SciGRID

Open Source Transmission Network Model

USER GUIDE

V 0.1

AUTHORS:
W. MEDJROUBI & C. MATKE

NEXT ENERGY

EWE-Forschungszentrum

für Energietechnologie e. V.

www.scigrid.de

Contents

1 Licence

SciGRID is free and open-source code based on OpenStreetMap data. The OpenStreetMap data is available under the Open Database License (ODbL) [1] and OpenStreetMap cartography is licensed as CC BY-SA. For more information on the copyright of OpenStreetMap please refer to [2]. All data and databases delivered in the SciGRID model folder are made available under the Open Database License [3]. Any rights in individual contents of the database are licensed under the Database Contents License [4] - See more at [5]. You can also redistribute and/or modify the data distributed with **SciGRID** under the same licenses and copyright.
The **SciGRID** code is licensed under the Apache License, Version 2.0. [6]. For more information concerning the Apache license and a description of the terms under which you can use the **SciGRID** code, please visit the webpage [6].

2 Introduction

2.1 Motivation and aim

Details of electrical transmission networks are currently integrated in a number of in-house energy system models which are not publicly available. The structure, assumptions, simplifications and the degree of abstraction involved in the transmission network models used are, hence, unknown and often undocumented. This fact implies, that the learning curve in the construction of (electrical) grid models is rather slow, since there is hardly any (scientific) discussion on the underlying approaches, procedures and results. At the same time, the output of energy system models takes an important role in the decision making process concerning future sustainable technologies and energy strategies. Recent examples of such strategies are the ones under debate and discussion for the Energiewende in Germany.

In this context, the availability of transmission network data and models is a critical and urgent issue which has to be addressed in order to allow the involvement of an increasing number of actors in the energy sector decision making. Furthermore, transmission network models are important when dealing with issues concerning cross-border congestion management, transmission capacity, grid stability and extension, to cite a few examples.

In this framework, the project **SciGRID** initiated by the research center NEXT ENERGY [7] aims at building an open source model of the electrical transmission network in Europe. The idea behind making both the **SciGRID** model and data available in the open source domain is that the energy modeling sector lacks reliable data on transmission networks. Data available under appropriate (open) licenses ensures that established models and the assumptions they incorporate can be published, discussed and validated in a well-defined and self-consistent manner. In addition, the methods which are developed for building the model will be published under suitable licenses, which is expected to foster and improve existing in-house models.

The main purpose of the **SciGRID** project is hence to open the door to new models and ideas in energy system modeling by providing freely available and well-documented data on the European electrical transmission networks. The open source philosophy offers a high level of transparency as the involved assumptions and simplifications are open to discussion and criticism. Such a discussion and the availability of documented approaches and methods can act as a motivation for more communication and scientific scrutiny in the construction of grid and network models. Furthermore, the **SciGRID** model will also be a benefit for existing in-house models, as its tools and results can be used as both reference implementation and reference data.

2.2 SciGRID project details

Project title:	Open Source Reference Model of European Transmission Networks for Scientific Analysis (Offenes Referenzmodell europäischer Übertragungsnetze für wissenschaftliche Untersuchungen)
Acronym:	SciGRID (Scientific GRID)
Funding period:	September 2014 - August 2017
Sponsoring body:	Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung), Germany
Project partner(s):	NEXT ENERGY (individual project)
Project webpage:	www.scigrid.de

2.3 Technical approach

On the technical level, the **SciGRID** network model is mainly based on the raw transmission data available in openstreetmap.org [8] under the Open Data Commons Open Database License (ODbL) [1]. The ODbL license offers the possibility to share both the model and the ensuing databases.

OpenStreetMap (OSM) data constitutes the skeleton of the **SciGRID** model. After downloading the OSM raw data, it is filtered and exported to relational databases. The relational databases containing the OSM data are then used to build the **SciGRID** transmission network model structure. The relational database approach allows for a flexible and a modular structure of the **SciGRID** model. For example, only networks with transmission lines of a certain voltage level, managed by a certain network operator, or for a certain type of cables can be filtered and graphically displayed. Fig.1 displays the different steps involves in the **SciGRID** model.

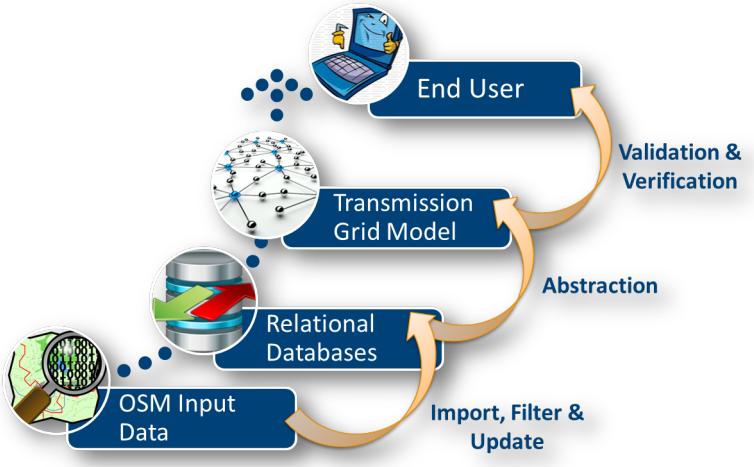


*Figure 1: Schematic representation of the different stages involved in the **SciGRID** model.*

We present here the documentation of the first version of the **SciGRID** transmission network model. This document includes a detailed user guide of the **SciGRID** model as well as the as-

sumptions and simplifications considered in building the model. The **SciGRID** model different components can be downloaded from the **SciGRID** webpage under the download section [9].

2.4 Data sources: OpenStreetMap

The **SciGRID** model is based on data available from the collaborative project OpenStreetMap (OSM). The data extracted from OSM is licensed under the Open Database License (ODbL), which states that also derived databases can be published. This offers the possibility to share the **SciGRID** model along with its output databases.

OSM data is available on the webpage of OpenStreetMap [8] and can be visualized directly as a map. The data can also be downloaded in an XML format, which follows a certain schema definition. OSM data is based on three data types, called "data primitives", which are nodes (defining points in space), ways (defining linear features and area boundaries) and relations (defining logical or geographical relationships between other elements). The geographical locations of the nodes are defined by their latitude and longitude (see Fig.2). Data sets relevant for the power grid are filtered by using the "power" tag, which in OSM identifies a wide range of facilities and features related to the generation, the transmission and the distribution of the electrical power. The relations having the key/value with route/power are filtered out and their members (ways and nodes) are identified and their geographical locations extracted (see Fig.2). The filtering is performed in an automated manner and the different steps it involves will be explained in this user guide. A map of OSM data showing in particular power-related details can be found in the webpage of itoworld [10].

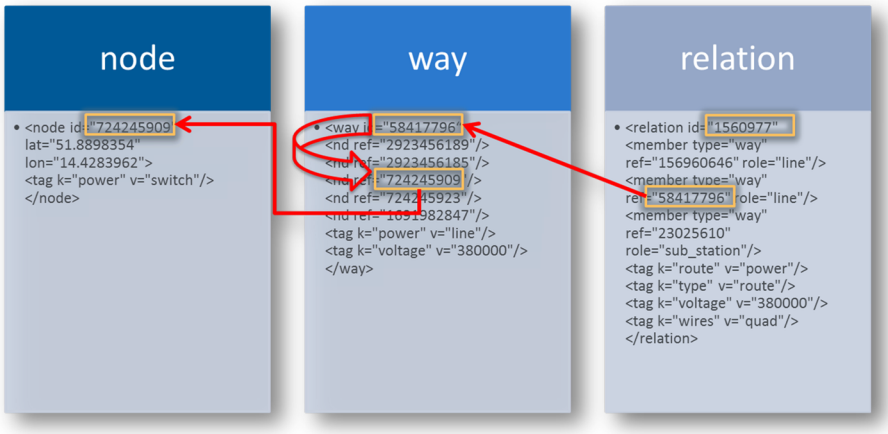


Figure 2: Schematic representation of the OSM data types. Relations with the "power" tag are chosen and also the subsequent ways and nodes belonging to the relations. The identification of relations and their members is done by using their "ID" tag.

OSM data also offers the possibility to isolate the different components of the transmission network by using sub-tags associated with these components. Some practical examples are the tags: "substation" indicating the electrical substations and "line" indicating the transmission lines. Other sub-tags provide technical details about the components of the transmission network as: the tag "voltage" indicates the voltage level(s) of a transmission line or a substation, the tag "cables" indicates the number of power-carrying cables represented by a transmission line, and the tag "tower" to represent the towers or pylons carrying the electricity cables (see Fig.3).

```
<relation id="1637161">
    <member type="way" ref="245128569" role="line"/>
    <member type="way" ref="118942023" role="line"/>
    <member type="way" ref="30181272" role="line"/>
    <member type="way" ref="177829920" role="sub_station"/>
    <member type="way" ref="106684941" role="line"/>
    <member type="way" ref="23837631" role="sub_station"/>
    <tag k="cables" v="3"/>
    <tag k="color" v="white"/>
    <tag k="colour" v="white"/>
    <tag k="frequency" v="50"/>
    <tag k="from" v="Conneforde"/>
    <tag k="operator" v="TenneT"/>
    <tag k="ref" v="304"/>
    <tag k="route" v="power"/>
    <tag k="to" v="Diele"/>
    <tag k="type" v="route"/>
    <tag k="voltage" v="380000"/>
    <tag k="wires" v="double"/>
  </relation>
```

Figure 3: Example of the keys, values and tags available for a power relation in OSM. The relation displayed has the OSM-ID=1637161 [8].

To build the **SciGRID** model using OSM data, the transmission network components need to be accurately and correctly defined, isolated and reproduced in a consistent and an automatized fashion.

3 How to use SciGRID

3.1 Prerequisites

The **SciGRID** model scripts are developed and tested on Linux, but should work with other UNIX systems. The different tools and software used in building the **SciGRID** model and their versions are listed below.

Operating system:	Ubuntu precise (12.04.5 LTS)
Osmosis version:	0.43.1
PostgreSQL version:	9.1.15 (64-bit)
POSTGIS version:	1.5.3
Osm2pgsql version:	0.83
pgAdmin version:	1.14.0
GNU Make version:	3.81
GNU bash:	4.2.25(1)-release
Python:	2.7.3
QGIS version:	2.7.0-Master

For more information on how to install some of the software listed above, refer to Section 6.

3.2 Getting Started

The **SciGRID** network model is based solely on raw OSM data. The latest file containing the OSM data of the whole planet, called "planet-latest.osm" can be downloaded for example from the website [11]. The term "latest" indicates that it is the latest available data of the whole planet. Older versions of the planet data are provided with a "date stamp" and are named as: planet-{date stamp}.osm. Other sources are available for downloading OSM data, for more information refer to [12]. The OSM data can be extracted using Osmosis as smaller data sets , e.g. for a continent, a region, country or a city, depending on the user needs.

The OSM data is available under different formats, we have chosen to work with the .pbf (Protocolbuffer Binary Format) format [13]. The .pbf file representing the OSM planet data contains all information available in the OSM world map. This represents a huge number of nodes, ways and relations (On 20.03.2015 OSM planet data included 2.7 Billion nodes [13]). As the **SciGRID** model deals only with the transmission network, the raw OSM data is filtered with respect to the tag "power" and spatially restricted to the region of interest.

