osmTGmod 0.1.1

Documentation, Version: 0.1.0

Malte Scharf, Arjuna Nebel

Wuppertal Institute

March 29, 2016

Contents

1. About osmTGmod

osmTGmod is a load-flow model of the German transmission-gird, based on the free geo-database *OpenStreetMap (OSM)*. The model, respectively the abstraction process employs a *PostgreSQL*-Database extended by *PostGIS*. The key part of the abstraction process is written in the languages *SQL* and *pl/pgSQL*, provided by the *PostgreSQL*-Database. The abstraction and all additional modules are controlled by a *Python*-environment.

2. License

osmTGmod is a free, open-source code and builds on OpenStreetMap data. The OpenStreetMap data is available under the Open Database License (ODbL) and OpenStreetMap cartography is licensed as CC-BY-SA. For further information on the copyright of OpenStreetMap, please refer to OpenStreetMap-Wiki (2015). All data and databases delivered with **osmTGmod** are made available under the Open Database License. Any rights in individual contents of the database are licensed under the Database Contents License. You can also redistribute and/or modify the data distributed with **osmTGmod** under the same licenses and copyright.

The **osmTGmod** code and this documentation are licensed under the Apache License, Version 2.0. Please visit the webpage for more information concerning the Apache License and for a description of the terms under which you can use the **osmTGmod** code.

Limitations of liability: in no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any contributor to **osmTGmod** be liable to any user for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such contributor has been advised of the possibility of such damages.

3. Installation and Requirements

3.1. Installation

osmTGmod employs additional software that needs to be installed (see section 3.2). **osmTGmod** itself is a *Python*-script (respectively a *Python*-module) that doesn't require any installation. **osmTGmod** is developed and tested on *Mac*.

3.2. Additional Software

In order to use **osmTGmod**, the following software are is required (**osmTGmod** has been tested with the stated versions, but should work fine with similar versions):

- *PostgreSQL 9.4.4*-Database-server (local or remote server).
- Database extensions: *PostGIS 2.1.7* and *pgRouting 2.0.0*
- *QGis 2.1* (or newer)
- *Python 2.7.10*

All necessary software is open-source and can be downloaded for free.

osmosis 0.44 is a command line java application which is used within osmTGmod. Therefore *Java 1.6* (or newer) is required as well.

3.3. Folder structure

osmTGmod has four folders:

- code
- raw_data
- qgis_projects
- results

The folder *code* contains all the necessary functions to run osmTGmod. The subfolder *osmosis* has a stable copy of the java tool *osmosis 0.44* which is needed to filter and upload data to the database.

The folder *raw_data* is the place, where all raw and filtered osm data files shall be stored.

Within the folder *qgis_projects* automatic generated local qgis files are saved.

The folder *results* contains the grid results as csv data.

4. OpenStreetMap

OpenSteetMap (OSM) is 'a free and editable map of the world' (Bennet, 2010). It is a Voluntary Geographic Information (VGI) project aiming to build a free and accessible geographic database of the the world. Created in 2004 by Steve Cost, OSM database is continuously growing in size, map and data details, data diversity and number of users [1]. OSM data includes geo-referenced features which can be mapped. It is collected on a voluntary basis by data collectors or "mappers". There exist many ways to collect geo-references data and link it to OSM. The most common ways are using Global Positioning System (GPS) receivers or satellite pictures provided by *Microsoft Bing*. Further ways include using data which license expired, data which is donated to OSM and data made available by governmental entities and communities.

In OSM, power data is represented by the three OSM data type, namely `nodes`, `ways` and `relations`. Nodes represent line-carrying towers and electrical poles. Transmission lines and underground cables are represented by open ways, while substations, generators and power plants are closed ways. Power relations represent electrical circuits and are constituted of one or several towers, substations and transmission lines. Such circuit-relations have the `key=value` combination `route=power`.

5. How to use osmTGmod

5.1. Getting started / Create a new osmTGmod-database

After installing additional software, **osmTGmod** can be used. In most cases it is recommended to execute **osmTGmod** as a script. It works as a command-line program that guides the user through its functionality. If you intend to use **osmTGmod** as a Python module, please refer to section 7.

To use **osmTGmod** as a command-line program, simply execute the file *osmTGmod.py* in the programming folder.

