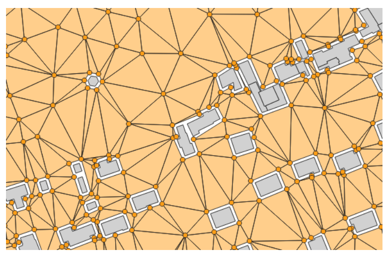
Overview

Computes a Delaunay grid of receivers. The grid will be based on:

- the BUILDINGS table extent (option by default)
- OR a single Geometry "fence" (Extent filter).

Two tables are returned:

- RECEIVERS
- TRIANGLES



Arguments

Mandatory inputs

`sourcesTableName` — *Sources table name*
: Name of the Road table. Receivers will not be created on the specified road width

  Type: `String`

`tableBuilding` — *Buildings table name*
: Name of the Buildings table. The table must contain:

  - THE_GEOM : the 2D geometry of the building (POLYGON or MULTIPOLYGON)

  Type: `String`

Optional inputs

`buildingBuffer` — ***Minimum distance to buildings (m)***
Do not add receivers closer than this distance to buildings (in meters)

Type: `Double`

Default: `2`

`exportTrianglesGeometries` — ***In the triangles table, export triangles geometries***
If enabled, the TRIANGLES table will contain the geometry of each triangle

Type: `Boolean`

Default: `false`

`fence` — ***Extent filter***
Create receivers only in the provided polygon (fence)

Type: `Geometry`

`fenceNegativeBuffer` — ***Negative buffer***
Reduce the fence(parameter, or sound sources and buildings extent) used to generate receivers positions. You should set here the maximum propagation distance (in meters) (FLOAT)

Type: `Double`

Default: `0`

`fenceTableName` — ***Fence table name***
Use the extent of a geometry table (e.g., from a shapefile) to limit receiver area

Type: `String`

`height` — ***Height***
Receiver height relative to the ground (in meters) (FLOAT)

Type: `Double`

Default: `4`

`isoSurfaceInBuildings` — ***Create IsoSurfaces over buildings***
If enabled, isosurfaces will be visible at the location of buildings

Type: `Boolean`

Default: `false`

`maxArea` — ***Maximum Area***
Set Maximum Area (in m2) (FLOAT). No triangles larger than provided area will be created.Smaller area will create more receivers

Type: `Double`

Default: `2500`

`maxCellDist` — ***Maximum cell size***
Maximum distance used to split the domain into sub-domains (in meters) (FLOAT). 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

Type: `Double`

Default: `600`

`outputTableName` — ***Name of output table***
Name of the output table. Do not write the name of a table that contains a space

Type: `String`

Default: `RECEIVERS`

`outputTableNameTriangles` — ***Name of triangles output table***
Name of the triangles output table.

Type: `String`

Default: `TRIANGLES`

`roadWidth` — ***Road width***
Set Road Width (in meters) (FLOAT). No receivers closer than road width distance will be created. You can set 0 m if you don't want to insert roads in the output but still want to skip cells without sources using the 'Skip cell no sources minimal distance' parameter

Type: `Double`

Default: `2`

`skipCellNoSourcesMinimalDistance` — ***Skip cell no sources minimal distance***
If provided, a sub-domain will not be computed if no sources geometries are near x meters from the sub-domain area

Type: `Double`

Output

`result` — ***Created table***
Name of the table containing the results of the computation. Can be used as input for another process.

Type: `String`

Random Grid

Overview

Computes a random grid of receivers. The output table is called RECEIVERS

Arguments

Mandatory inputs

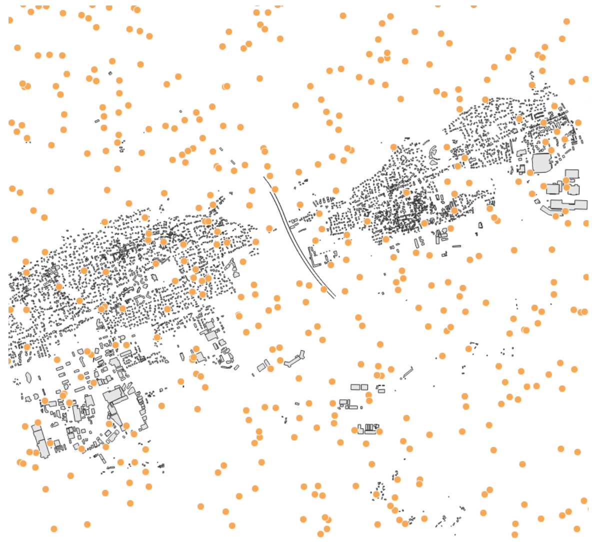
`buildingTableName` — ***Buildings table name***
Name of the Buildings table The table must contain:

- THE_GEOM: the 2D geometry of the building (POLYGON or MULTIPOLYGON)
- HEIGHT: the height of the building (FLOAT)

Type: `String`

`sourcesTableName` — ***Sources table name***
Keep only receivers at least at 1 meters of provided sources geometries The table must contain :

- THE_GEOM: any geometry type.



Type: `String`

Optional inputs

`fence` — ***Extent filter***
Create receivers only in the provided polygon. Must be in the WGS84 (EPSG:4326) projection system

Type: `Geometry`

`fenceTableName` — ***Filter using table bounding box***
Extract the bounding box of the specified table then create only receivers on the table bounding box. The table must contain :

- THE_GEOM: any geometry type.

Type: `String`

`height` — ***Height***
Height of receivers (in meters) (FLOAT)