The tag "power" is assumed to represent all data necessary to build the **SciGRID** network model. Power data is represented by relations, ways and nodes having the "power" tag (either as a key or as a value). More precisely, it is assumed that the relations having a tag with the key "route" and value "power" cover all the available information about transmission circuits in OSM. For more information about the tags with the key "power" and how the transmission network is represented in OSM, please refer to the reference [13]. [1]

To filter the data with respect to the region of interest a bounding polygon (a .poly file) is used to extract the desired region data from the whole planet OSM data. A .poly file describes the extent of a region, where parts of the interior region can excluded. As an example, it is possible to extract the data of the state of Lower Saxony in Germany excluding the data for the city of Bremen. For further information about .poly files refer to [14]. Another option for data extraction is to use a bounding box, which describe the extent of a region in terms of a box surrounding it. The usage of the bounding box and .poly files in the **SciGRID** model is described in the following section.

3.3 OSM data download and filtering

The raw OSM data is filtered *thematically* using the "power" tag and *spatially* by only including the region for which the transmission network will be created. As an example, in this user guide the region of interest will be Germany. However, the procedures introduced here can be adapted and applied for any other region, as long as the relation representation of the the transmission network is available. The *Osmosis* tool used to filter the data with respect to the "power" tag will also be introduced.

[1] Note that not all the transmission network details are covered by relations. This is the case for example for most of the European countries, where the relation representation of the transmission network is not available. In Germany however, the relation representation is widely used and covers a high percentage of the transmission network.

OSM data download

The planet OSM data file need to be download as .pbf file to the folder `SciGRID/data/01_osm_raw_data` of the **SciGRID** model folder. In this folder the raw OSM data is stored. The size of the .pbf file is about 28 GigaByte (Status: Mai 2015). Depending on the internet connection speed, the user will need about two hours to download the whole planet OSM .pbf file. It is not necessary that the user downloads the planet OSM data file. In the SciGRID model folder, the resulting data extracted using Osmosis is provided in the folder `SciGRID/data/02_osm_raw_power_data` under the name `de_power_150601.osm.pbf`.

OSM data filtering: the "power" data

The OSM data with the "power" tag for the region "Germany", defined by the coordinates of a bounding box, is extracted using the command line java application *Osmosis*. *Osmosis* is a useful open source application which allows for OSM data manipulation and processing. For more information about *Osmosis* and how to install it, refer to the web-page [15] or the Section 6.1 of this user guide. **Osmosis Version 0.43.1** is used for the present version of the **SciGRID** model.

To filter the OSM data for Germany and for the "power" tag, the raw data is filtered in a first step for the power tag (key=power, value=*) for nodes, ways and relations. This is accomplished by the following osmosis command:

Listing 1: Data filtering power

```
osmosis \
  --read-pbf file='/data/01_osm_raw_data/planet-150601.osm.pbf' \
  --tag-filter accept-relations power=* \
  --tag-filter accept-ways power=* \
  --tag-filter accept-nodes power=* \
  --used-node --buffer \
  --bounding-polygon file='/data/01_osm_raw_data/germany.poly' \
  completeRelations=yes \
  --write-pbf file='/data/02_osm_raw_power_data/de_power_extract1_150601.osm.pbf'
```

note that the symbol \ at the end of the command line is a line-breaking symbol and is not part of the Osmosis command. Note also that the "150601" used in the name of the database and the files throughout this user guide indicates the "date" stamp of the OSM data used. In this case the data used correspond to the 1st of June 2015. As a result of the filtering command in Listing 1, we obtain a "power extract" `de_power_extract1_150601.osm.pbf` with the power tag for nodes, ways and relations.

In a second step, the OSM data is filtered for the relations with the "route=power" tag for Germany, using the following osmosis command:

Listing 2: Data filtering route=power

```
osmosis \
  --read-pbf file='/data/01_osm_raw_data/planet-150601.osm.pbf' \
  --tag-filter accept-relations route=power \
  --used-node --buffer \
  --bounding-polygon file='/data/01_osm_raw_data/germany.poly' \
  completeRelations=yes \
  --write-pbf file='/data/02_osm_raw_power_data/de_power_extract2_150601.osm.pbf'
```

This command results in a second extract `de_power_extract2_150601.osm.pbf` with the relations having the key/value of route/power.

In a third and final step, the previously obtained .pbf files are merged together using the osmosis command:

Listing 3: Data merging

```
osmosis \
  --read-pbf file='/data/02_osm_raw_power_data/de_power_extract1_150601.osm.pbf'\
  --read-pbf file='/data/02_osm_raw_power_data/de_power_extract2_150601.osm.pbf'\
  --merge \
  --write-pbf file='/data/02_osm_raw_power_data/de_power_150601.osm.pbf'
```

The three .pbf files: the two extracts and the merged files are delivered with the SciGRID model in the folder `SciGRID/data/02_osm_raw_power_data`. The .poly file used is in the folder `SciGRID/data/02_osm_raw_power_data` for Germany and is named germany.poly.

In the following is a detailed description about the osmosis commands listed in Listings 1-3:

- `SciGRID/data/01_osm_raw_data` and `SciGRID/data/02_osm_raw_power_data` are the folders where the input file for the OSM planet data and the output of the OSM data filtering are stored, respectively.
- `--read-pbf` enables reading OSM data files in .pbf format. If the file format is .osm the option should be changed to `--read-xml`. However, it is recommended to use .pbf files. It is also possible to use the file formats .osm.gz or .osm.bz2. This is done by unpacking the file using a pipeline: `bzcat planet-latest.osm.bz2 | osmosis file=- ...` .
- `--tag-filter` option is used to filter elements (relations, ways and nodes) based on their type and optionally based on their tag values. It is useful to accept or reject elements that match the filter specification. Note that, one can only specify one filter at a time for nodes (ways and relations) in an osmosis command.
- The combination `--tag-filter accept-relations route=power` is used to include the relations with the key "route" and the value "power".
- To allow the filtering of ways having the key "power", the combination `--tag-filter accept-ways power=*` is used, where * means that any value is accepted. The same syntax is used for nodes and relations, using accept-nodes and accept-relations keywords, as in Listing 1.

- To export nodes which belong only to the filtered ways and relations, the option `--used-node` is used. This option guarantees that other nodes which do not belong to the filtered relations and ways will not be included. This reduces dramatically the number of exported nodes.
- `--buffer` allows the pipeline processing to be split across multiple threads. This is useful if multiple CPUs are available, as multiple tasks consume significant CPU time.
- The data is *spatially* restricted by using a .poly file for Germany. This is accomplished with the option `--bounding-polygon`. The .poly files consist of a list with arbitrary many polygon points, which build the exterior of a region of interest. The .poly file for Germany (germany.poly) is available in the folder `SciGRID/data/01_osm_raw_data` in the **SciGRID** model folder.
- Using `--completeRelations=yes` in combination with the previous bounding polygon option allows for including all available relations which are members of relations which have at least one member in the bounding box. This option implies also that the ways are also completed. This is important in the case where in a relation some relation members (transmission lines, substations) are in Germany but some members are outside of Germany. This is the case for example for cross-border transmission lines (see Figure 4), which extend between two countries. The option `--completeRelations=yes` guaranties that all relation members are exported even if some of them are "physically" outside of the bounding box.
- The option `--write-pbf` allows to write the results of the filtering into the indicated .pbf file.
- Using `--merge` combines two *Osmosis* extracts to one file.

After executing the *Osmosis* command in "Listing 1,2 and 3", the resulting data file containing the filtered "power" OSM data is labelled `de_all_power_150601.osm.pbf` and is stored in the `SciGRID/data/02_osm_raw_power_data` folder. A copy of this file is provided with the **SciGRID** model folder. The data set includes only relations, ways, and nodes having the key "power" for the region Germany. This data will be labelled "power data" throughout this user guide. Note that the label "latest" in the name of the "power" data file can be changed to indicate the "date stamp" of the original raw OSM data. This will help the user keep track of which raw OSM data file was used for extracting data.

For more information about other options available in *Osmosis*, please refer to the OSM wiki webpage [13] or type the command `osmosis --help` in a shell terminal.

Using bounding box for OSM data filtering

When filtering for a certain region one can also use a bounding box, which is less precise than a .poly file in defining borders.This is done by using the osmosis option –boundig-box instead of –bounding-polygon. The user needs to indicate the latitudes of the top, bottom bounding box, and the longitudes of the left and right edges of the bounding box; respectively. The user can adapt the bounding box coordinates to other regions by changing the coordinates. For Germany for example, the osmosis command using a bounding box will be `--bounding-box top=56 bottom=46 left=5 right=16`.

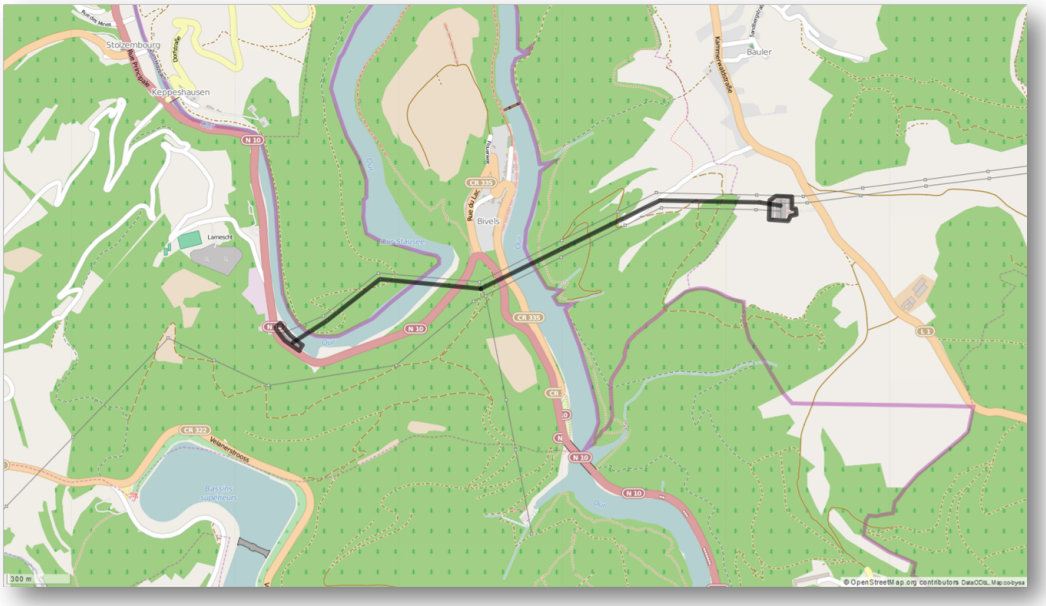


Figure 4: Example of a relation representing a cross-border transmission line. One transmission substation and the transmission lines of the relation are in Germany (right side of the figure) and the second transmission substation is in Luxembourg (left side of the figure).

3.4 Power data export to the database

To store and manipulate the "power data", it needs to be exported to a database. This is accomplished by using the command-line based program *Osm2pgsql* [13], which converts OSM data to *PostGIS*-enabled *PostgreSQL* databases. The open source *PostgreSQL* [16] is an object-relational database management system. *PostGIS* [17] is a spatial database extender for *PostgreSQL* object-relational databases. *PostGIS* is very useful in the context of OSM data as it enables support for geographic objects allowing location queries to be run in *PostgreSQL*.