At first, the user hast to provide the following connection-parameters to his PostgreSQL-Database (see listing 1):

```
osmTGmod started as Script
Please provide connection-parameters:
database name: my_desired_database_name
password: my_server_password
```

[1] http://wiki.openstreetmap.org/wiki/Stats

```
QGis-Processing Path (default: .../user/.qgis2/processing/scripts/): my_qgis_path
host (default 192.168.0.46):
port (default 5432):
user (default postgres):
```

Listing 1: **osmTGmod** connection parameters

- `database name:` If **osmTGmod** is executed for the first time, the user will still not have a grid-model database. In this case simply type the desired database-name and provide the general connection-parameters to the database-server.
- `password:` Password for connection to database server.
- `QGis-Processing Path:` QGis software is employed for viewing abstraction-results and inserting grid-development data. For this purpose, **osmTGmod** uses QGis processing scripts that are copied to the respective folder (so they appear in the Qgis processing toolbox). The proposed value (default) is generated from system information and should work fine in most cases.
- `host, port, user:` These entries have default values and are only required if they are different from the defaults.

After providing connection parameters, **osmTGmod** connects to server and database. In case no database ca be found under the given name, the user is asked if he wishes to create a new grid-model-database. This will create and build-up a new database and import all **osmTGmod** functions, tables and data (except OSM-data) (see listing 2).

```
Connecting to Server...
Connected to Server!
Connecting to Database...

There is no Database named 'my_desired_database_name'!
Do you wish to create Database 'my_desired_database_name' and build up osmTgmod-
    database?: yes

OK lets go! Creating Database...
New Database Created!
Connecting to new Database...
Connected to Database!
Checking Database-Status...
Database not ready! Building up Database...
```

```
Loading Functions and Result-Schema...
Executing SQL-file extensions.sql...
...
osmTGmod-database has been built up!

Checking OSM-Data
No OSM-Data found in Database!
Do you wish to load OSM-Data into Database?: yes
```

Listing 2: Building up grid-model database

After building up the **osmTGmod**-database, OSM-data needs to be imported. The OSM Import/Update/Filter-Dialog opens (see 5.2.1).

After building up the grid-model-database and importing OSM-data, **osmTGmod** ist ready to use and the main menu appears (see listing 3).

If **osmTGmod** is started again with the same database, it will connect to the previously built-up database an directly move to the main menu.

5.2. The main menu

The main menu (listing 3) appears after building up a new database (see steps in 5.1) or connecting to any existing **osmTGmod**-database.

```
Grid Model is ready to use!

          Options:

          1. Import/Update/Filter OSM-Data

          2. Start an Abstraction

          3. Export Results to CSV

          4. View Results/OSM - Metadata, Grid-Devel.-Plans

          5. Insert new (empty) Grid-Development-Plan

          6. Copy Development-Measures from one Grid-Development-Plan to another

          7. Quit

```

```
What would you like to do?
```

Listing 3: **osmTGmod** main menu

5.2.1. Import/Update/Filter OSM-Data

In this dialog, OSM-data can be downloaded, filtered and imported into the **osmTGmod**-database. Import overwrites existing OSM-data in the database and thus serves as an update-option.

```
Loading/updating OSM Data...

        1. Download, filter and import OSM-Data

        2. Filter OSM-Data

        3. Import an existing (filtered) OSM-file into Database
```

Listing 4: Download filter and import OSM-data

- `1:` **osmTGmod** automatically downloads the latest version of OSM-data from `http://download.geofabrik.de`, filters it and imports the filtered OSM-data into the database. If the computer a good internet connection (the file size is approx. 2GB) this option is preferable.

- `2:` With this option the user can manually filter a previously downloaded `osm.pbf`-file. The file to be filtered needs to be in programming/raw_data-folder. The user is asked for the filename of the the unfiltered `osm.pbf`-file and the desired filename for the filtered data (see listing 5).

```
Info: Unfiltered osm.pbf-File needs to be in "raw_data"-Directory

Name of raw OSM-file (e.g. germany-latest.osm.pbf): raw_filename.osm.pbf
Name of filtered (destination) OSM-file (e.g. germany-latest-filtered.osm.
    pbf): filtered_filename.osm.pbf
Filtering OSM-Data...
```