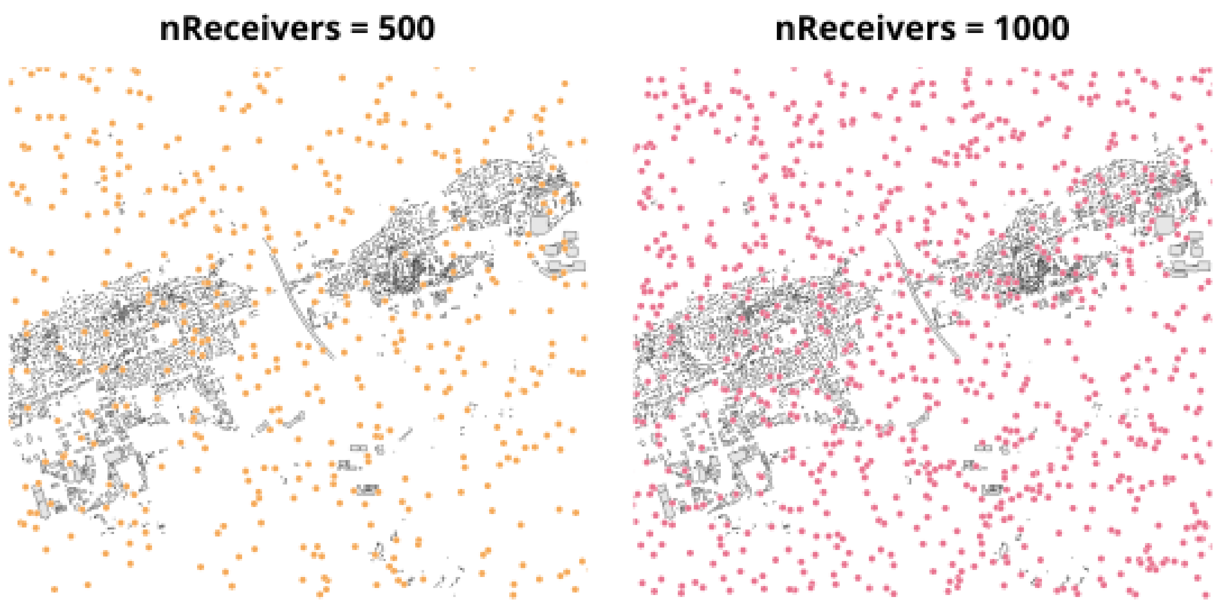
Type: `Double`

Default: `4`

`nReceivers` — ***Number of receivers***
Number of receivers to return

Type: `Integer`

Default: `100`



Output

result — *Created table*
: Name of the table containing the results of the computation. Can be used as input for another process.

  Type: `String`

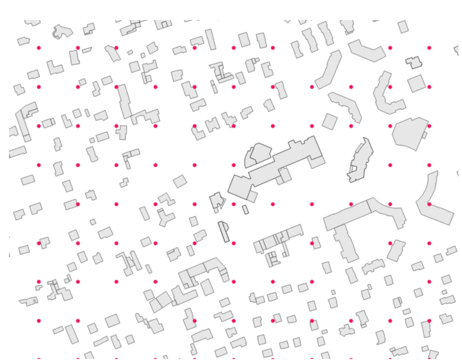
Regular Grid

Overview

Computes a regular grid of receivers. The receivers are spaced at a distance "delta" (Offset) in the Cartesian plane in meters. The grid will be based on:

- the BUILDINGS table extent (option by default)
- OR a single Geometry "fence" (see "Extent filter" parameter).

The output table is called RECEIVERS



Arguments

Mandatory inputs

`fenceTableName` — *Table bounding box name*
Using the bounding box of the given table name, define the envelope of the output grid:

- Extract the bounding box of the specified table,
- then create only receivers on the table bounding box.

The given table must contain:

- THE_GEOM : any geometry type with the appropriate SRID

Type: `String`

Optional inputs

`buildingTableName` — *Buildings table name*
Name of the Buildings table. Receivers inside buildings will be removed.The table must contain:

- THE_GEOM : the 2D geometry of the building (POLYGON or MULTIPOLYGON)

Type: `String`

`delta` — *Offset*
Offset in the Cartesian plane (in meters)

Type: `Double`

Default: `10`

`exportTrianglesGeometries` — *In the triangles table, export triangles geometries*
If enabled, the TRIANGLES table will contain the geometry of each triangle (outputTriangleTable must be enabled too)

Type: `Boolean`

Default: `false`

`fence` — *Extent filter*
Create receivers only in the provided polygon (fence)

Type: `Geometry`

`height` — *Height*
Height of receivers (in meter) (FLOAT)

Type: `Double`

Default: `4`

`outputTriangleTable` — *Output triangle table*
Output a triangle table in order to be used to generate iso contours with Create_Isosurface

Type: `Boolean`

`receiverstablename` — *Name of receivers table*
Name of the output table. Do not write the name of a table that contains a space

Type: `String`

Default: `RECEIVERS`

`sourcesTableName` — ***Sources table name***

Keep only receivers at least at 1 meters of provided sources geometries The given table must contain:

- THE_GEOM : any geometry type.

Type: `String`

Output

`result` — ***Created table***

Name of the table containing the results of the computation. Can be used as input for another process.

Type: `String`

5.25 Blocks

NoiseModelling provides processing blocks (referred to as **Blocks**) that encapsulate geospatial computations. These blocks are built on top of the OGC Web Processing Service (WPS) standard.

5.25.1 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.

5.25.2 NoiseModelling and WPS

Since release v.3.0.0, NoiseModelling comes with various scripts, encapsulated in so-called Blocks. These Blocks (also referred to as groovy scripts or `.groovy` files *(openable in any text editor)*) are written in the Groovy script language.

Physically stored in the `NoiseModelling/scripts` directory.

Tip: To know the functionality of each 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!

5.25.3 Create your own block

Please see *Create your own block*, because now you want to be one!

5.26 Builder

5.26.1 What is the Builder ?

The NoiseModelling Builder (referred to as the **Builder**) allows you to create 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 the Builder adapted to the needs of NoiseModelling. This version being very close to the original version initially developed by former BoundlessGEO company.

5.26.2 Frequently Asked Question

What do the colors correspond to?

- Orange blocks are mandatory
- Beige blocks are optional
- Green blocks are the output of the process*
- Blocks get solid border when they are ready/filled

Can I save my Builder project?

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

1. Export the blocks state into a JSON file
2. Export the blocks state and the whole database into a zip file (limited to 500 MB database)