What is *Osm2pgsql* actually doing? First, it is provided with a so called .style file, where all the settings of how *Osm2pgsql* deals with the OSM raw data are stored. These settings indicate which columns are created for the tables containing the "power data" in the *PostgreSQL* database. It is also possible to define which information in the "power data" are ignored and will not be stored in the database. In the context of **SciGRID**, it is important to use a customized .style file, and not the default one available in the *Osm2pgsql* folder because the default .style file does not consider the "power" related tags when exporting OSM data to the database. A customised .style file called power.style is provided with the **SciGRID** model in the folder `SciGRID/data/02_osm_raw_power_data`.

In the following, the different steps involved in exporting the "power data" to the *PostGIS* enabled *PostgreSQL* database are described in details. Note that, the following steps represent step 1 to 3 of the makefile and the bash script used to run the **SciGRID** model and are provided in the folder `/code`.

- Step1: Create a *PostGIS* template database using the command line *PostgreSQL* tool *psql* which is hstore enabled. Four commands are necessary to create the *PostGIS* template.
- Create a *PostGIS* database named `scigrid_template`, which enables the PL/pgSQL

language for all databases. This is accomplished by the command:

```
createdb -U postgres -h localhost plpgsql scigrid_template
```

Note that this step is a necessary one as many of the *PostGIS* functions are written in the PL/pgSQL procedural language. The option `-U postgres` indicates the username of the *PostgreSQL* system, the default username is "postgres". The option `-h localhost` indicates the *PostgreSQL* server host or socket directory. The default value for the localhost is 127.0.0.1. To enable the PL/pgSQL language the keyword `plpgsql` is used, and finally the name of the *PostGIS* database to be created is indicated as `scigrid_template`.

- Load the *PostGIS* object and function definitions into the database by loading the postgis.sql definitions file (located by default in the folder [prefix]/share/contrib). Change the location of this file, if different, in the makefile or the bash script provided with the **SciGRID** model.

```
psql -d scigrid_template -U postgres -h localhost
-f /usr/share/postgresql/9.1/contrib/postgis-1.5/postgis.sql
```

The option `-f ../../postgis.sql` indicates that a file containing SQL commands will be executed, in this case the file postgis.sql.

- Install the spatial reference system for *PostGIS* which is necessary for a complete set of EPSG coordinate system definition identifiers. The `spatial_ref_sys`.sql definitions file has to be loaded and will populate the `spatial_ref_sys` table. This is done with the command:

```
psql -d scigrid_template -U postgres -h localhost
-f /usr/share/postgresql/9.1/contrib/postgis-1.5/spatial_ref_sys.sql
```

This will permit the use of very useful *PostGIS* functions, for example the `ST_Transform()` operations on geometries. For more information about the *psql* command line and the available options, consult the *psql* section in the *PostgreSQL* documentation [16].

- Create the "hstore" extension for postgis database `scigrid_template`. The "hstore" data type permits storing sets of key/value pairs within a single *PostgreSQL* value. This can be useful in various scenarios, such as rows with many attributes that are rarely examined, or semi-structured data. Keys and values are simply text strings. This is accomplished by the command:

```
psql -d scigrid_template -U postgres -q -h localhost
-c "CREATE EXTENSION hstore;"
```

- Step2: Create an empty database with the name `de_power_150601`, using the template created in the previous step. The user may change the name of the database accordingly.

The empty database will hold the "power data" (de_all_power_150601.osm.pbf) filtered form the OSM data in Section 3.3.

```
psql -U postgres -d scigrid_template -h localhost -c "CREATE DATABASE
de_power_database WITH TEMPLATE = scigrid_template;"
```

The option `-c SQL_command` specifies that *psql* is to execute one command string, SQL_command, and then exit.

- Step3: Export the data in de_all_power_150601.osm.pbf to the database de_power_150601 using *Osm2pgsql*. This is accomplished by using the following command:

Listing 4: Data export

```
osm2pgsql -r pbf \
-U postgres -H localhost -P 5432 -d de_all_power_150601  \
-S /data/02_osm_raw_power_data/power.style -k -s  \
-C cash_size_in_MB  \
--number-processes nb-processors \
/data/02_osm_raw_power_data/de_all_power_150601.osm.pbf
```

  - The first option `-r pbf` indicates the input reader format, in this case the OSM binary format .pbf is used. The path and the name of the power data file to be exported is indicated on the last line of the command as: `SciGRID/data/02_osm_raw_power_data/de_all_power_150601.osm.pbf` . This file is provided in the folder SciGRID/data/02_osm_raw_power_data of the **SciGRID** model.
  - The server hostname (or socket location) is indicated by using `-H localhost` . The default value of the server hostname is 127.0.0.1. The server port is defined using `-P 5432` , where 5432 is the default port number, which can be changed by the user when installing *PostgreSQL*.
  - The database to which the data is exported is defined using `-d de_power_150601` , which in this case is de_power_150601, where "de" stands for Germany (or Deutschland).
  - The option `-S /data/02_osm_raw_power_data/power.style` indicates the location and name of the style file. If not set by the user, the default .style file location is: `/usr/local/share/osm2pgsql/default.style` . Please note that, to be able to export the power ways extracted with *Osmosis* (Section 3.3) the default style file provided need to be changed. In **SciGRID** an appropriate .style file under the name power.style is provided in the folder `SciGRID/data/02_osm_raw_power_data` and called power.style.
  - `-k` option enables adding tags which are not stored into columns, to an additional hstore (key/value) column in the *PostgreSQL* databases. The tags to be added as columns are indicated, as stated above, in the power.style file.

- To reduce the RAM usage it is useful to use the "slim mode" indicated by `-s` in the export command. The data is then stored temporary in the database, although the data export is slower.
  An important advantage of using the slim mode option is that the relations will be exported to the database as well, which is not the case when using the default export mode. As the relations constitute the backbone of **SciGRID** model it is mandatory to use the slim mode option when exporting the "power data" to the database.
- When using the slim mode, the option `-C cache_size_in_MB` is mandatory. The default cache size is 800 MB of RAM.
- The user can specify the number of processors to run the data export in parallel by making use of the option `-number-processes nb_processors`, where `nb_processors` is the number of processors available.

After executing the command in Listing 4 , the "power data" of Germany is exported into the PostgreSQL database `de_power_150601`, created in Step 2. The `de_power_150601` is extended with *PostGIS* and hstore extensions.

3.5 Abstraction

After downloading, filtering and exporting the OSM "power data" for Germany into the database `de_power_150601`, the input data necessary for the **SciGRID** network model is now available.

The **SciGRID** network model is based on the "power" relations, i.e. relations with a key/value pair route/power. Relations with the key "route" and value "power" are typically constituted of one or more substations and one or more transmission lines. The relations considered in this first version of the **SciGRID** model are:

- Relation with only two substations and one/several transmission lines linking them, an example of such relations is shown on Fig. 5. Substations are defined in OSM by the key "power" and the values "substation", "sub_station", "station", "plant", "generator". Transmission lines are on the other side defined by the key "power" and the values "line" and "cable".

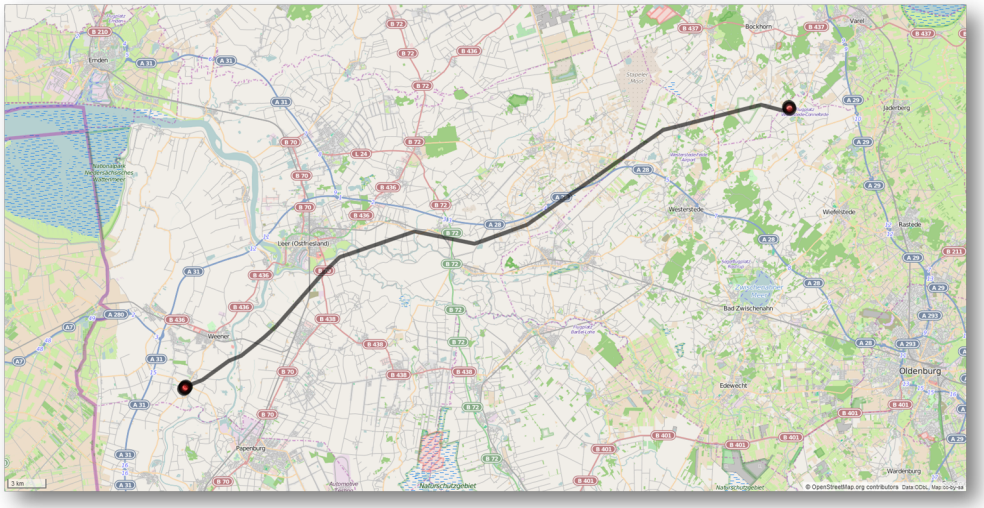


Figure 5: Example of a relation constituted of two substations. The two red circles are the two substations contained in the relation (OSM-ID=1637161) and the black line are the transmission lines (this relation contains four transmission lines). Figure obtained using overpass-turbo.eu.

- Relations with three substations and with a T-junction. T-junctions are connections where transmission lines branch, an example of a relation with 3 substations and a T-junction is shown in Fig. 6. The nodes at which the lines branch are called T-nodes, an example is shown on Fig. 7.

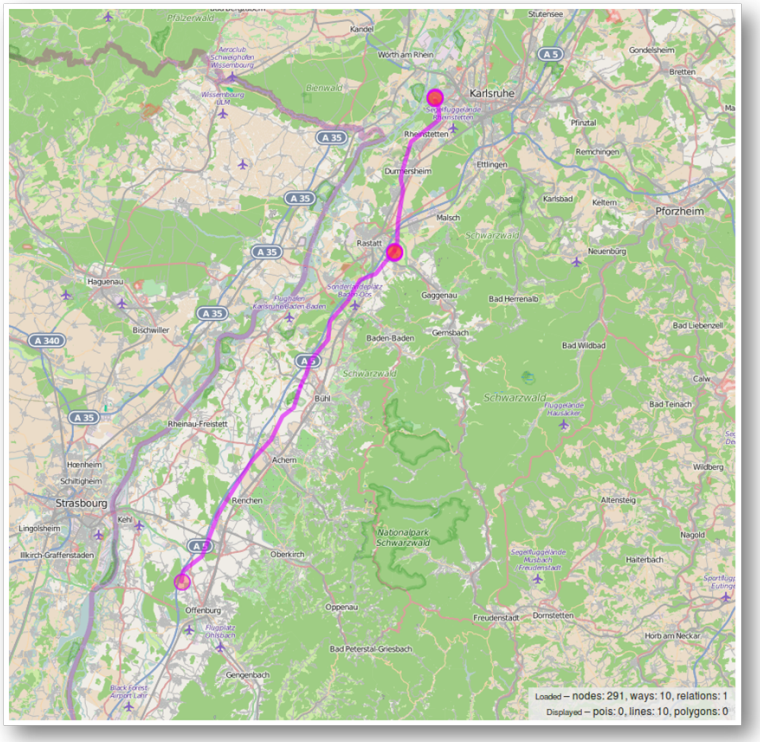


Figure 6: Example of a relation (OSM-ID=339244) containing three substations (circles), a T-junction and seven transmission lines. Figure obtained using overpass-turbo.eu.