Listing 5: Filtering OSM-data

- `3:` With this option the user can import a previously downloaded and filtered `osm.pbf`-file into the database (see listing 6).

```
Info: Filtered osm.pbf-File needs to be in "raw_data"-Directory

Name of filtered OSM-file: filtered_filename.osm.pbf
Download Date (E.g. 2015-10-23): 2015-11-11
```

Listing 6: Importing OSM-data

5.2.2. Start an Abstraction

With this option an abstraction process of OSM-data is started. Before the abstraction starts, a selected grid development plan is applied to the OSM-data so that the final grid contains the development measures of the respective plan (see listing 7). Depending on the performance of the database server, one abstraction process may take approximately 30 minutes. The results of the abstraction are saved to the results database and can be accessed via the `Export results to CSV option` (see 5.2.3).

```
Development plan IDs (e.g. 1;2) (default None): 1;2
Year of application (e.g. 2020) (default None): 2020
Substation with slack node (OSM-ID) (default 35176751): 35176751
User comment: My abstraction
```

Listing 7: Starting an Abstraction

- `Development Plan IDs:` The user is asked to enter the development plan IDs, separated by semicolon. The development plan must previously exist and its development measures need to be entered in the database (see 6). If the user intends to not apply any grid development, the field should be left empty (defaults to `None`).
- `Year of application:` Furthermore, the year of application (of the development measures) must be entered. All measure of the selected plans before or equal this year will be applied to the OSM-data and will be available in the final grid. This field should be left empty in case no grid development is applied.
- `Substation with slack node:` **osmTGmod** ensures that the final grid is connected and deletes not connected parts of the grid (that don't belong to the main graph). The nodes of the "substation with slack node" are considered as a nodes of the main graph. Furthermore one of the nodes of this substation is marked as "slack node" in the result database.

- `User comment:` The user comment can be used to further specify this abstraction. Besides OSM metadata, timestamps etc., the comment appears in the metadata table of the results database.

Apply grid development

After entering grid development measures (see section 6), these measures can be applied to the OSM while abstracting the grid. To do this, the above mentioned values `Development Plan IDs` and `Year of application` must be entered (duplicate measures, that are listed in more than one development plan, are only applied once). However, the application of development measures to the OSM data is prone to error. Since OSM is a constantly changing database, objects may be deleted or changed. Every time the user updates the OSM data this may occur. For this reason, **osmTGmod** checks, if the corresponding objects still exist. In case any object (to which a measure should be applied) doesn't exist (anymore), the process stops and the error message `ID ... nicht in power_ways zu finden` is displayed. In this case, the development measures must be adapted to the (newer) version of OSM data.

5.2.3. Export Results to CSV

With this option a subset of the results database can be exported to csv tables. For this, at least one abstraction must to be executed first (otherwise the results database is empty). The content of the exported tables is documented in chapter 8.

```
Result ID: 1
Result path (default: .../results/): desired_path
writing branch_data...
[...]
```

Listing 8: Export results to csv-tables

- `Result ID:` Each time an abstraction is executed, a new result ID is created and the information on the results is stored in the results metadata. The user can view the results metadata (and thus the ID) by selecting the `View Results/OSM-Metadata , Grid-Devel.-Plans` option in the main menu (see 5.2.4). The results matching the entered ID will be exported.

- `Filepath:` Complete Path to the desired results folder.

5.2.4. View Results/OSM-Metadata , Grid-Devel.-Plans

With this option the user receives information on the most important aspects of the **osmTG-mod** database.

```
        You can display the following information:

        1. OSM - Metadata

        2. Grid-Development-Plans

        3. Result-Metadata
```

Listing 9: View information

- `1:` Information on the download-date and import-date of the OSM-data:

```
Downloaded:  2015-12-03
Imported: 2015-12-03
```

Listing 10: OSM metadata

- `2:` Information on the inserted grid development plans:

```
ID:  1
Grid-Development-Plan: EnlaG_2015
Comment: Stand des Ausbaus nach dem Energieleitungsausbaugesetz (EnLAG) zum
     dritten Quartal 2015
```

Listing 11: View grid development plans

- `3:` View results metadata:

```
ID:  1
OSM-Downloaded: 2015-12-03
Abstraction Date: 2015-12-03
Applied Devel. Plans: None
Applied Devel. Year: None
User Comment: Erster Durchlauf zum Test (kein Zubau)
```

Listing 12: View results metadata

5.2.5. Insert new (empty) Grid Development Plan

With this option the user can insert a new (empty) development plan. This plan can be used to enter development measures via QGis (see 6):

```
New ID of Development-Plan: 2
Name of new (empty) Development-Plan (e.g. EnlaG): NEP2014
Comment: my comment
Plan successfully inserted!
```

Listing 13: Insert new development plan

- `New ID of Development-Plan:` The new development plan will have this ID
- `Name of new Development-Plan:` Acronym of the new development plan.

