FlexiGIS Documentation

Release 1.0.0

tum-ens

Feb 22, 2023

USER'S GUIDE:

1 Getting Started **3**
- 1.1 FlexiGIS Installation . . . 4
- 1.2 Running FlexiGIS . . . 5
  - 1.2.1 FlexiGIS Config . . . 6
- 1.3 Directory Description . . . 8

2 FlexiGIS Components **9**
- 2.1 Module I: Spatial urban energy system platform . . . 9
- 2.2 Module II: Modeling urban energy requirements . . . 9
- 2.3 Module III: Flexibisation optimisation . . . 13

3 FlexiGIS Module **15**
- 3.1 Makefile . . . 15
- 3.2 flexigis_buildings.py . . . 16
- 3.3 flexigis_road.py . . . 17
- 3.4 weather_data.py . . . 17
- 3.5 data_format.py . . . 17
- 3.6 feedin.py . . . 17
- 3.7 flexigis_simulate.py . . . 17
- 3.8 flexigis_optimize.py . . . 17
- 3.9 plot_polygons.py . . . 18
- 3.10 flexigis_utils.py . . . 19

4 Publications List **23**

5 License **25**

6 Indices and tables **27**

Python Module Index **29**

Index **31**

FlexiGIS is an open source GIS-based platform for modelling energy systems and flexibility options in urban areas. FlexiGIS stands for Flexibilisation in Geographic Information Systems (GIS). It extracts, filters and categorises the geo-referenced urban energy infrastructure, simulates the local electricity consumption and power generation from on-site renewable energy resources, and allocates the required decentralised storage in urban settings. FlexiGIS investigates systematically different scenarios of self-consumption, it analyses the characteristics and roles of flexibilisation technologies in promoting higher autarky levels in cities. The extracted urban energy infrastructure are based mainly on Open Street Map data.

Author
Alaa Alhamwi, <alaa.alhamwi@dlr.de>

Organization

DLR - Institute of Networked Energy Systems, Department of Energy Systems Analysis

Version
1.0.0

Date
13.12.2019

CHAPTER

ONE

GETTING STARTED

FlexiGIS is developed and tested on Linux (Ubuntu 16.04.6). The tools and software used and their versions are listed in the following:

- Operating system: Ubuntu 16.04.6 LTS, Release: 16.04, Codename: xenial
- PostgreSQL version: 11.2 (64-bit)
- PostGIS version: 1.5.3
- osmosis version: 0.44.1
- osm2pgsql version: 0.88.1 (64bit id space)
- GNU Make version: 4.2.1
- Python: 3.6.7
- GNU bash: 4.3.48(1)-release (x86_64-pc-linux-gnu)

Before running FlexiGIS ensure the following dependencies are installed. PostgreSQL: To install *PostgreSQL*, refer to the PostgreSQL page. Please note that you need at least the same version or higher to run FlexiGIS.

Osmosis: To use Osmosis, unzip the distribution (which can be obtain from Osmosis) in the location of your choice. On unix/linux systems, make the bin/osmosis script executable (ie. chmod u+x osmosis). If desired, create a symbolic link to the osmosis script somewhere on your path (eg. ln -s appdir/bin/osmosis ~/bin/osmosis).

osm2pgsql: Instruction on how to download and install osm2pgsql for Linux systems are available on Osm2pgsql page.

Python: Ensure you can run python (with version 3 and above) on your OS. Download Python or the Anaconda distro. cbc solver: See here for cbc installation instruction.

To use the FlexiGIS spatial-temporal package download the FlexiGIS code and data folder as a zip file or clone the repository from the FlexiGIS GitHub repository. After downloading the FlexiGIS code, unzip the folder FlexiGIS in the location of your choice. The file structure of the FlexiGIS folder is as follows:

- **FlexiGIS**
  - ├── code
  - ├── data
  - │ ├── 01_raw_input_data
  - │ ├── *02_urban_output_data*
  - │ ├── *03_urban_energy_requirements*

- │ └── 04_Visualisation
- ├── doc
- ├── README.md
- └── requirements.txt

see *Directory Description* for detailed description of the FlexiGIS subdirectory contents.

Note: folder *02_urban_output_data* and *03_urban_energy_requirements* are created as output from FlexiGIS code, these folders are used to store output files.