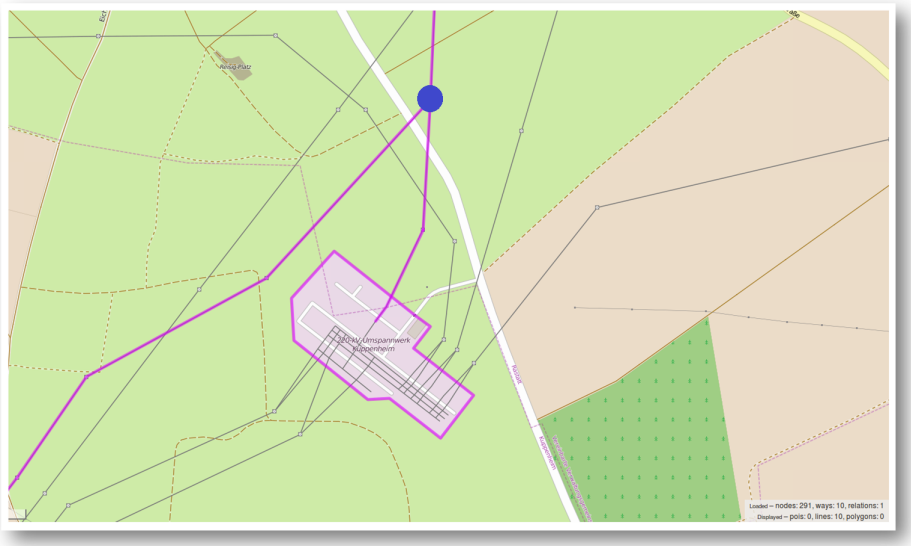


Figure 7: T-connection from relation (OSM-ID=339244). The T-node represented by the blue filled circle, is the node at which the transmission line at the top part of the figure branch into two parts. Figure obtained using overpass-turbo.eu.

These simplifications are used in this first version of the **SciGRID** model as it is quite straightforward to extract the network model when only considering such relations. Relations which have more than three substations are not considered in a first approximation due to the difficulty of calculating the transmission lines length. This is due to the fact that the relation members (nodes and ways) may not belong to each possible connection between two substations. This makes it difficult to define which transmission lines are connected to which substation, so that the length of the transmission lines is not possible to calculate without the use of an elaborated algorithm or a routing routine. Relations with zero or one substation are also not considered in the **SciGRID** model as they constitute incomplete electrical circuits. In **SciGRID** only circuits with two substations and one or more transmission lines are considered.

The abstraction involves extracting the vertices (nodes) and links (edges) of the transmission network without considering the path followed by the transmission lines. The vertices of the network are represented by the geographical center positions of the substations which are members of the "power" relations. This procedure guarantees that possibly multiple relations with a shared substation will be abstracted with the same vertex, since the geometric polygon of the substation will be abstracted to its geometric center. The links (or edges) of the network are represented by the transmission lines as straight lines, without including the information about their paths. Therefore, the network produced by the **SciGRID** model is said to be an "*abstracted*" transmission network, as it does not reproduce the transmission lines actual paths. It is however straightforward to conserve the information about the topology of the transmission lines if needed.
The abstraction step in **SciGRID** will produce the vertices and the links of the abstracted transmission grid, which can be stored as .csv files. The abstraction is divided in several sub-steps and is executed by running the python script SciGRID.py. The sub-steps involved in the abstraction script SciGRID.py are each accomplished by calling a function in SciGRID.py. In the following is a listing of the different steps involved in the abstraction process. The abstraction procedure is going to be presented first for relations with two substations (Section 3.5.1) then

for relations with 3 substations and a T-junction (Section 3.5.2).

3.5.1 Relations with 2 substations

The abstraction is performed using a python script called SciGRID.py located in the code directory provided with the **SciGRID** model. In the following, the different steps involved in the abstraction are introduced.

Abstraction step 1: "power" relations analysis

To be able to abstract relations with two substations, we need to perform an analysis of the available "power" related relations we extracted for Germany, and which are contained in the database `de_power_150601`. The analysis will concern the number of the substations (and also transmission lines) contained in each "power" relation. Note that, only relations where the voltage tag has a value of 220kV and higher are considered, as in **SciGRID** only high and very high voltage transmission systems are modelled.
The analysis of the relation is performed by the function `relation_analysis.py` called by the SciGRID.py script. Several *SQL* functions are defined in `relation_analysis.py` to list all relations in the database `de_power_150601` and check for different attributes. The relations have a tag "parts", where all parts of a relation (including relations, ways, and nodes) are listed with their respective IDs. The relations "parts" or elements usually consists of transmission lines and substations. The relation "parts" can also have extra tags other than the "power" tag. If a part of a relation has the tags "power=construction", "power=planned" or "power=fixme" or has no power tag, it is listed as a "discarded" part and the relation is excluded from the **SciGRID** model. The number of substations and transmission lines involved in each relation is listed and stored in the data table `_analysis_rels`. The functions defined in `relation_analysis.py` use the relation ID as a variable (input) value and each function call analyses the relation in terms of:

- the relation electrical properties (voltage, cables, wires, frequency)
- the (relation,way,node) IDs of substation parts
- the (relation,way,node) IDs of transmission line parts
- the (relation,way,node) IDs of discarded parts
- number of substation parts
- number of transmission line parts
- number of discarded parts
- the total number of parts
- an analysis for T-junctions IDs

An example output of the relations analysis for Germany is shown in Table 1 and stored in the table `_analysis_rels`.

relations to be fixed / being planned / being under construction	52
relations with 0 substation	4
relations with 1 substation	23
relations with 2 substations	643
relations with 3 substations	67
relations with 4 substations	3
total number of "power" relations in Germany	792

Table 1: Number of relations with key/value pair route/power in the data set of Germany in OSM (Status: 18.05.2015). The total number of relations is subdivided into sets of relations with different numbers of substations or categorized as discarded.

Abstraction step 2: creating the vertices table

As stated earlier in this section, relations containing only 2 substations can be trivially connected. Using these relations, the polygons of the substations are abstracted to their geometric center. The centres of all substations will then build the vertices database of the abstracted **SciGRID** network model. As an example, see Fig.8, where the relation with OSM-ID:3756858 is abstracted. The geometric centres of the two substation in orange color are the vertices of the network model.

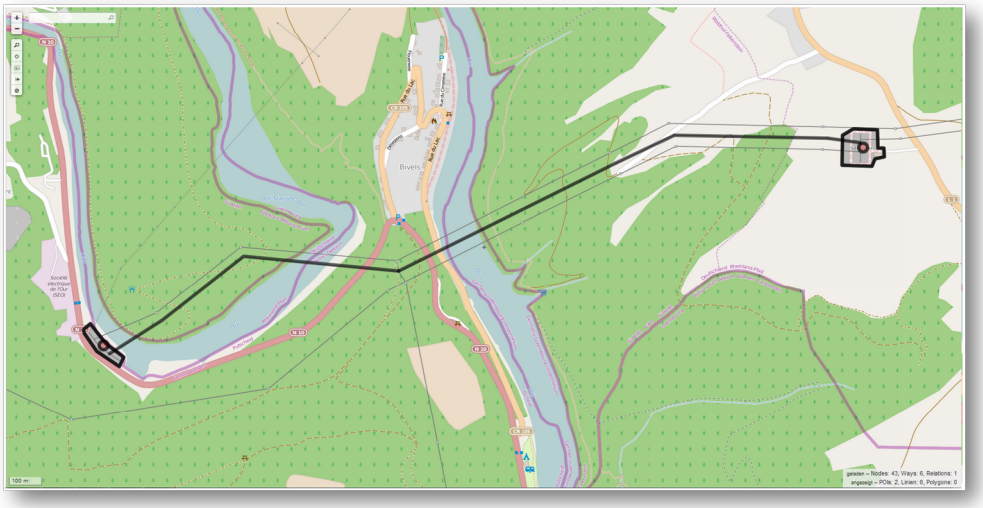


Figure 8: Abstraction of the substations in the relation OSM-ID:3756858. The centres of the substations in orange represent the network vertices, and the lines in black are the original transmission lines. Figure obtained using overpass-turbo.eu.

After analysing the relations as explained previously, three tables are needed to obtain the list of vertices derived from relations with exactly 2 substations. They are created using the following three *SQL* commands, which are part of the script `db_create_tables.py`:

Listing 5: Intermediate vertices table

```
CREATE TABLE _vertices (
    id              serial PRIMARY KEY NOT NULL,
    osm_id          bigint,
    osm_id_typ      char,
    geo_center      geometry,
    longitude       float,
    latitude        float,
    role            text,
    voltage         text,
    from_relation   bigint);
```

Listing 6: Intermediate table of abstracted substations IDs

```
CREATE TABLE vertices_ref_id (
    v_id            serial PRIMARY KEY NOT NULL,
    osm_id          bigint,
    osm_id_typ      char,
    visible         smallint);
```

Listing 7: Table of abstracted substations

```
CREATE TABLE vertices (
    v_id            bigint PRIMARY KEY,
    lon             float,
    lat             float,
    typ             text,
    voltage         text,
    geom            geometry);
```

The script `db_create_tables.py`, called by the SciGRID.py script creates all tables needed for the abstraction process. The table `_vertices` is an intermediate list of vertices, which also contains the relation's ID, given by the table `_analysis_rels` from Step 3.5.1. Since different relations can share the same substation (vertex), the table `_vertices` can contain repeated vertices. Therefore an intermediate step is necessary to "clean" these repeated vertices so that the table `vertices` contains non-repeated "unique" vertices. A list of unique vertices is obtained by adding vertices only once to the table `vertices`. These vertices get a new ID `v_id`, which identifies them within the **SciGRID** model. This new `v_id` is necessary, since vertices can be derived from different data types in OSM which may share the same `osm_id`. The IDs are only unique within a data type, namely nodes, ways, or relations. This were the table `vertices_ref_id` comes into play as it links the original OSM ID `osm_id` and the unique **SciGRID** `v_id`.

Abstraction step 3: creating the links table

Similarly to obtaining a table of vertices, two tables are needed to abstract the table of links derived from relations with exactly 2 substations. These tables are created using the following two *SQL* commands:

Listing 8: Intermediate links table

```
CREATE TABLE _links (
    id                  serial PRIMARY KEY NOT NULL,
    osm_id_1            bigint,
    osm_id_1_typ        char,
    osm_id_2            bigint,
    osm_id_2_typ        char,
    length_m            integer,
    voltage             integer,
    cables              integer,
    wires               text,
    wires_nb            integer,
    frequency           text,
    from_relation       bigint,
    from_transmissions  bigint[]);
```

Listing 9: Table of abstracted connections between substations

```
CREATE TABLE links (
    l_id                serial PRIMARY KEY NOT NULL,
    v_id_1              bigint,
    v_id_2              bigint,
    voltage             integer,
    cables              integer,
    wires               integer,
    frequency           text,
    length_m            integer,
    r                   float,
    x                   float,
    c                   float,
    i_th_max            float,
    geom                geometry);
```

These two commands are also part of the script `db_create_tables.py`. The relation IDs are given by the table `_analysis_rels` from Step 3.5.1. For each relation, a function which abstracts the connections between two substations to a link is applied. The link is defines as a straight line connecting two abstracted substations (see Fig.9). This step is accomplished by the function `abstract_rel_with_2subs` included in the python script `relation_abstraction.py` and called by the SciGRID.py script. Each link properties, including its length, the number of wires and cables and the frequency are added to the table `_links`. The table `_links` is an

intermediate table, since the begin and end of a link need to have a unique ID. Therefore an intermediate step is necessary to "clean" these repeated vertices so that the table `links` contains non-repeated "unique" links, as it was the case for the vertices table. A list of unique links is again obtained by adding vertices only once to the table `links`. These links are assigned a new ID `v_id` identifying links in the **SciGRID** model. The table `links` uses the referenced `v_id` obtained from table `vertices_ref_id`.