5.2.6. Copy Development-Measures from one Grid-Development-Plan to another (expert)

This option can be used to copy already existing plans respectively the provided measures by this plan into another plan. If e.g. a new NEP is developed, all measure of the old NEP can be copied into the new one. Thus, not all measure have to be entered again via QGis. In the current version of **osmTGmod**, please use this option with care since the correct results depend on correct user input (no database protection has been implemented).

```
ID of Plan FROM which measures will be copied: 1
ID of Plan TO which measures will be copied (Plan must exist): 2
Plan successfully copied!
```

Listing 14: Copy Development-measures

6. Grid Development

osmTGmod allows the user to apply certain grid development plan to the OSM data so that the abstracted grid contains the corresponding development measures. All measure must be manually entered into the **osmTGmod** database via QGis. As soon as this process is complete, the corresponding grid development plan can be selected when executing an abstraction (see section 5.2.2).

6.1. Getting started

In general, it is recommended to work with two independent QGis projects (one for entering development measures, and one for viewing the desired results). After creating a new **osmTGmod** database, these QGis projects are automatically created and can be found in the **osmTGmod** `qgis_projects` folder.

Before the user can start entering grid development measures via QGis the following steps must be performed:

1. Close Qgis
2. Start **osmTGmod** as a script (processing scripts are copied to the QGis processing folder)
3. Create a new development plan via the option `Insert new (empty) Grid Development Plan` of the main menu (see section5.2.5)
4. Open the Qgis project-file `your_database_grid_devel_project.qgs` in the **osmTGmod** `qgis_projects` folder.
5. Check if the processing scripts `metadata grid development`, `new relation` and `show result` can be found in the QGis processing toolbox. If this is not the case, close QGis, restart **osmTGmod** and check if the QGis processing path is correct (see section 5.1).
6. Save the QGis project, so that the above mentioned steps don't need to be performed again.

Furthermore it is recommended to use the QGis-OpenStreetMap Plugin to display a helpful map background.
Transmission lines and cables (OSM objects with the tags `power=line` and `power=cable`) and substations (`power=substation`) are represented by the data-type (object class) *way* in OSM. These objects are represented by the by the **osmTGmod** table `power_ways`.

Electrical circuits are implemented in OSM as *relation* and are represented by the **osmTGmod** table `edit_power_relations`

`vw_change_log` is a view that allows the user to view the entered development measures.

The tables `power_ways` and `edit_power_relations` are used to enter measures. The user inserts or deletes or modifies objects of these tables. However, these actions have no direct effect on these tables. Instead, the actions are recorded and saved in the database. The user can view and modify these changes via the view `vw_change_log`.