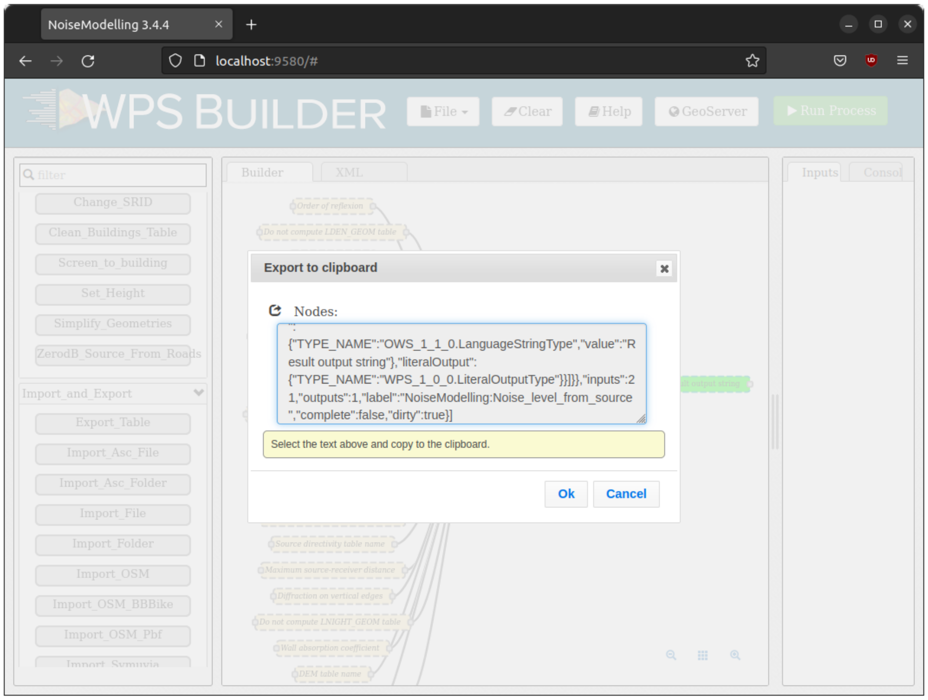
1. Export/Import the blocks state

Click on the `File` icon and then choose `Save project`. The browser will save the file in your download folder.

Once you want to recover the saved state, click on `File / Open project` and select the saved file.

2. Export/Import the database

Click on the `File` icon and then choose `Save project with database`. The browser will save the file in your download folder.



How to run multiple processing at once ?

You can use the output of a processing block (like `Import File`) as the input of another process. To do this keep the left button of your mouse down while dragging the white square on the right side of a green output block to the left white square of the input of another process. Then run the last process in the chain in order to execute the whole processing.

I want to run the same processing but using a script not using my web browser, how to do it ?

NoiseModelling WebServer is using the standard protocol named OGC Web Processing Service (WPS) Interface Standard. When you run a Block, the Builder generates an equivalent Python script in the Python tab of the user interface. You can just copy/paste the script in a Python console and it should work (no dependency) as long as the NoiseModelling software is running in the background.

The generated Python script is using the synchronous WPS execution, so the server will not respond until the process is done or after the 60 seconds default timeout.

If the timeout is reached it will always return a message “Long running process…” like in the Builder (but the job will still run on the server).

You can use the asynchronous WPS API so the server will return a message immediately with links to follow the progression of the execution of your job.

You can use the OwsLib Python library to do so, here is an example of how to do it:

Listing 1: List all available processes

```
import logging
import sys
from owslib.wps import WebProcessingService

```

(continues on next page)

(continued from previous page)

```python
def main():
    # Configure the root logger to output to stdout
    logging.basicConfig(stream=sys.stdout, level=logging.INFO)

    url = "http://localhost:8000/builder/ows"

    # use jwt cookie header for authentification
    jwt_token = ""
    wps = WebProcessingService(url, skip_caps=False, headers={"Authorization": jwt_token}
↪)

    # 2. List all available processes
    print(f"\n--- Available Processes ---")
    for process in wps.processes:
        print(f"ID: {process.identifier}")
        print(f"   Title: {process.title}")
        print(f"   Abstract: {process.abstract}\n")
        process_details = wps.describeprocess(process.identifier)
        print(f"   Inputs:")
        for data_input in process_details.dataInputs:
            print(f"      ID: {data_input.identifier}")
            if hasattr(data_input, 'title'):
                print(f"      Title: {data_input.title}")
            if hasattr(data_input, 'dataType'):
                print(f"      Data Type: {data_input.dataType}")
            if hasattr(data_input, 'minOccurs') and data_input.minOccurs == 0:
                print(f"      Optional")
            if hasattr(data_input, 'abstract'):
                print(f"      Abstract: {data_input.abstract}")
            if hasattr(data_input, 'allowedValues') and len(data_input.allowedValues) >␣
↪0:
                print(f"      Allowed values: {data_input.allowedValues}")
            if hasattr(data_input, 'defaultValue') and data_input.defaultValue is not␣
↪None:
                print(f"      Default : {data_input.defaultValue}")
            print("")

if __name__ == "__main__":
    main()
```

Listing 2: Execute a process

```python
import logging
import sys
from owslib.wps import WebProcessingService, monitorExecution

def main():
    # Configure the root logger to output to stdout
    logging.basicConfig(stream=sys.stdout, level=logging.INFO)

    url = "http://localhost:8000/builder/ows"

```

(continues on next page)

(continued from previous page)

```
    # use java web token for authentification
    jwt_token = ""
    wps = WebProcessingService(url, skip_caps=False, headers={"Authorization": jwt_token})

    target_id = "Database_Manager:Display_Database"

    inputs = [('showColumns', "true")]

    # Execute (sync or async)
    execution = wps.execute(target_id, inputs)

    # Monitor progress if it's a long task
    monitorExecution(execution, sleepSecs=2)

    if execution.isSucceded():
        for output in execution.processOutputs:
            print(f"Result {output.identifier}: {output.data}")
    else:
        print("Execution failed.")

if __name__ == "__main__":
    main()
```

5.27 Create your own block

5.27.1 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.

Scripts for NoiseModelling are written in Groovy language. They are located in the `NoiseModelling/scripts/` 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 Block is to be inspired by those that are already made. See how the tutorial is built or contact us for many more examples.

5.27.2 General Structure

1. Import used libraries

```
import groovy.sql.Sql
import java.sql.Connection
import org.h2gis.api.ProgressVisitor
```

2. Script meta data

```
title = 'My script title'
description = 'My script description, I support <b>html</b> !'
inputs = [
    my_optional_parameter: [name: 'option1', title: 'option1', description :
↪'Description, you can use html here. This parameter is optional because you have␣
↪provided a default value', type: String.class, default: 'MY_RESULT_TABLE'],
    my_integer_parameter: [name: 'my_integer_parameter', title: 'my_integer_parameter',␣
↪description : 'Restrict to an int, you can use html here', type: Integer.class],
    my_double_parameter: [name: 'my_double_parameter', title: 'my_double_parameter',␣
↪description : 'Restrict to an floating point value, you can use html here', type:␣
↪Double.class],
    my_boolean_parameter: [name: 'my_boolean_parameter', title: 'my_boolean_parameter',␣
↪description : 'A checkbox parameter', type: Boolean.class],
    my_choice_parameter: [name: 'my_choice_parameter', title: 'my_choice_parameter',␣
↪description : 'A list box with limited choices', type: String.class, allowedValues : [
↪"Choice 1", "Choice 2", "Choice 3"], default : "Choice 2"],
]

// Optional (default 60 seconds)
// For synchronous execution, it will wait this time (in seconds) before returning a␣
↪message, but it will still run the execution in the background
executionTimeout = 120


outputs = [
    result: [name: 'result', title: 'result', description : 'Result output, generally␣
↪the result output table name. You can use this output as an input for another␣
↪processing', type: String.class]
]
```