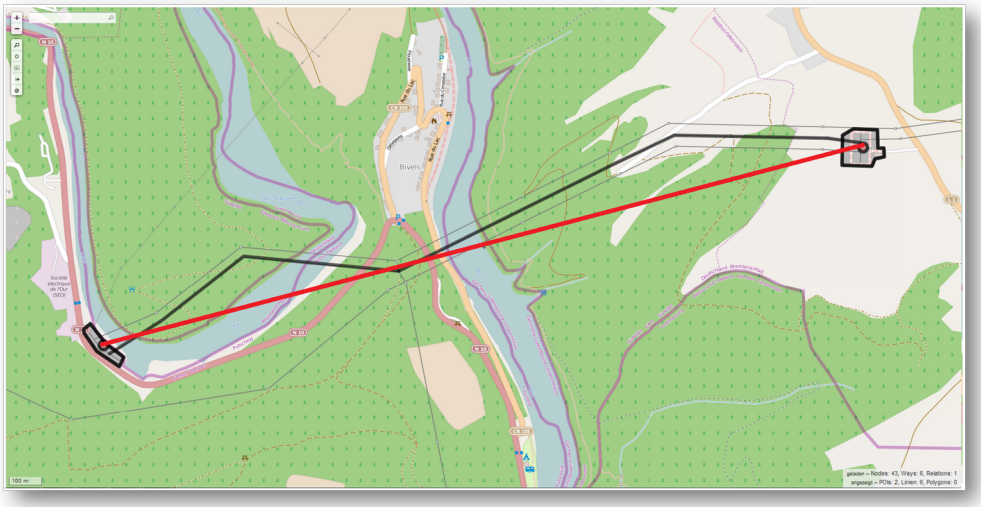


*Figure 9: Abstraction of the relation OSM-ID:3756858: the path followed by the transmission lines (in black) is not considered in the **SciGRID** abstracted model and the network link is represented by a straight line instead (in red). Figure edited using overpass-turbo.eu.*

3.5.2 Relations with 3 substations and a T-junction

As mentioned earlier, T-junctions are located where transmission lines branch (see Fig. 6 and Fig. 7). In a first approximation, only relations with 3 substations and a T-junction are going to be abstracted in this version of the **SciGRID** model. As for the relations with exactly 2 substations previously introduced, the abstraction of relations with three substations and a T-junction is accomplished by using the SciGRID.py script. For simplicity, these relations are labelled "T-junction relations" in the remainder of this user guide.

The abstraction of T-junction relations follows the same sub-steps introduced in Section 3.5.1, which are: extracting the desired relations from the relation analysis results, abstracting the vertices and links and exporting the results to the *vertices* and *links* tables.

After running the analysis script `relation_analysis.py`, the relations with 3 substations are sorted out and their IDs identified. This is achieved in the python script `relation_abstraction`, (called by the SciGRID.py main script) by the function `abstract_rel_with_T_node`.

Not all relations with 3 substations contain a T-junction. In order to find T-junctions in relations, the relations with three or more substations are analysed. First, a list of all nodes is selected of all relation parts associated with transmission lines. In this nodes list, the nodes are counted. Since nodes within one associated part of transmission line occurs only once, the nodes which appear 3 times in the list of nodes over all associated part of transmission lines belong to three

different transmission line segments. Thus, these three segments share the same node which is then identified as T-node.

Since, relations with 3 substations and more than 3 cables can have more than one T-node, one must also check if a connection from each of the three segments start at the T-node and end in a different substation. This is achieved by by the function `abstract_3subs_T_node` in `relation_abstraction`. The relations for which a T-nodes exists are then abstracted by using the T-node as an "auxiliary vertex" linking the three substations present in the relation (see Fig. 10 and Fig. 11). The relations with 3 substations which have no T-junction are not included in the present **SciGRID** version.

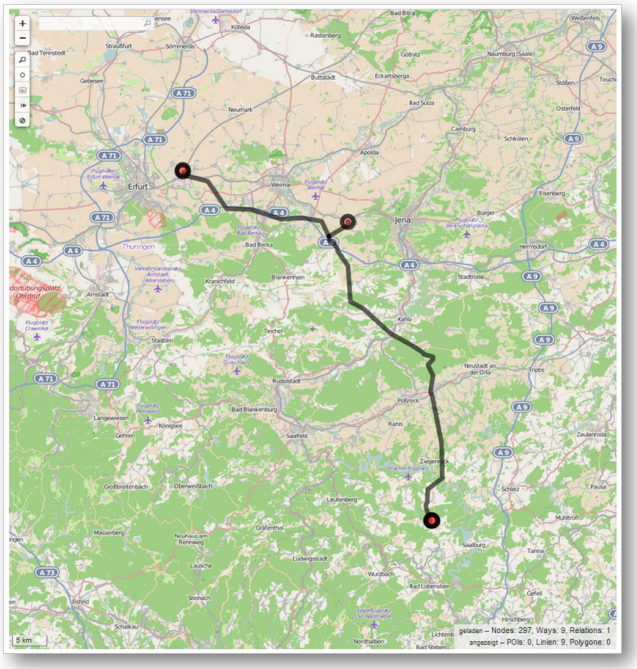


Figure 10: Example of power relation (OSM-ID:918569) with 3 substations and a T-junction. The transmission lines are in black and the stations geometrical centres in orange. Figure obtained using overpass-turbo.eu.

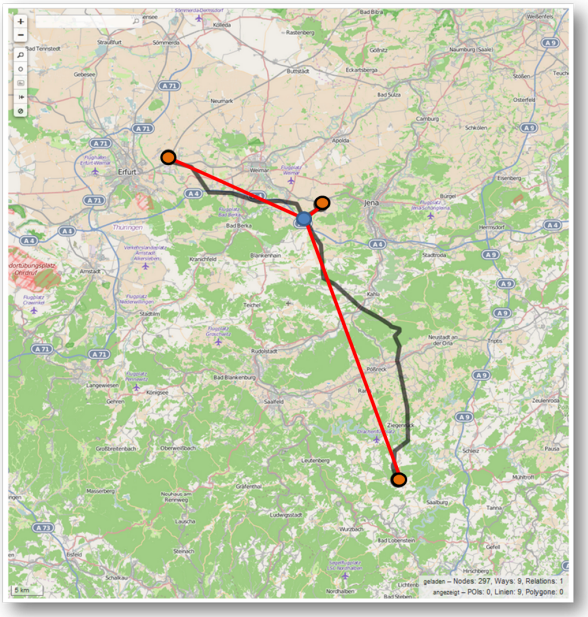


Figure 11: Example of the abstraction of a T-junction relation (OSM-ID:918569) with 3 substations and a T-junction. The transmission lines in black represent the real path and the red lines the abstracted lines. They rely the T-node in blue, which is used as an auxiliary vertex, and the 3 substations vertices in orange.

To define which transmission lines lies between the 4 abstracted vertices (3 as the substation nodes and the T-junction node), the function `separate_parts` is used by the abstraction script `relation_abstraction`. Then the function `insert_segments` is used to insert and update the tables `vertices` and `links` for the vertices and links obtained.

Vertices and links tables: The SciGRID model output

After executing the abstraction process described above, two new tables are added to the `de_power_150601` database. First, the tables `vertices`, defined in Listing 7, contains the vertices of the transmission network. Second, the table `links`, defined in Listing 9, which contains the links of the transmission network.

Additionally, in the links table the electrical properties of the transmission lines are calculated by the electrical_properties.py which is called by SciGRID.py. The electrical properties calculated are: resistance r, reactance x, capacitance c, and maximum current-carrying capacity $I_{\max}$ and they inserted in the links table as columns. r, x, c and $I_{\max}$ depend on: the length, the voltage level, the number of wires, and the number of cables represented in a link. The equation for the specific resistance is given as:

$$r = C_r \cdot \frac{l}{s \cdot w},$$

where r is the specific resistance in [ohm/km], l the transmission line length, C_r a coefficient which depends on the voltage level, s the number of systems in a link which is the number of cables divided by 3 and w the number of wires in a cable. For the reactance and the capacitance

equation 3.5.2 cannot be applied. The calculation of these two values is done by multiplying the coefficients C_x and C_c from reference [18] by the length of the link. Finally, the maximum current-carrying capacity is calculated with the equation:

$$I_{\max} = C_i \cdot l \cdot s \cdot w,$$

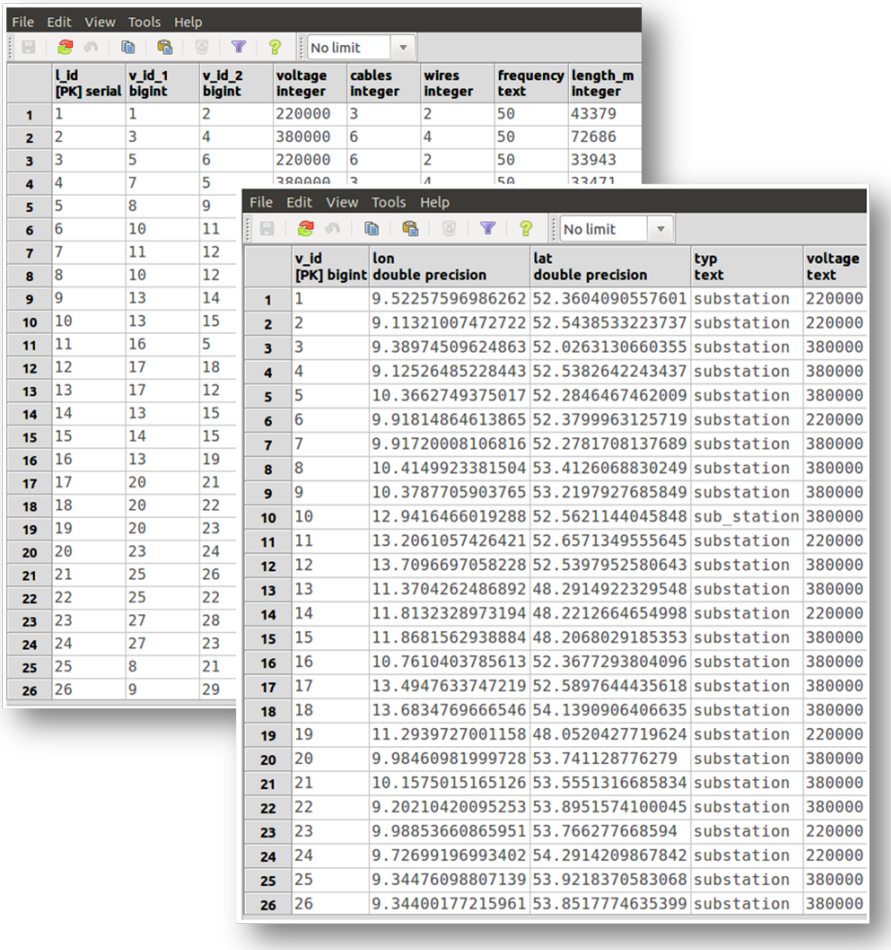
The information about cables and wires is available in OSM but not for all power relations. For the relations where no wires and cables information is available the electrical properties are not calculated.
There exist several references for the values of C_r, C_x, C_c and C_i. The ones used in **SciGRID** are based on the data provided by the German Energy Agency (dena) [19] in their annual report of 2012 [18].