1.1 FlexiGIS Installation

After ensuring all system requirements are satisfied, create a Python virtual environment where the required python dependencies can be installed using pip. Python virtual environment can be created by following the steps described here. After creating a python virtual environment, FlexiGIS can easily be installed by following these steps:

1. First clone the FlexiGIS code from the GitHub repository. In the next step, create a Python virtual environment (e.g. _env_name) where the required python dependencies can be installed using pip

```
user@terminal:~$ git clone https://github.com/FlexiGIS/FlexiGIS.git
user@terminal:~$ python3 -m venv _env_name
```

2. activate the virtual environment

```
user@terminal:~$ source _env_name/bin/activate
```

3. cd into the cloned FlexiGIS folder and install the required python dependencies

```
(_env_name) user@terminal:~$ cd ../FlexiGIS
(_env_name) user@terminal:~/FlexiGIS$ pip install -r requirements.txt
```

clone the oemof-feedinlib package from flexigis github repository(recommended) and install localy for the renewable feedin simulations. Also install the oemof-solph python package for the modelling and optimization of energy systems. An additional eomof package oemof_visio is required as a dependency for generating nice plots of the optimization results. *Note: The default solver used here for the linear optimization by FlexiGIS is the 'CBC' slover*

4. install oemof packages for feedin and optimization

```
(_env_name) user@terminal:~/FlexiGIS$ git clone https://github.com/
↪FlexiGIS/feedinlib.git
(_env_name) user@terminal:~/FlexiGIS$ pip install -e feedinlib
(_env_name) user@terminal:~/FlexiGIS$ pip install oemof.solph
(_env_name) user@terminal:~/FlexiGIS$ pip install git+https://github.com/
↪oemof/oemof_visio.git
```

Now you are ready to excute FlexiGIS using the make commands.

1.2 Running FlexiGIS

To run the first two components of the FlexiGIS package, Go into the folder code, check the parameters in *config.mk* file. Ensure you have all parameters in the config.mk properly set, see *FlexiGIS Config*. Also ensure the poly file of the spatial location of choice is available in the data/01_raw_input_data directory before running FlexiGIS.

FlexiGIS is executed using the make command, To run the available makefile options, go into the code folder of the flexigis directory in your Linux terminal. The available make options are:

- make-all executes multiple make options, from download to the simulation of load and PV profiles for the urban infrastructures, and finally the optimization of electricity supply and the alocated storage system:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make all
```

- make download, downloads spatial OSM data of a given location:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make download
```

- filters the downloaded OSM planet data:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make filter_data
```

- exports the spatially filtered OSM data to a PostgreSQL database using osm2pgsql:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make export_data
```

- extract highway, building and landuse data from filtered OSM data, stores them as shape files and also generate geopandas plots of the urban infrastructures:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make abstract_data
```

- simulates wind and pv power for given weather data:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make feedin
```

- executes the simulation of load and PV profiles for urban infrastructure:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make demand_supply_simulation
```

- provides option to either drop the database or not:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make drop_database
```

- Optimizes the feedin of the energy system and allocates the required decentralised storage in urban settings:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make optimization
```

- Download ECMWF ERA5 weather data from Climate Data Store CDS, based on the coupled feedinlib interface:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make weather_data
```

- provides option to either drop the database or not:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make feedin_data_format
```

- make example can be run to generate an example simulation of aggregated load and PV profile for Oldenburg and also model the optimal allocated storage and onsite renewable supply:

```
(_env_name) user@terminal:~/FlexiGIS/code$ make example
```

The make example the routine imports spatially filtered OSM Highway, landuse and building data stored as csv files in the **../data/01_raw_input_data/example_OSM_data** folder. After running the FlexiGIS package using the makefile commands, the resulting aggregated load and PV profiles of the urban infrastructure, and optimization results are stored in folder **../data/03_urban_energy_requirements**, also static plots of the urban infrastructures and simulated load and PV profiles are created and stored in the data/04-visualisation folder. To visualise the extracted georeferenced urban infrastructures data interactively, the generated shape file of the extracted urban infrastructures, can be used in QGIS to generate interactive plots.

Note: After running the FlexiGIS package using the makefile, the resulting aggregated load and PV profiles of the urban infrastructure are stored as .csv file, and the result from the optimization is stored as pickle file in folder data/03_urban_energy_requirements.

1.2.1 FlexiGIS Config

Before executing the available FlexiGIS make commands on your environment, ensure that the config.mk file variables are properly configured. Change the following default variables according to your respective system environment, the OpenStreetMap data location of interest and renewable feedin simulation parameters.

For OpenStreetMap data download (spatial location):

- URL of the OSM raw data (used for OSM raw data download):

```
# download other pbf file for other spatial areas from geofabrik
# Change the pbf file to the pbf file name of the location of interest.