Note: You may want to add an optional parameter but without a default value (the input will not be in the input map) to do so instead of defining a default value `default: 'MY_RESULT_TABLE'` use the entry `min : 0`

4. Set main method to execute

```
/**
 * Main method
 * @param connection SQL Connection
 * @param input Map of inputs, should provide the same keys as described in the input␣
↪metadata
 * @param progress Can be used to display the progression of the computation, and to␣
↪check if the user canceled the execution
 * @return A map as described in the result metadata
 * @throws SQLException if something went wrong
 */
def exec(Connection connection, Map inputs, ProgressVisitor progress) {
    Sql sql = new Sql(connection)
    // Get the parameter value. For optional parameters, do not forget to provide a␣
↪default value in the metadata.
    def myTableName = inputs.my_optional_parameter
    sql.execute("CREATE TABLE IF NOT EXISTS $myTableName(PK SERIAL PRIMARY KEY, DATA␣
↪INTEGER)".toString())
    return [result : '$myTableName']
}
```

5. Make your method available

You have to save the script into the folder `NoiseModelling/scripts/`. You have to refresh the web page Builder in order to see your new script.

5.28 Access NoiseModelling database

5.28.1 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)*:

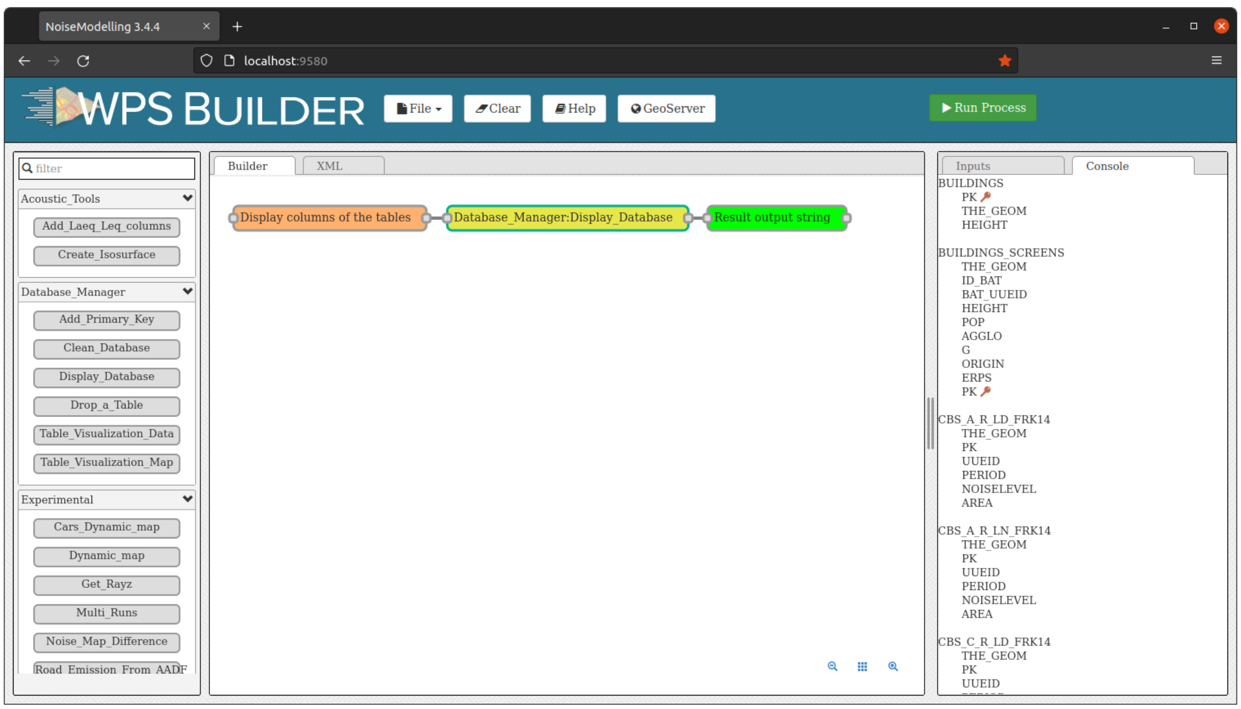
1. Use blocks (inside)
2. Use H2/H2GIS web client
3. Use DBeaver client

5.28.2 1. Use blocks

Once NoiseModelling UI is launched (open http://localhost:8000/ in your web browser), you can manage your data thanks to the `Database_Manager` 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` Block



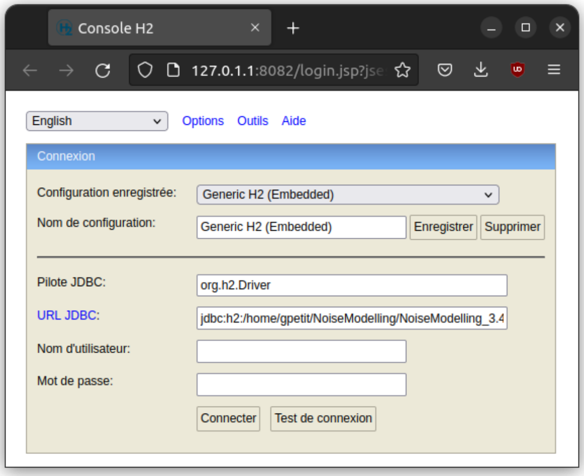
5.28.3 2. Use H2/H2GIS web client

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://127.0.1.1:8082/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`
- `JDBC URL`: the JDBC address of the NoiseModelling database. By default, this database is placed in here `~/.noisemodelling/user_001.mv.db`. So, fill this text area with `jdbc:h2:/~/.noisemodelling/user_001.mv.db`.
- `User name`: the db user name. By default, sa
- `Password`: the db password. By default, sa

> **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.

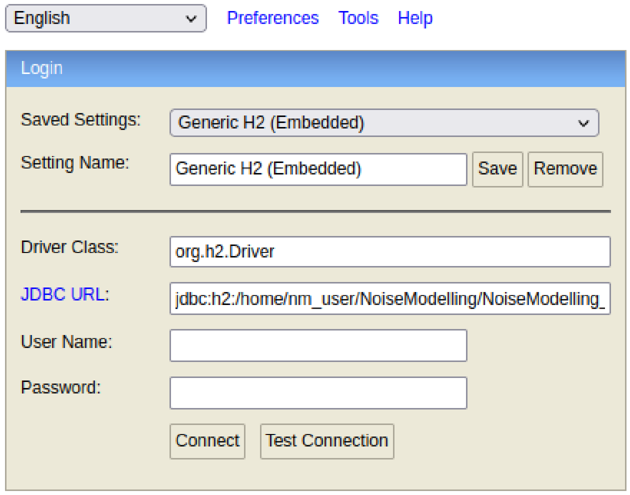
Below is an example, with a database located on the computer here: `/home/nm_user/.noisemodelling/user_001.mv.db`. We want to open the db while NoiseModelling is running.