Table 2 lists the values used for C_r, C_x, C_c and C_i obtained in [18] and used in this version of the **SciGRID** model. The user can however use his/her own values by adapting the `electrical_properties.py` file in the /code folder.

Voltage level	C_r	C_x	C_c	C_i
380 kV	0.025	0.25	0.0137	2.6
220 kV	0.080	0.32	0.0115	1.3

Table 2: Electrical properties coefficients from reference [18].

The *vertices* and *links* tables are part of a "relational database", i.e. their rows have a unique key. Because each row in a relational database has its own unique key, rows in other tables that are related to it can be linked to it by storing the original row's unique key as an attribute of the secondary row. Fig. 12 displays an example of the tables vertices and links obtained when executing the abstraction script SciGRID.py.



File Edit View Tools Help

No limit

	l_id [PK] serial	v_id_1 bigint	v_id_2 bigint	voltage integer	cables integer	wires integer	frequency text	length_m integer
1	1	1	2	220000	3	2	50	43379
2	2	3	4	380000	6	4	50	72686
3	3	5	6	220000	6	2	50	33943
4	4	7	5	380000	3	4	50	33471
5	5	8	9					
6	6	10	11					
7	7	11	12					
8	8	10	12					
9	9	13	14					
10	10	13	15					
11	11	16	5					
12	12	17	18					
13	13	17	12					
14	14	13	15					
15	15	14	15					
16	16	13	19					
17	17	20	21					
18	18	20	22					
19	19	20	23					
20	20	23	24					
21	21	25	26					
22	22	25	22					
23	23	27	28					
24	24	27	23					
25	25	8	21					
26	26	9	29					

File Edit View Tools Help

No limit

	v_id [PK] bigint	lon double precision	lat double precision	typ text	voltage text
1	1	9.52257596986262	52.3604090557601	substation	220000
2	2	9.11321007472722	52.5438533223737	substation	220000
3	3	9.38974509624863	52.0263130660355	substation	380000
4	4	9.12526485228443	52.5382642243437	substation	380000
5	5	10.3662749375017	52.2846467462009	substation	380000
6	6	9.91814864613865	52.3799963125719	substation	220000
7	7	9.91720008106816	52.2781708137689	substation	380000
8	8	10.4149923381504	53.4126068830249	substation	380000
9	9	10.3787705903765	53.2197927685849	substation	380000
10	10	12.9416466019288	52.5621144045848	sub_station	380000
11	11	13.2061057426421	52.6571349555645	substation	220000
12	12	13.7096697058228	52.5397952580643	substation	380000
13	13	11.3704262486892	48.2914922329548	substation	380000
14	14	11.8132328973194	48.2212664654998	substation	220000
15	15	11.8681562938884	48.2068029185353	substation	380000
16	16	10.7610403785613	52.3677293804096	substation	380000
17	17	13.4947633747219	52.5897644435618	substation	380000
18	18	13.6834769666546	54.1390906406635	substation	380000
19	19	11.2939727001158	48.0520427719624	substation	220000
20	20	9.98460981999728	53.741128776279	substation	380000
21	21	10.1575015165126	53.5551316685834	substation	380000
22	22	9.20210420095253	53.8951574100045	substation	380000
23	23	9.98853660865951	53.766277668594	substation	220000
24	24	9.72699196993402	54.2914209867842	substation	220000
25	25	9.34476098807139	53.9218370583068	substation	380000
26	26	9.34400177215961	53.8517774635399	substation	380000

*Figure 12: Example of the **SciGRID** model network vertices and links tables obtained using the abstraction command.*

The data obtained when running the abstraction script constitutes the output of the **SciGRID** model. The table of the vertices and links of the transmission network (only with relations containing 2 substation and T-junction relations, for the moment) can be used and/or edited as a table or a .csv file. The .csv files of the vertices and links are both provided with the **SciGRID** model in the folder `SciGRID/data/03_network`. They are created by the function `create_csv_files` included in the SciGRID.py script.

The user can adapt the previous steps, of download, filtering and abstraction for any region or any relations tag other than "power". The same apply for the open source tools used. The next step in **SciGRID** model development will be to also include the rest of the relations, which contain more than three substations.

3.6 Visualization

There are different ways to visualize the **SciGRID** abstracted network model. When running the SciGRID.py abstraction script, the last step of the abstraction is to create a plot of the abstracted network. This is done by the function `create_plots.py`. The plot is stored as a .pdf file in the folder `SciGRID/data/04_visualization` provided with the **SciGRID** model. An example is

shown in Fig. 13.

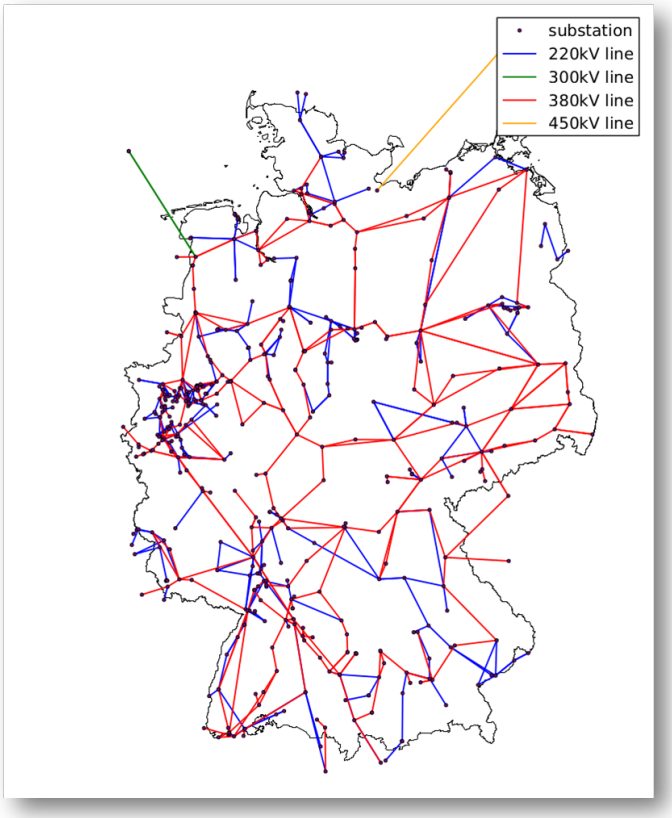


*Figure 13: Example of the plotted **SciGRID** abstracted model network using the plot function available in SciGRID.py.*

Another option to visualize the resulting **SciGRID** abstracted model is to use the QGIS application [20], which is a free and open-source desktop Geographic Information System (GIS) application providing data viewing, editing, and analysis capabilities for GIS enabled tables and databases under different formats. For more information about QGIS and how to install it, refer to [20].

In the following, the connection to the database containing the "power" data as well as the `_vertices` and `_links` tables of the **SciGRID** model using QGIS is introduced.

After installing QGIS, load/open it by typing the command `./qgis` in a shell terminal. A connection to the *PostgreSQL* database `de_power_150601` needs to be established, therefore go to the Browser (left in the GUI) under *PostGIS*, see Fig. 14). Click on the right mouse button, then click on “New Connection” (see Fig. 15) to establish a database connection.

A dialogue window appears, where you have to add a name for the connection and the necessary connection parameters to the PostgreSQL database (Host, Database, Port, Username, and Password), see Fig. 16. When this is done, click on the “Test Connection” button and you should see a message saying “Connection to databasename was successful”. Once this is successful, click OK. Now a connection between QGIS and the `de_power_150601` is established.

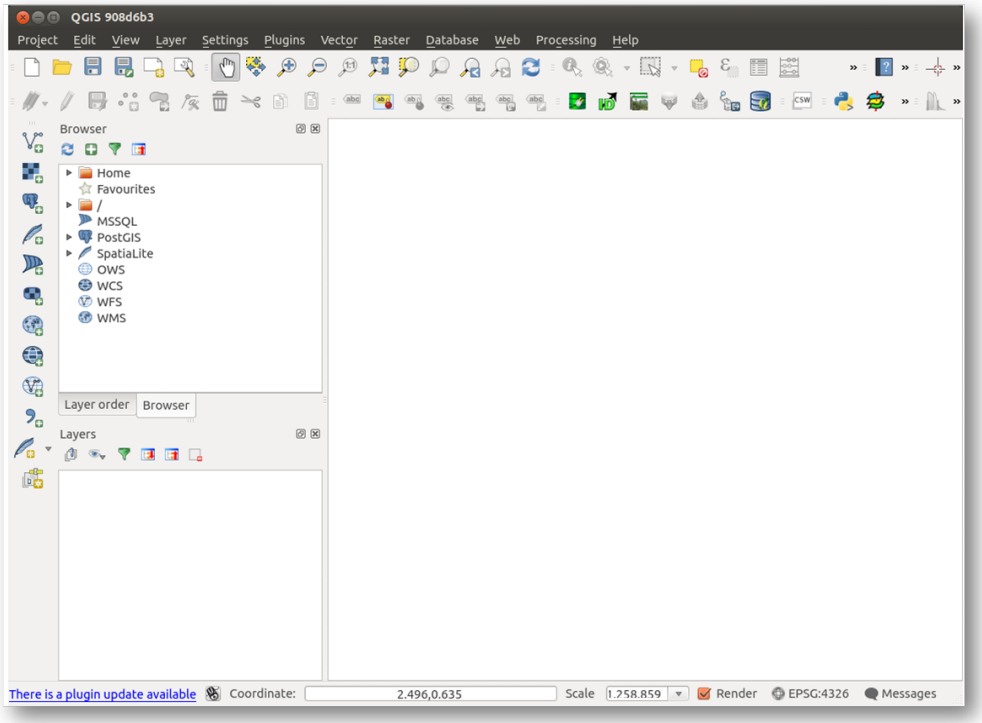


Figure 14: QGIS desktop GUI. The browser window at the left contains the button to add database connection (highlighted in orange).

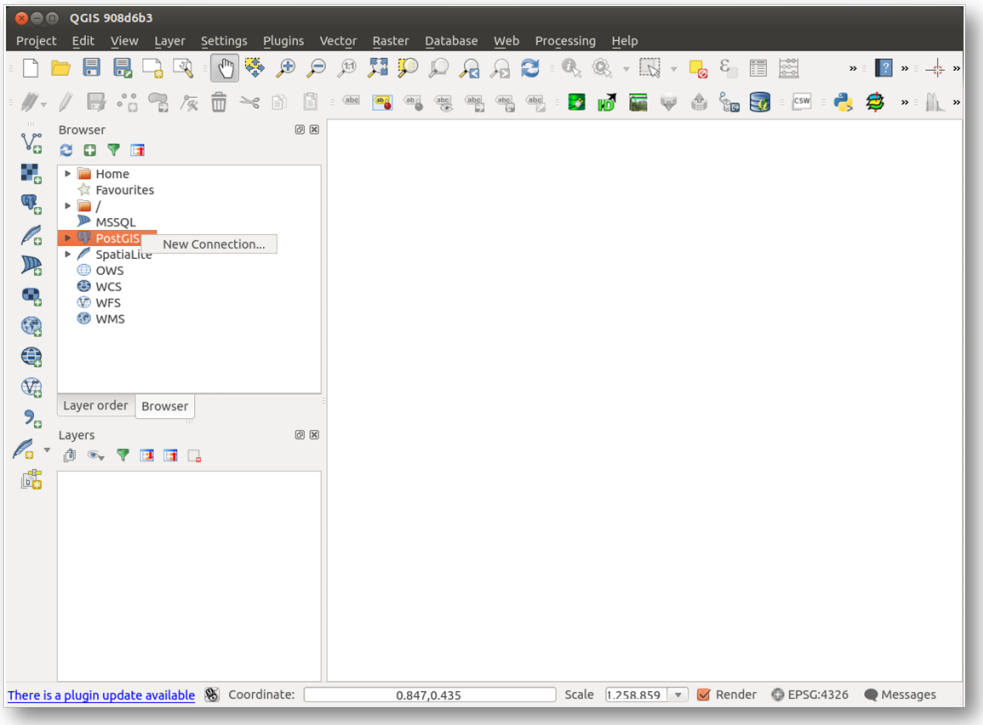


Figure 15: Establishing a connection with a database in QGIS desktop: click on new connection.