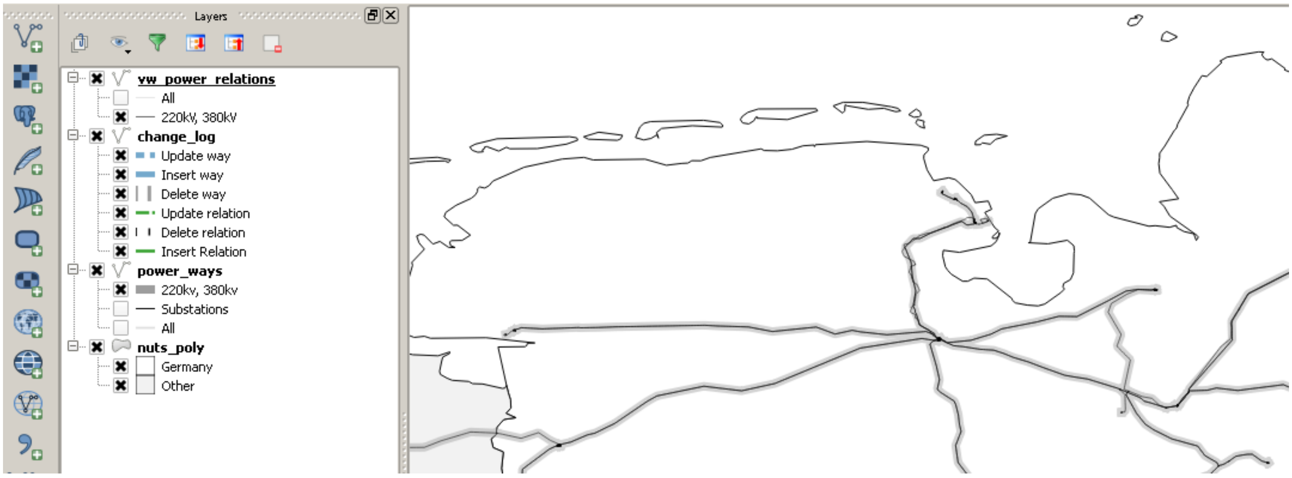

Figure 1: Loaded layers in QGis in order to enter development measures (layer names may vary from this documentation)

With every abstraction, the user can choose grid development plans (measures that have been previously saved in the database) that are applied to the OSM data.

6.2. Set Metadata

Before the user may start entering grid development measures, its metadata must be set. **IMPORTANT: For every single measure the user enters the metadata needs to be updated!** For this the processing script `metadata_grid_development.py` (find script in QGis processing toolbox) must be executed. This will open a new window in which the user may enter the following values:

- `plan id`: Enter the IDs of the development plans (separated by semicolon) to which the measure belongs.
- `year`: Year in which the measure is planned to implemented (according to the plan-document). Only one year can be set (no separation by semicolon).
- `plan intern id`: Grid development plans use internal IDs to identify the different measure (e.g. 34c). Enter these IDs here to better identify the measures. **IMPORTANT: The number of semicolons of the `plan intern id` and `plan id` must be equal (plan intern ids belong to the plans entered in `plan id`)!**
- `description`: Description of the corresponding measure.
- `user_comment`: To indicate problems e.g. incomplete digitizing.

Confirm the entered values by clicking on `Run`. For all in the following entered development measures, this metadata is applied until the user runs the `metadata_grid_development.py` again.

The view `vw_change_log` that shows the entered measures is also dependent on the entered metadata. **IMPORTANT: Only the measures of the corresponding development plans (set in metadata) are displayed.**

6.3. Entering Grid Development measures

After setting the metadata (see section 6.2), the user can enter a development measure. In general, the user accesses the above mentioned QGis layers (`power_ways` and `edit_power_relations`), enables QGis editing (by clicking on the pencil symbol) and enters the desired measure. After clicking on `save layer edits` or disabling editing, the entered actions are saved in the database (but not directly applied to the tables (see section 6.1)).

6.3.1. Basic Actions

In order to enter certain development measures (e.g. modification of existing transmission lines), the user must perform a series of basic actions. In the following, the 7 basic actions of QGis editing are explained:

1. **Modify a transmission line:** Transmission lines (OSM way objects) are represented by the table `power_ways`. In the context of a development measure, a transmission can be modified. To do this, perform the following steps:
   a) Enable editing of layer `power_ways`.
   b) Identify the transmission line with the `Identify Feature` tool (If `Auto Open Form` is enabled, the feature form will pop um after identifying an object).
   c) In case the from doesn't pop up automatically, open it manually by right-clicking on the desired object-ID in the `Identify Results` panel and choosing `View Feature Form`.
   d) In the feature form the attributes `power`, `voltage`, `cables`, `wires`, `circuits`, `frequency` and `name` can be changed. All values must have the same syntax as in OSM (e.g. `voltage=380000;220000` for a transmission line with a 380 kV and a 220 kV level [2]).

[2] DC-lines are represented by `frequency=0`. Germany railway lines have `frequency=16.7`. in OSM, multiple

e) Close the form and save the changes by clicking on `save layer edits`. The modification will be saved in the database and displayed in the (`vw_change_log` layer. The changes have no direct effect on the `power_ways` object itself.