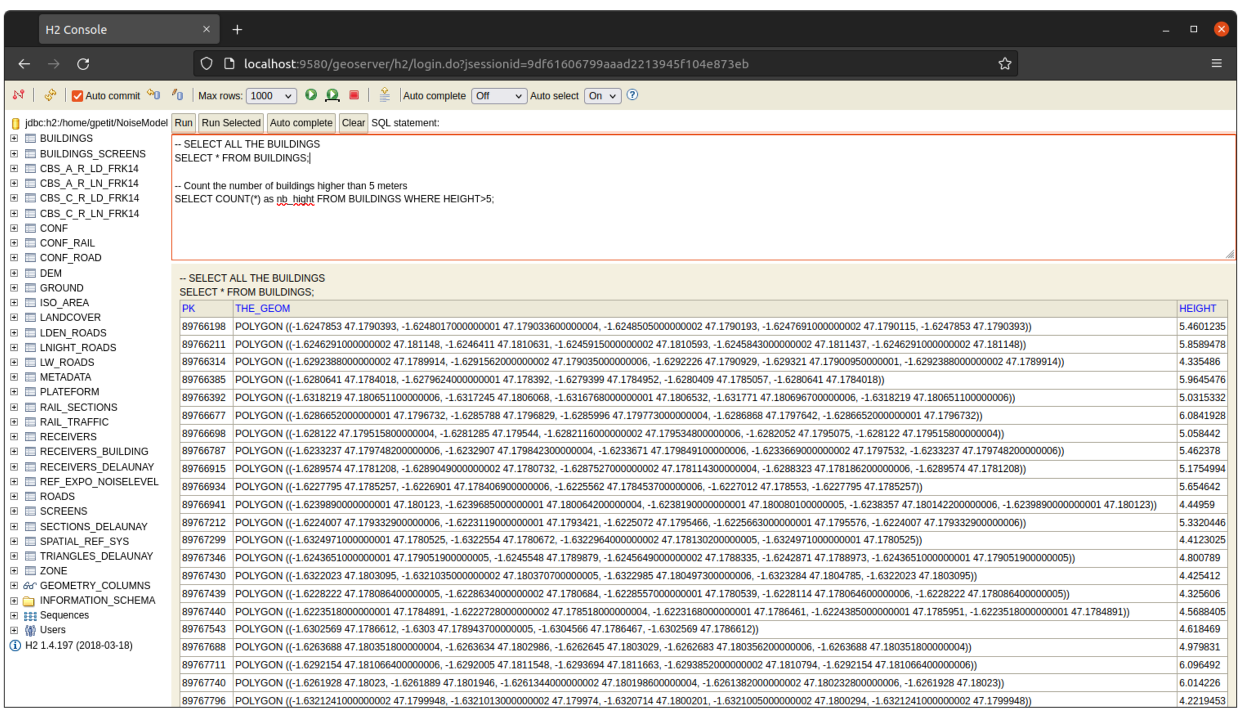
- `JDBC URL`: `jdbc:h2:/home/nm_user/.noisemodelling/user_001;AUTO_SERVER=TRUE`
- `User name`: *empty*
- `Password`: *empty*

> **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.


English
Preferences
Tools
Help
Login
Saved Settings:
Generic H2 (Embedded)
Setting Name:
Generic H2 (Embedded)
Save
Remove
Driver Class:
org.h2.Driver
JDBC URL:
jdbc:h2:/home/nm_user/NoiseModelling/NoiseModelling_
User Name:
Password:
Connect
Test Connection


H2 Console
localhost:9580/geoserver/h2/login.do?jsessionid=9df61606799aaad2213945f104e873eb
Auto commit
Max rows: 1000
Auto complete Off
Auto select On
Run
Run Selected
Auto complete
Clear
SQL statement:
-- 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;
BUILDINGS
BUILDINGS_SCREENS
CBS_A_R_LD_FRK14
CBS_A_R_LN_FRK14
CBS_C_R_LD_FRK14
CBS_C_R_LN_FRK14
CONF
CONF_RAIL
CONF_ROAD
DEM
GROUND
ISO_AREA
LANDCOVER
LDEN_ROADS
LNIGHT_ROADS
LW_ROADS
METADATA
PLATEFORM
RAIL_SECTIONS
RAIL_TRAFFIC
RECEIVERS
RECEIVERS_BUILDING
RECEIVERS_DELAUNAY
REF_EXPO_NOISELEVEL
ROADS
SCREENS
SECTIONS_DELAUNAY
SPATIAL_REF_SYS
TRIANGLES_DELAUNAY
ZONE
GEOMETRY_COLUMNS
INFORMATION_SCHEMA
Sequences
Users
H2 1.4.197 (2018-03-18)
PK
THE_GEOM
HEIGHT

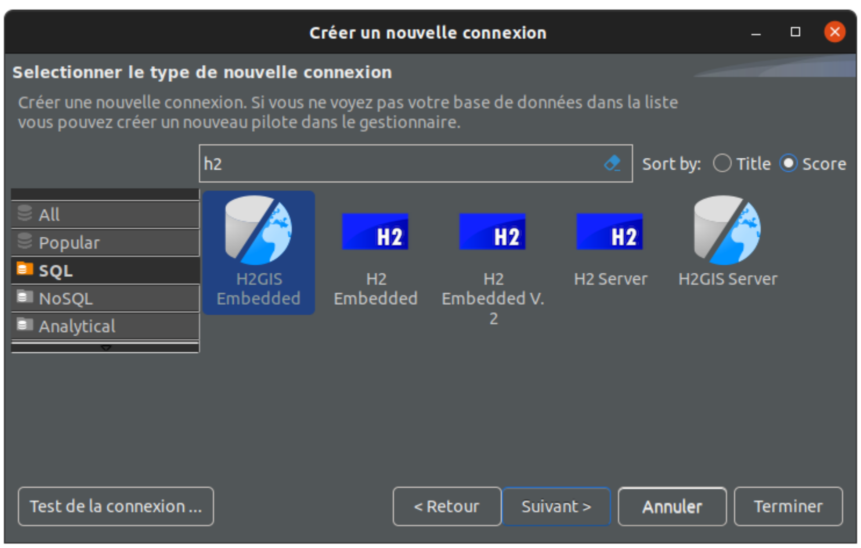
5.28.4 3. Use DBeaver client

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.