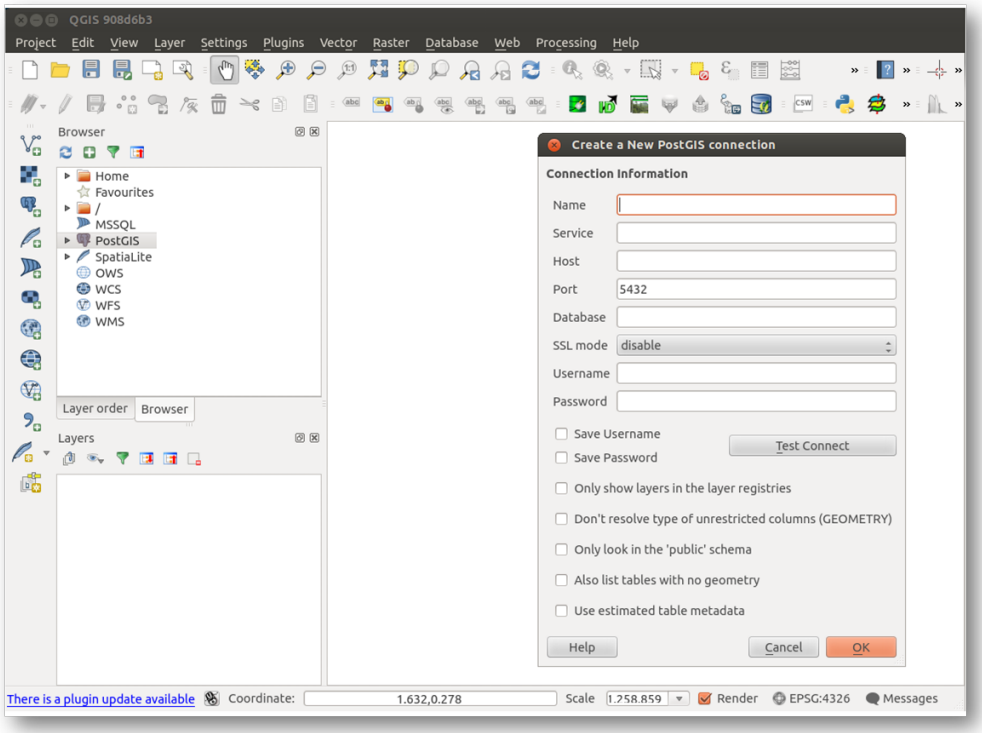


Figure 16: Establishing a connection with a databse in QGIS desktop: enter the connection parameters as indicated in the dialogue box.

To visualize the abstracted network click on the newly added database `de_power_database` in the browser window in QGIS. The database will expand the database components. Click on "public" which will expand all the tables available in the database "`de_power_150601`", see Fig.17.

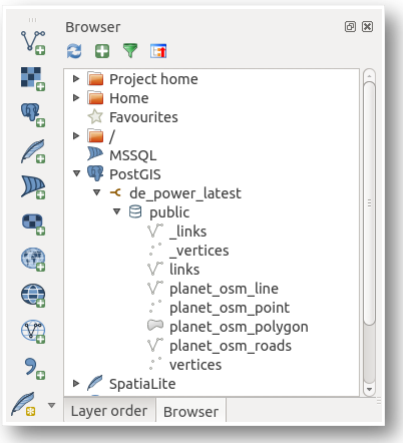


Figure 17: Database `de_power_database` components in QGIS browser.

For a better visualization, the territorial boundaries (or border) of Germany need also to be displayed as the background of the abstracted network. This is done using a relation containing the landmass border of Germany, which has the relation OSM-ID: 62781. The relation is down-

loaded as a .gpx file and loaded using QGIS, as follows:
In QGIS select Vector, GPS, GPS Tools (see Fig.18) or click the `gps_importer` GPS Tools icon in the toolbar and open the Load GPX file tab (see Fig.19). Browse to the folder `SciGRID/data/04_-visualization` in the **SciGRID** model folder, select the GPX file `landmass_germany.gpx` and click Open.

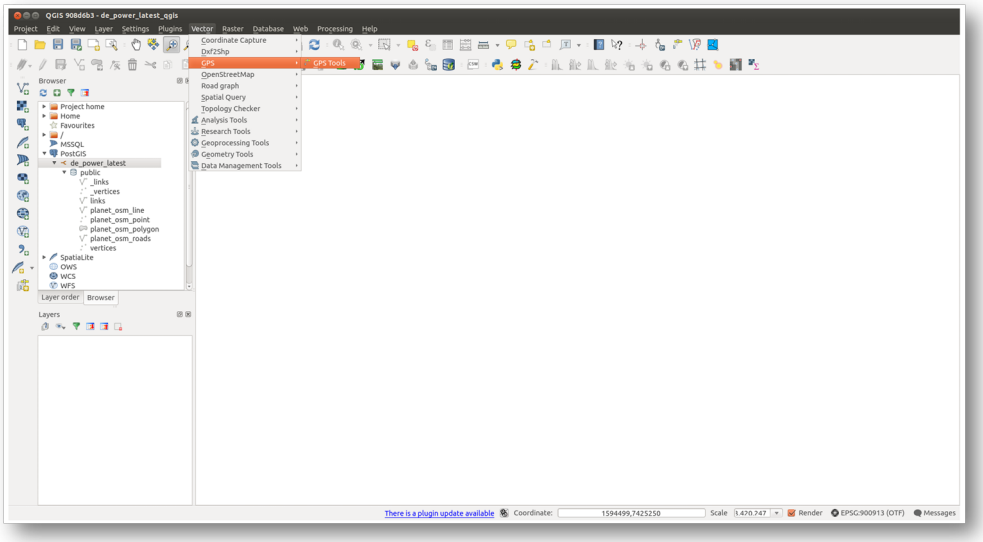


Figure 18: Load GPX files in QGIS Desktop.

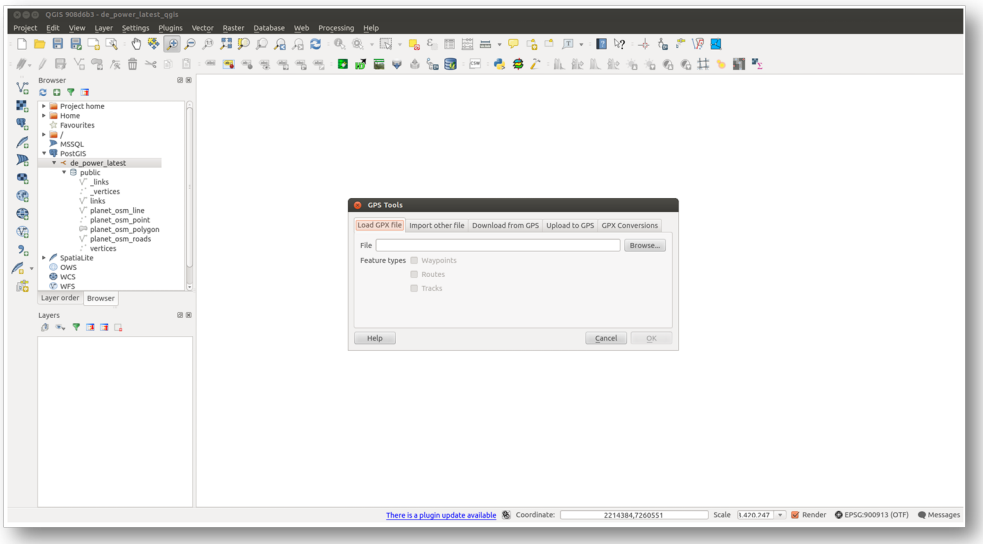


Figure 19: Load GPX file importer dialogue browser in QGIS Desktop.

The landmass border of Germany is then displayed (see Fig.20) and can be used as a background to the **SciGRID** model for Germany.

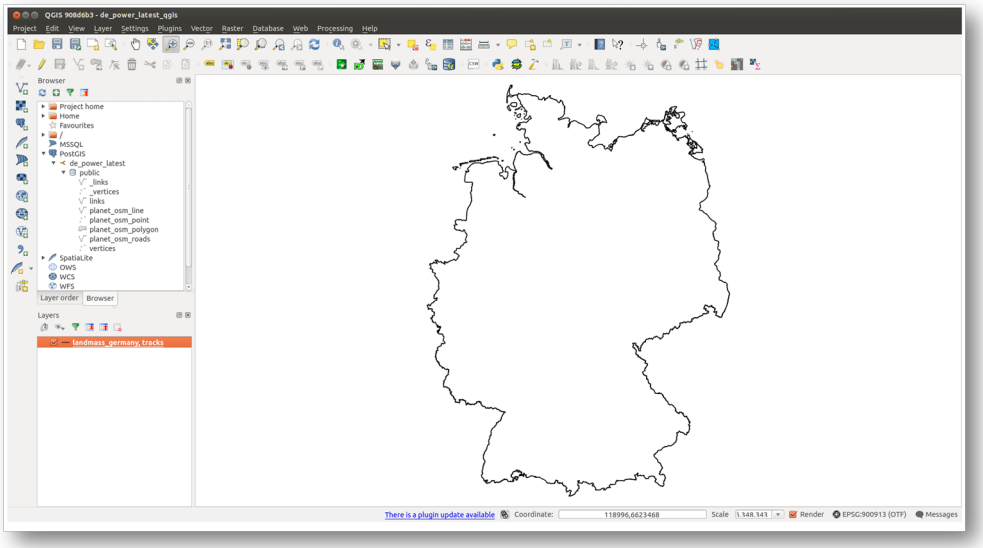


Figure 20: Representation of Germany landmass using the GPX file of the relation OSM-ID:62781 in QGIS Desktop.

To display the vertices and the links of **SciGRID** model network click on "`vertices`" and "`links`" tables in the QGIS browser. The resulting display includes the landmass border of Germany and the transmission network vertices and links outputted by **SciGRID**, see Fig.21.