The geometric change of transmission lines is not implemented (since this is rarely the case in development measures). In this case, delete an existing transmission line and insert a new one with the desired geometry

2. **Insert a new transmission line:** New transmission lines can be inserted with the following steps:

   a) Enable editing of layer `power_ways`.

   b) Digitize the geometry of the new transmission line with the `Add Feature` tool. It is highly recommended to use the QGis snapping option (enable `Snap to Vertex` and set `Tolerance` to approx. 15 pixels). In order to connect a transmission line to a certain substation, the start- or end-vertex of the line geometry must be inside the corresponding substation area.

   c) After digitizing the line geometry, the feature form pops up and allows the user to enter the attributes `power`, `voltage`, `cables`, `wires`, `circuits`, `frequency` and `name`. All values must have the same syntax as in OSM (see above).

   d) Close the form and save the changes by clicking on `save layer edits`. The new transmission line will be saved in the database and displayed in the (`vw_change_log` layer. The changes have no direct effect on the `power_ways` table itself.

3. **Delete a transmission line:** Deconstructing transmission lines as part of a development measure requires deleting transmission lines. This can be done with the following steps:

   a) Enable editing of layer `power_ways`.

   b) Select a transmission line with the `Select Features` tool (not the `Identify Feature` tool!).

   c) Delete the line by pressing delete on your keyboard

   d) Save the changes by clicking on `save layer edits`. The deletion of the transmission line will be saved in the database and displayed in the (`vw_change_log` layer. The deletion has no direct effect on the `power_ways` table itself.

frequencies on a transmission line are separated by semicolon (e.g. `frequency=50;16.7`). However **osmTGmod** can not process this information (this is, because different frequencies are simply listed up, but can not be clearly assigned to the voltage levels of a transmission line). In this case, circuit-relations are necessary to provide the lacking information.

4. **Modify a circuit-relation:** Normally, a transmission line is overlaid by a "circuit" (relation) object (represented by the table `edit_power_relations`). Relations provide the information on the wiring of transmission lines. In OSM, circuits and transmission lines provide redundant information. Thus, if e.g. the voltage of a transmission line is changed (as part of a development measure), the circuit (relation) voltage must be changed as well. A circuit-relation can be modified with the following steps:

   a) Enable editing of layer `edit_power_relations`.

   b) Identify the circuit-relation with the `Identify Feature` tool (If `Auto Open Form` is enabled, the feature form will pop um after identifying an object).

   c) In case the from doesn't pop up automatically, open it manually by right-clicking on the desired object-ID in the `Identify Results` panel and choosing `View Feature Form`.

   d) In the feature form the attributes `voltage`, `cables`, `wires`, `circuits`, `frequency` and `name` can be changed. All values must have the same syntax as in OSM.

   e) Close the form and save the changes by clicking on `save layer edits`. The modification will be saved in the database and displayed in the (`vw_change_log` layer. The changes have no direct effect on the `edit_power_relations` object itself.

   Note that adding or deleting members of an existing circuit-relation is not implemented. In this case delete the circuit and insert a new one with the desired members.

5. **Insert a circuit-relation:** A circuit-relations provide information on the wiring of transmission lines. They are constituted of a collection of transmission lines and substations (members of a circuit-relation). If a new circuit-relation is created, first the user must choose its members. Members can be previously existing transmission lines of the layer `power_ways` and/or new transmission lines of the layer `vw_change_log` (that have been digitized as part of a development measure). Since this action can not be performed with the standard QGis tools, a QGis scipt is used instead to insert new circuit-relations. This can be done with the following steps:

   a) Activate the layer `power_ways` and select the desired transmission lines of this layer, using the `Select Features` tool (not the `Identify Feature` tool!). Note that you can select multiple objects by keeping the `command` button pressed. Editing of the layer can remain disabled.

   b) Activate the layer `vw_change_log` (make sure the selected features of `power_ways` remain selected) and also select the desired transmission lines (this step is unnecessary in case you only wish to select power_ways objects or vice versa!)[3].

c) Run the QGis processing script new relation (find script in QGis processing toolbox). This will open a form where you may enter the attributes voltage, cables, wires and frequency of the new circuit-relation. To save the new circuit-relation in the database, click on Run. The changes have no direct effect on the layers power_ways and vw_change_log.