You can download DBeaver on this webpage.

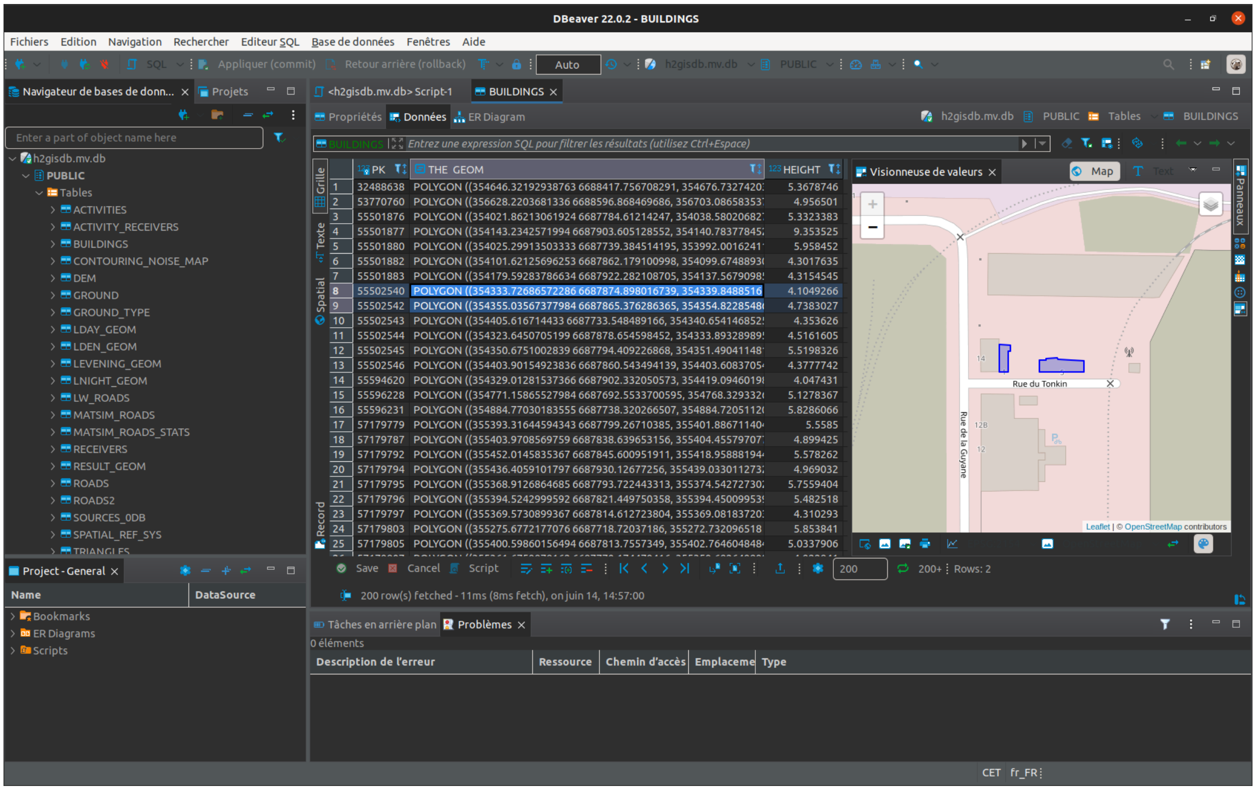
Connect DBeaver to your database

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` directory and is named `user_001.mv.db`.
5. In the `Path` text area, remove `.mv.db` at the end of the address
6. The default user is `sa` and password `sa`
7. 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/user_001;AUTO_SERVER=TRUE`)
8. Click on `Terminate` to open your dabatase!



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 `user_001.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.



5.29 Use NoiseModelling with a PostGIS database

5.29.1 Introduction

NoiseModelling is designed to use H2GIS as the default database.

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 want to use QGIS while using NoiseModelling).

Note: Using PostGIS may result in slower performance than H2GIS due to network overhead, as data is transferred over a connection rather than written directly to local storage.

NoiseModelling core functions has been written with the idea of maintaining the H2GIS/PostGIS compatibility.

Warning: NoiseModelling Groovy scripts may use incompatible syntax with PostGIS. We are currently working on checking the compatibility with the PostGIS database.

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

5.29.2 Using ScriptRunner with PostGIS

You can run Groovy scripts using a PostGIS connection instead of the default H2GIS embedded database by using the *ScriptRunner* program.

This is particularly useful for large-scale simulations where a dedicated database server is preferred.

Command Line Example

To import a Shapefile into a PostGIS database, you can use the following command:

```
./bin/ScriptRunner \
    -w /path/to/working/dir \
    -s src/main/groovy/org/noise_planet/noisemodelling/scripts/Import_and_Export/Import_
↪File.groovy \
    -d noisemodelling_db \
    -u noisemodelling \
    -p noisemodelling \
    --host localhost \
    --port 5432 \
    --pathFile /path/to/your/buildings.shp \
    --tableName BUILDINGS
```

Parameters Description

- **-w, –working-dir**: Path where the application logs and temporary files will be stored.
- **-s, –script**: Path to the Groovy script you want to execute.
- **-d, –database-name**: Name of the PostGIS database.
- **-u, –username**: Database username.
- **-p, –password**: Database password. If not provided and a host is specified, it will try to fetch it from the *.pgpass* file.
- **–host**: PostGIS database host name. Specifying this parameter tells NoiseModelling to connect to PostGIS instead of using H2GIS.
- **–port**: PostGIS database port (default: 5432).

Additional Script Parameters

Any parameter specific to the Groovy script (defined in its *inputs* map) can be passed as a double-dash argument. In the example above:

- **–pathFile**: The path to the file to import (required by *Import_File.groovy*).
- **–tableName**: The name of the table to create in the database.

Using .pgpass for Credentials

If you don't want to provide the password in the command line, you can use a *.pgpass* file as specified in the PostgreSQL documentation.

The file should contain lines in the following format:

```
hostname:port:database:username:password
```

NoiseModelling will automatically fetch the password if the host is specified and the password parameter is omitted.

5.30 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 5 main libraries:
   - `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
   - `noisemodelling-scripts` : Host the web server, the Groovy scripts (Blocks) and the final packaging instructions
4. Enjoy & feel free to contact us!