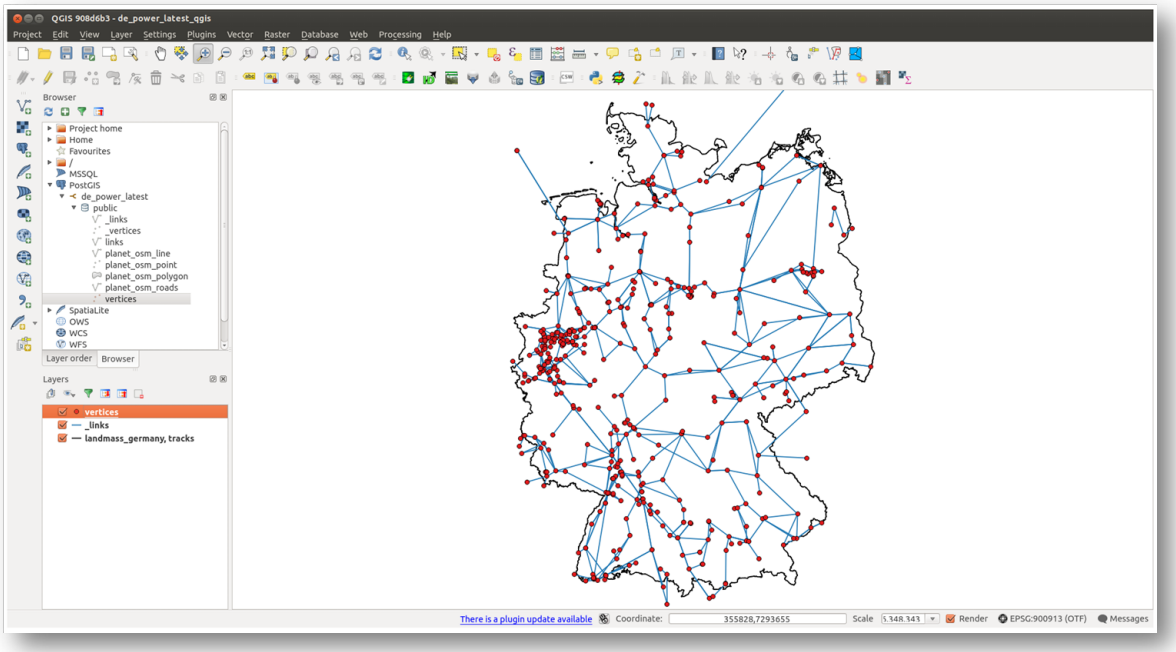


Figure 21: Visualization of the abstracted network vertices and links in QGIS Desktop GUI with the administrative boundaries of Germany as background image.

Further analysis of the data can be conducted using QGIS, for example "nearest neighbour" search. For more information about the available data analysis tools in QGIS refer to [20].

3.7 Running SciGRID with a script

The different steps necessary to build the **SciGRID** abstracted network model introduced in the previous sections can be run using either a makefile or a bash script. Both scripts extracts necessary information about the name of the database to be created, the path to the input and output folders from the file config.txt. This file is provided with the **SciGRID** model. In the following, the scripts and their usage are introduced.

3.7.1 Makefile

The makefile of the **SciGRID** model is provided in the `/code` folder. The advantage of using a makefile is that the different steps necessary to build the abstract network can be executed at once or separately. Another advantage is that the makefile enables the input parameters of **SciGRID** to be extracted from a configuration file (config.txt) and the folder paths to be indicated as environment variables. This reduces errors which are associated with naming the different necessary databases and indicating the input/output folders path.

Before running the makefile provided with the **SciGRID** model, make sure that you already installed *PostgreSQL* and *Osmosis*. For more information about how to install *PostgreSQL*, please visit the webpage [16]. Read Section 6.1.1 about how to install *Osmosis* on a Linux system.

The user needs to make sure that the names of the databases provided in the config.txt are unique, i.e. there are no databases with the same names which already exist. Otherwise the makefile will not execute and will exit with an error.

In the file config.txt, the user needs to indicate the names of the databases and the paths to the **SciGRID** data folder, Osmosis binary folder, osm2pgsql binary folder and the files postgis.sql and `spatial_ref_sys.sql`. The connection parameters to the PostgreSQL database have also to be indicated in the config.txt file. This step is necessary when using both the makefile or the bash script. If the paths to the folders and executable are not correctly indicated the scripts will not run properly.

The makefile is run by typing "make" in a bash terminal in the `/code` folder, where the makefile is stored. The makefile executes the following steps:

1. Extract the "power" data as .pbf file from the file planet-latest.osm.pbf, as shown in Section 3.3. The output is directed to the file `de_all_power_150601.osm.pbf` in the folder `SciGRID/data/02_osm_raw_power_data`. As the OSM planet file has a quite big size, it is not possible to provide it with the **SciGRID** model folder. However, the user can download the OSM planet file and filter the data for the "power" tag.

2. Create the *PostGIS* template database, following the same procedure introduced in Step 1, Section 3.4.

3. Create the hstore extension for the power database, as introduced in Step 1, Section 3.4.

4. Create an empty database `de_power_150601` using the template *PostGIS*. The empty database will hold the power data extracted form the OSM data file, following the same procedure introduced in Step 2, Section 3.4.

5. Export power OSM data extracted and filtered with Osmosis to the newly created database using *Osm2pgsql*, as introduced in Step 3, Section 3.4.

6. Runs the SciGRID.py script, containing the SciGRID class, which performs the abstraction on the "power" database as explained in Section 3.5. The file SciGRID.py is stored in the folder `SciGRID/code`, along with the necessary files it requires for the abstraction. These files are: `db_create_tables.py`, `relation_analysis.py`, `relation_abstraction.py`, `electrical_properties.py`, `store_network_as_csv.py` and `create_plots.py`.

   The input and output folders have to be indicated in the config.txt if they are different than the ones provided with the **SciGRID** model. When the abstraction is completed the vertices and links of the network are stored as .csv files in the folder `SciGRID/03_network`, a plot of the resulting network is stored in the `SciGRID/04_vizualisation` folder.

3.7.2 Shell script

A bash script named scigrid.sh is also provided in the `SciGRID/code` folder. This bash script is used to run the **SciGRID** model. The advantage of using a bash script over a makefile is that the user can interact with the script. As an example, if the created database has a name which is already used, an error message is displayed. The user can either change the database name or delete the existing database.

The bash script is run by typing `./scigrid.sh` in a bash terminal, after changing the permissions of the scigrid.sh file by typing `chmod u+rwx scigrid.sh`. The shell script also extracts the names and the paths needed from the `config.txt` file, and executes mainly the same steps as the makefile.

4 Troubleshooting

- Avoid naming databases or data files using the '-' symbol. This will create an error when using psql and Osm2pgSQL.
- Avoid using capital letters to name databases or data files. This will create an error when using psql and Osm2pgSQL.
- If you are using PgadminIII while running the **SciGRID** model, an error may occur when the databases are selected in PgadminIII. You need to close PgadminIII while running the **SciGRID** model.
- When executing the **SciGRID** model on Mac, the paths to postgis.sql and `spatial_ref_sys.sql` (both in the *Contrib* folder of PostgreSQL folder) may be different that the one on a Linux machine. The *PostgreSQL* folder on Mac is usually in the `/Library` folder.

5 Tutorial: Abstracted Transmission Network of Germany

A tutorial of the **SciGRID** model applied to Germany is included in the folder `SciGRID/tutorial`. To run the tutorial execute either the makefile or the bash script (both in the folder `SciGRID/code` provided.

6 How to install software necessary for SciGRID?

6.1 How to install Osmosis?

6.1.1 On Linux

The installation of Osmosis in the latest version is done in three steps.
Open a shell script, and type the following command to download the Osmosis build-in package for Linux:

```
$ wget http://bretth.dev.openstreetmap.org/osmosis-build/osmosis-latest.tgz
```

Then, unpack the downloaded osmosis build-in package by typing:

```
$ tar xvfz osmosis-latest.tgz
```

Finally, to be able to use Osmosis, change the rights of the Osmosis binary file by typing:

```
$ chmod a+x bin/osmosis
```

6.1.2 On Mac OS

The easiest way to install Osmosis on a Mac is to use homebrew [21]. In a shell terminal type:

```
$ brew install osmosis
```

6.2 How to install PostgreSQL?

6.2.1 On Linux

To install the built-in PostgreSQL and PostGIS for Linux, type the following command in a shell terminal:

```
$ sudo apt-get install postgresql-9.1 postgis
```

6.2.2 On Mac OS

To install PostgreSQL on Mac, a graphical installer is available, which includes PostgreSQL, and the StackBuilder utility for installation of additional packages. For PostgreSQL 9.0 and 9.1, Mac OS X 10.5 and above are supported, on 32 and 64-bit Intel CPUs, and PostgreSQL 9.2 and later support Mac OS X 10.6 and above on 32 and 64-bit Intel CPUs. The installer can be downloaded from the webpage [22].

There is also a possibility to install PostgreSQL by downloading the Postgres.app, which is a simple, native Mac OS X app that runs in the menu bar without the need of an installer. After downloading the Postgres.app [23], open it, and a PostgreSQL server is then ready and awaiting new connections. To shut the server down, you just need to close the app.

6.3 How to install Osm2pgsql?

A detailed webpage [24] on the OpenStreetMap wiki page exists where it is explained how to install Osm2pgsql on Linux and Mac OS systems.

References

[1] Open Data Commons Open Database License. Odbl. http://opendatacommons.org/licenses/odbl/.

[2] OpenStreetMap. Copyright and license. http://www.openstreetmap.org/copyright.

[3] Open Database License. (odbl) v1.0. http://opendatacommons.org/licenses/odbl/1.0/.

[4] Database Contents License. (dbcl) v1.0. http://opendatacommons.org/licenses/dbcl/1.0/.

[5] Open Data Commons Open Database License. (odbl). http://opendatacommons.org/licenses/odbl/#sthash.C4HJvcBW.dpuf.

[6] Apache License. Version 2. http://www.apache.org/licenses/LICENSE-2.0.

[7] Research Center Next Energy. EWE-Forschungszentrum fur Energietechnologie e. V. http://www.next-energy.de.

[8] OpenStreetMap. http://www.openstreetmap.org.

[9] SciGRID webpage. Scigrid developers. http://www.scigrid.de.

[10] Ito world website. http://www.itoworld.com.

[11] Index of mirrors openstreetmap.org. openstreetmap.org planet data. http://ftp.heanet.ie/mirrors/openstreetmap.org.

[12] openstreetmap.org planet data. http://wiki.openstreetmap.org/wiki/Planet.osm.

[13] OpenStreetMap. project wiki webpage. http://wiki.openstreetmap.org.

[14] OpenStreetMap project Wiki webpage. http://wiki.openstreetmap.org/wiki/.poly.

[15] Osmosis. Detailed information about osmosis. http://wiki.openstreetmap.org/wiki/Osmosis#Detailed_usage.

[16] Databases and data access APIs. openstreetmap wiki webpage. http://www.postgresql.org.

[17] Postgis, a spatial database extender for postgresql. http://postgis.net/.

[18] Deutsche Energy-Agentur (dena). Ausbau- und innovationsbedarf der stromverteilnetze in deutschland bis 2030. Technical report, 2012.

[19] Deutsche Energy-Agentur (dena). Website. http://www.dena.de/.

[20] QGIS. A free and open source geographic information system. http://www2.qgis.org.

[21] HomeBrew. http://brew.sh/.

[22] EnterpriseDB. Download postgresql. http://www.enterprisedb.com/products-services-training/pgdownload#osx.

[23] Postgres.app. Download postgres.app. http://postgresapp.com/.

[24] OpenStreetMap Wiki Page. Install osm2pgsql. http://wiki.openstreetmap.org/wiki/Osm2pgsql#Installation.