6. **Delete a circuit-relation:** While entering development measures, it is often necessary to delete existing circuit-relations (e.g. to replace them by new ones with different members). This can be done with the following steps:

   a) Enable editing of layer vw_change_log

   b) Select a circuit-relation with the Select Features tool (not the Identify Feature tool!).

   c) Delete the relation by pressing delete on your keyboard

   d) Save the changes by clicking on save layer edits. The deletion of the circuit-relation will be saved in the database and displayed in the (vw_change_log layer. The deletion has no direct effect on the edit_power_relations table itself.

7. **Modify previously entered actions:** Every single action is recorded and saved in the database. These action are displayed in the vw_change_log layer - depending on the current metadata (see section 6.2). Via this layer, the actions can also be modified or deleted (e.g. if the user made a mistake while entering actions). This can be done with the standard QGis layer editing tools or in the layers attribute table. Note that these changes have direct effect on the table and can not be restored!

6.3.2. Types of measures

In this section it is explained how real development measures are translated into **osmTGmod** basic actions.

- **Modification of an existing transmission line:** Some development measures include the modifications of existing transmissions lines. Adding new cables or changing the voltage level of a circuit are examples for such measures.

[3]ways that have already been edited can either be selected from power_ways or from vw_change_log (both reference the same way object).

Transmission lines (OSM ways) are represented by the table `power_ways`. Normally, a transmission line is overlaid by a "circuit" object (represented by the table `edit_power_relations`). If the user wants to change attributes of a transmission line, he must perform actions 4) and 1). **IMPORTANT: In order to modify existing transmission lines, the overlaying relation object as well as all the underlying way objects must be changed.**

- **Construction of New Transmission Lines:** Building new transmission lines is a frequent measure. Entering this measure requires to perform action 2) (insert a new transmission line). Furthermore, it is recommended to digitize a new circuit-relation (action 3)), containing the new transmission line.
- **Deconstruction of Transmission Lines:** This measure involves performing action 3) and 6).

7. Run osmTGmod as Python Module

If the user intends to automate abstractions or other functions of **osmTGmod**, it is recommended to run **osmTGmod** as a python module.

After importing the `osmTGmod` module, a `grid_model` object (instant of the `grid_model` class) can be created. This is done by passing it the required connection parameters (see listing 15). A `grid_model` object provides most of the the above mentioned functions.

```
## My Python program, using osmTGmod as a module ##

# Import osmTGmod as module
import osmTGmod

# Create osmTGmod-object called grid_model
grid_model = osmTGmod.grid_model(database = 'my_database',
                    password = 'my_password',
                    qgis_processing_path = 'my/Qgis/processing/path',
                    host='192.168.0.46',
                    port='5432',
                    user='postgres')

# This will start an abstraction
grid_model.execute_abstraction(None, None, 35176751, 'My very important test
    ')
```

Listing 15: osmTGmod as Python Module

8. Results

After executing an abstraction (see section 5.2.2), the results are saved in the result schema of the **osmTGmod** database. The results can be viewed with QGis or exported as CSV tables. An explanation of the resulting tables and it's columns can be found in X.I.

The system base power (base MVA), used for converting power into per unit quantities, is set to **100 MVA**[4].

8.1. View Results with QGis

In general, it is recommended to work with two independent QGis projects (one for entering development measures, and one for viewing the desired results). After creating a new **osmTGmod** database, these QGis projects are automatically created and can be found in the **osmTGmod** `qgis_projects` folder.

In order to view the results with QGis, perform the following steps:

1. Open the Qgis project-file `your_database_results_project.qgs` (in the **osmTGmod** `qgis_projects` folder).
2. View the results metadata via the **osmTGmod** main menu (see section 5.2.4) to get an overview of the executed abstractions (at least one abstraction must be successfully executed).
3. Run the QGis processing script `show results` (find script in QGis processing toolbox). This will open a form where you may enter the desired result ID. To display this result, press run.

8.2. CSV tables

The data format (column names, units) of the result tables, in the main, corresponds with the *Matpower* input format (Zimmerman, 2014). A few columns have been added for further information on the objects.

[4]The value baseMVA must be specified when calculating load flows with Matpower

8.3. Delete Results from the Database

This functionality is not implemented in Python. In order to delete results, use the database manager *PG-admin* and open the table `results.results_metadata`. This table contains all results metadata. In order to delete a certain result, simply delete the corresponding row in the metadata table. The actual result data is then deleted on cascade.