Important: A Java documentation is available here

5.31 Conformity to ISO 17534-1:2015

5.31.1 Clarifications on the ISO Standard

It is important to note that the ISO standard provides recommendations rather than regulatory obligations. While it serves as a reference framework, its application is not mandatory from a legal standpoint.

5.31.2 Conformity table

Conform

- Do not the deviate more than ±0,1 dB
- Percentage of conformity : 100% (28/28)

NLD Conform

- Do not the deviate more than ±0,1 dB neglecting lateral diffraction
- Percentage of conformity : 100% (28/28)

Test Case	Conform ?	NLD Conform ?	Largest Deviation	Details
TC01			0.00 dB @ 2000 Hz	*TC01*
TC02			0.07 dB @ 1000 Hz	*TC02*
TC03			0.05 dB @ 500 Hz	*TC03*
TC04			0.07 dB @ 1000 Hz	*TC04*
TC05			0.00 dB @ 1000 Hz	*TC05*
TC06			0.06 dB @ 500 Hz	*TC06*
TC07			0.02 dB @ 1000 Hz	*TC07*
TC08			0.02 dB @ 1000 Hz	*TC08*
TC09			0.01 dB @ 250 Hz	*TC09*
TC10			0.02 dB @ 250 Hz	*TC10*
TC11			0.01 dB @ 125 Hz	*TC11*
TC12			0.04 dB @ 500 Hz	*TC12*
TC13			0.01 dB @ 500 Hz	*TC13*
TC14			0.04 dB @ 8000 Hz	*TC14*
TC15			0.03 dB @ 125 Hz	*TC15*
TC16			0.00 dB @ 2000 Hz	*TC16*
TC17			0.05 dB @ 500 Hz	*TC17*
TC18			0.02 dB @ 500 Hz	*TC18*
TC19			0.02 dB @ 63 Hz	*TC19*
TC20			0.00 dB @ 8000 Hz	*TC20*
TC21			0.02 dB @ 8000 Hz	*TC21*
TC22			0.00 dB @ 125 Hz	*TC22*
TC23			0.02 dB @ 63 Hz	*TC23*
TC24			0.01 dB @ 250 Hz	*TC24*
TC25			0.01 dB @ 2000 Hz	*TC25*
TC26			0.02 dB @ 4000 Hz	*TC26*
TC27			0.02 dB @ 1000 Hz	*TC27*
TC28			0.01 dB @ 4000 Hz	*TC28*

5.31.3 TC01

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	1000
L	0.00 dB	500

5.31.4 TC02

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.05 dB	1000
L	0.07 dB	1000

5.31.5 TC03

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.08 dB	1000
L	0.05 dB	500

5.31.6 TC04

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.06 dB	1000
L	0.07 dB	1000

5.31.7 TC05

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	1000
L	0.00 dB	1000

5.31.8 TC06

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.04 dB	500
L	0.07 dB	500

5.31.9 TC07

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.06 dB	1000
L	0.00 dB	250

5.31.10 TC08

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.06 dB	1000
L	0.01 dB	1000

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.08 dB	2000
L	0.06 dB	1000

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.03 dB	1000
L	0.04 dB	1000

5.31.11 TC09

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	500
L	0.00 dB	63

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.03 dB	500
L	0.01 dB	125

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.01 dB	250
L	0.01 dB	250

5.31.12 TC10

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	500
L	0.00 dB	500

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.02 dB	125
L	0.02 dB	125

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.02 dB	125
L	0.02 dB	125

5.31.13 TC11

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	4000
L	0.00 dB	4000

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.03 dB	500
L	0.03 dB	500

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.03 dB	500
L	0.03 dB	500

5.31.14 TC12

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.02 dB	250
L	0.02 dB	250

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.06 dB	4000
L	0.06 dB	4000

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.04 dB	1000
L	0.04 dB	1000

5.31.15 TC13

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.01 dB	1000
L	0.00 dB	8000

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.01 dB	125
L	0.01 dB	125

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.00 dB	125
L	0.00 dB	125

5.31.16 TC14

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.04 dB	8000
L	0.04 dB	4000

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.06 dB	8000
L	0.06 dB	8000

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.08 dB	2000
L	0.08 dB	2000

5.31.17 TC15

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.01 dB	125
L	0.00 dB	1000

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.03 dB	63
L	0.03 dB	63

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.04 dB	8000
L	0.04 dB	8000

5.31.18 TC16

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	1000
L	0.00 dB	1000

Reflection

Parameters	Maximum Difference	Frequency
L	0.00 dB	63
L	0.00 dB	4000

5.31.19 TC17

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.04 dB	500
L	0.07 dB	500

Reflection

Parameters	Maximum Difference	Frequency
L	0.04 dB	500
L	0.03 dB	500

5.31.20 TC18

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.06 dB	500
L	0.00 dB	2000

Reflection

Parameters	Maximum Difference	Frequency
L	0.00 dB	250
L	0.00 dB	1000

5.31.21 TC19

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.02 dB	63
L	0.02 dB	63

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.07 dB	500
L	0.01 dB	125

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.04 dB	500
L	0.01 dB	8000

5.31.22 TC20

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	8000
L	0.00 dB	8000

5.31.23 TC21

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.02 dB	125
L	0.02 dB	250

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.08 dB	8000
L	•	•

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.07 dB	500
L	•	•

5.31.24 TC22

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.01 dB	4000
L	0.00 dB	4000

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.06 dB	500
L	0.02 dB	125

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.05 dB	500
L	0.01 dB	125

5.31.25 TC23

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.02 dB	250
L	0.02 dB	250

5.31.26 TC24

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.02 dB	250
L	0.02 dB	63

Reflection

Parameters	Maximum Difference	Frequency
L	0.02 dB	63
L	0.02 dB	63

5.31.27 TC25

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	1000
L	0.01 dB	250

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.01 dB	1000
L	0.01 dB	1000

Left Lateral

Parameters	Maximum Difference	Frequency
L	0.01 dB	2000
L	0.01 dB	2000

Reflection

Parameters	Maximum Difference	Frequency
L	0.00 dB	1000
L	0.00 dB	2000

5.31.28 TC26

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.00 dB	4000
L	0.06 dB	4000

Reflection

Parameters	Maximum Difference	Frequency
L	0.01 dB	4000

5.31.29 TC27

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.02 dB	2000
L	0.03 dB	1000

Reflection

Parameters	Maximum Difference	Frequency
L	0.03 dB	125
L	0.08 dB	4000

5.31.30 TC28

Vertical Plane

Parameters	Maximum Difference	Frequency
L	0.01 dB	63
L	0.01 dB	8000

Right Lateral

Parameters	Maximum Difference	Frequency
L	0.09 dB	2000
L	0.01 dB	250

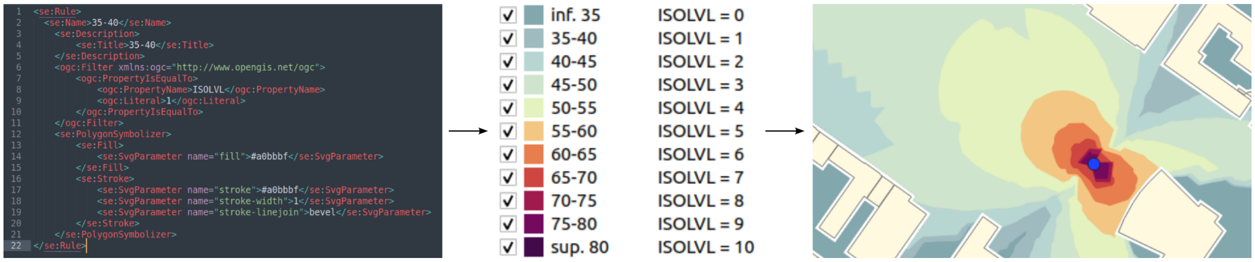
Left Lateral

Parameters	Maximum Difference	Frequency
L	0.08 dB	2000
L	0.01 dB	1000

5.32 Noise Map Color Scheme

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

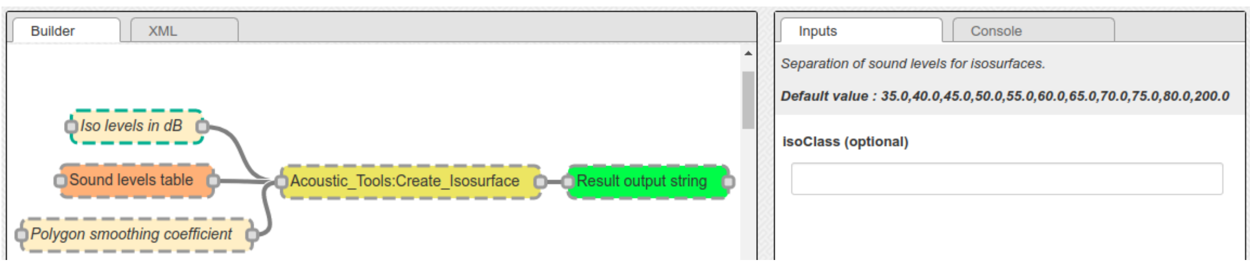
Note: If you want to feed this list with other schemes, please contact us (see *Support* page).



5.32.1 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.



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 7: 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.

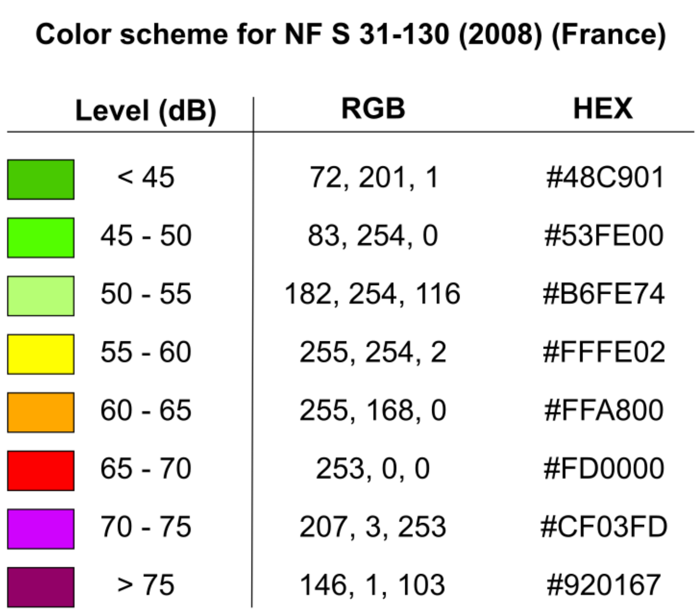
To know how to load an `.sld` file, you can also consult the NoiseModelling tutorial *Noise Map from Point Source - GUI* in the section "`Step 3 - Apply a color palette adapted to acoustics`"

5.32.2 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

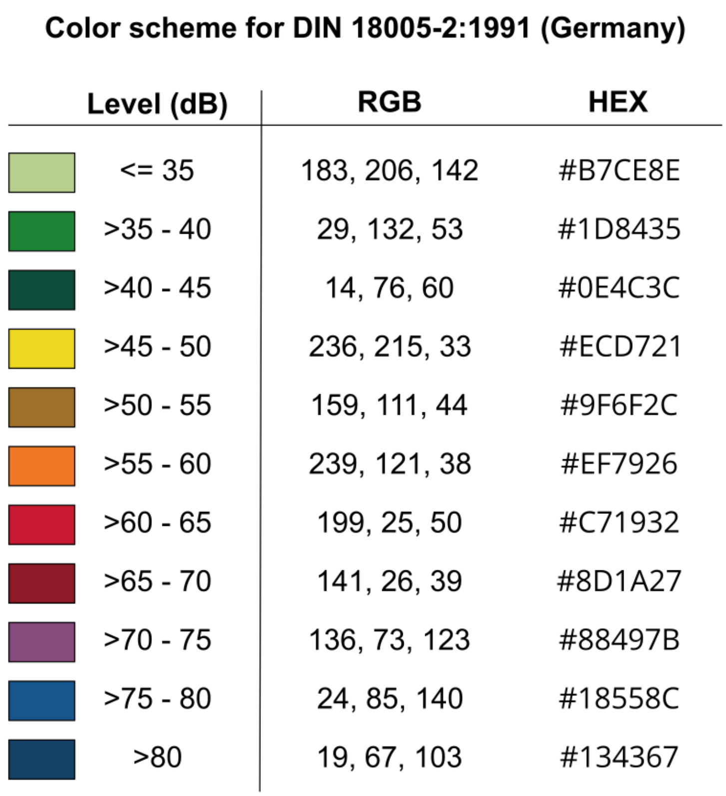
> **Warning:** This style will work only if you specified `Iso levels in dB` = `45, 50, 55, 60, 65, 70, 75, 200` when exectuting the `Acoustic_Tools:Create_Isosurface` script

5.32.3 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

5.32.4 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 8: 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

5.32.5 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

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

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

5.32.6 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>
```

(continues on next page)

(continued from previous page)

```
    </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>
```

5.33 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 !

5.34 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).

5.35 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

SIX

INDICES AND TABLES

- genindex
- modindex
- search