9. The Abstraction Process

A description of the abstraction process is still not included in this documentation. A detailed description of the corresponding process can be found in the Malte Scharf's bachelor thesis "Entwicklung eines Modells des deutschen Übertragungsnetzes auf Basis der offenen Geodatenbank OpenStreetMap".

References

Bennet, J. (2010). *OpenStreetMap*. Pakt Publishing.

OpenStreetMap-Wiki (2015). Copyright — openstreetmap wiki,. [Online; accessed 3-Dezember-2015, URL: http://wiki.openstreetmap.org/w/index.php?title=Copyrightoldid=1175945]. Available from: `http://wiki.openstreetmap.org/w/index.php?title=Copyright&oldid=1175945`.

Zimmerman, R. D. (2014). *Matpower 5.0 Manual*.

X. Appendix

X.I. Result Tables

Column explanation of the resulting tables.

branch_data	
Column	**Description**
result_id	Database internal ID of the result
view_id	ID for plotting purpose
f_bus	Matpower columns (See Manual p. 108)
t_bus	
br_r	
br_x	
br_b	
rate_a	
rate_b	
rate_c	
tap	
shift	
br_status	
link_type	Link Type (transformer = "transformer", overhead line = "line", underground cable = "cable")
branch_voltage	Voltage of the overhead lines and underground cables
geom	Postgis WGS84 MultiLine geometry

dcline_data	
Column	**Description**
result_id	Database internal ID of the result
view_id	ID for plotting purpose
f_bus	Matpower columns (See Manual p. 110)
t_bus	
br_status	
pf	
pt	
qf	
qt	
vf	
vt	
pmin	
pmax	
qminf	
qmaxf	
qmint	
qmaxt	
loss0	
loss1	
link_type	Link Type (overhead line = "line", underground cable = "cable")
branch_voltage	Voltage of the overhead lines and underground cables
geom	Postgis WGS84 MultiLine geometry

bus_data	
Column	**Description**
result_id	Database internal ID of the result
view_id	ID for plotting purpose
bus_i	Matpower columns (See Manual p. 106)
bus_type	
pd	
qd	
gs	
bs	
bus_area	
vm	
va	
base_kv	
zone	
vmax	
vmin	
osm_substation_id	ID of the substation in OSM (0 = Linking node to other country (see cntr_id), NULL = Node is auxiliary nodes (don't allocate generation or load to these nodes))
cntr_id	Country code of the node (DE = Germany, NL = Netherlands, etc.)
geom	Postgis WGS84 Point geometry

problem_log	
Column	**Description**
result_id	Database internal ID of the result
view_id	ID for plotting purpose
object_type	Object Type ("power_line" = transmission line, "circuit" = circuit, "power_line_as_branch" = abstracted transmission line, "circuit_member" = member of a circuit)
line_id	corresponding OSM line Ids
relation_id	corresponding OSM relation Ids
way	Postgis WGS84 MultiLine geometry
voltage	Voltage of object in Volt (if existent when problem occurs)
cables	Number of conductors of object (if existent when problem occurs)
wires	Type of wires of object (if existent when problem occurs)
frequency	Frequency of object in Hz (if existent when problem occurs)
problem	Description of the problem

substation	
Column	**Description**
result_id	Database internal ID of the result
view_id	ID for plotting purpose
osm_id	OSM ID of the substation object
voltage	voltage in Volt (derived from OSM) of the substation object
s_long	Estimated long term rating of the substation in VA. This value is used to allocate load and generation.
name	name (derived from OSM) of the substation
geom	Postgis WGS84 Line geometry (ground area of the substation)
center_geom	Postgis WGS84 Poin geometry (geometric center of the substation ground area)

plz_subst	
Column	**Description**
result_id	Database internal ID of the result
plz	Postleitzahl
substation_id	OSM ID of the substation object
percentage	PLZ's power allocation share of this substation
distance	Distance form the geometric center of a substation to the corresponding plz-area (NULL = substation within plz area)

nuts3_subst	
Column	**Description**
result_id	Database internal ID of the result
nuts_id	ID of the corresponding NUTS3 area
substation_id	OSM ID of the substation object
percentage	NUTS3's power allocation share of this substation
distance	Distance form the geometric center of a substation to the corresponding NUTS3-area (NULL = substation within NUTS3-area)