OSM_raw_data_URL:=https://download.geofabrik.de/europe/germany/
↪niedersachsen-latest.osm.pbf
```

- Name of the OSM raw data file (used for data filtering by osmosis):

```
# replace with downloaded pbf file name
OSM_raw_data:=../data/01_raw_input_data/niedersachsen-latest.osm.pbf
```

- Name of the bounding polygon file (used for data filtering by osmosis):

```
# Use other polyfiles for other spatial areas
polyfile:=../data/01_raw_input_data/Oldenburg.poly
```

For PostgreSQL database connection, parameters should be change to match user's database

- PostgreSQL connection parameters:

```
# change parameter to match your database connection
postgres_cluster:=9.1/main
postgres_database:=database_name
postgres_user:=user_name
postgres_port:=port_number
postgres_host:=host_address
```

To download ERA5 weather data using make weather_data, the below lines in the config.mk file should be properly edit to suit personal prefrence.

- Weather data download parameters:

```
# stores the downloade netcdf weather data as ERA5_data.nc
target_file:= ../data/01_raw_input_data/ERA5_data.nc

# select weather data timestamp or download period
start_date:=2015-01-01
end_date:= 2015-12-31

#set region to "True" or "False" if you wish to download weather for a␣
↪region or for single location
region:=False
# For single coordinate or location single location (e.g single location␣
↪in Oldenburg)
lon_single_location:=8.10
lat_single_location:=53.15

# For download of weather data for a region (e.g: Berlin region)
# Longitude 'west'-'East' and Latitude 'North'-'South'
lon_region:= 13.1,13.6
lat_region:= 52.3,52.7
```

- To generate renewable feedin time series:

```
# defualt power system parameters
hub_height:= 135
# wind data in feedinlib format, for wind power simulation
wind_data:= wind_data.csv


pv_panel:= Advent_Solar_Ventura_210___2008_
inverter_type:= ABB__MICRO_0_25_I_OUTD_US_208__208V_
# pv data in feedinlib format, for pv power simulation
solar_data:= solar_data.csv
```

see feedinlib-pv on how to get available PV power system parameters and feedinlib-wind on how to get available wind power system parameters.

1.3 Directory Description

- code

  The folder *code* contains a "makefile" for running the FlexiGIS model, the configuration file "config.mk" which contains different parameters necessary to run the FlexiGIS model. Also inside the code folder are different python scripts that are executed after make commands are ran on a Linux terminal.

- data

  The folder *data* contains or should contain all the input data needed and output data generated after running FlexiGIS. The data folder contains 4 sub folders, which are introduced as follows:

  01_raw_input_data: contains the poly file of the respective investigated urban area (e.g. Oldenburg.poly). The poly file is used to filter the OSM planet datasets to include the data of case study. This is achieved by Osmosis and executed by running "make filter_data". However, the OSM planet data file is stored in this folder. The OSM planet data is filtered spatially to include urban infrastructure such as building, highway and landuse data.

  Note: A sample OSM planet data file (used for in the case study) is not included in the present distribution of FlexiGIS because of its file size. However, OSM data for desired location can be downloaded via https://download.geofabrik.de as a pbf file.

  Other input data in this folder are, Urban style data "urban.style", Standard Load profile, economic data for storage and feedin investment optimization, and weather data (wind and pv data), used for the urban energy requirements simulation.

  02_urban_output_data: contains the resulting abstracted urban infrastructure based on landuse, building and highway tags from the OSM planet data.

  03_urban_energy_requirements: contains the results data from Module II and III, which are calculated load and normalized PV and wind power time series for different urban infrastructure and dispatch optimization result stored as an pickle file.

  04_visualization: contains plots of the abstracted urban infrastructure and aggregated load and PV energy requirements.

- doc

  Here FlexiGIS documentation and user guide are stored.

CHAPTER

TWO

FLEXIGIS COMPONENTS

FlexiGIS is structured into three modules:

2.1 Module I: Spatial urban energy system platform

This module establishes urban energy infrastructure. It extracts, acquires and processes urban geo-referenced data extracted from OpenStreetMap. In order to extract the OpenStreetMap georeferenced datasets of urban energy infrastructure and its required features, this module derives an automatised extraction procedure. Firstly, the raw OpenStreetMap data is downloaded from the OpenStreetMap database for the investigated urban space from Geofabrik. Second, the OpenStreetMap datasets are filtered for the respective case study defined by a .poly file using osmosis, an open source java tool. The OpenStreetMap data are filtered for the following OSM tags: landuse, building, and highway.

Note:

- ***landuse*:**
  provides information about the human use of land in the respective area (see *Figure 1*)
- ***building*:**
  describes all mapped objects considered as buildings of different types. e.g houses, schools, etc. (see *Figure 2*)
- ***highway*:**
  describes all lines considered as streets, roads, paths, etc. (see *Figure 3*)

After filtering the OSM raw data, the geo-referenced building and highway infrastructure (case study: the city of Oldenburg) are exported to a relational postgis-enabled database using the open source *osm2pgsql*. Theses datasets can be exported as .csv files or visualised as maps (see Figures 1-3).

2.2 Module II: Modeling urban energy requirements

This module simulates urban energy requirements (consumption and generation). The spatio-temporal electricity consumption and renewable energy generation from PV and wind, in the defined urban area are modeled. This component models the combined spatial parameters of urban geometries (Module I) and links them to real-world applications using GIS. Here, a bottom-up simulation approach is developed to calculate local urban electricity demand and power generation from available renewable energy resources. For instance, using open source datasets like Standardised Load Profiles and publicly available weather

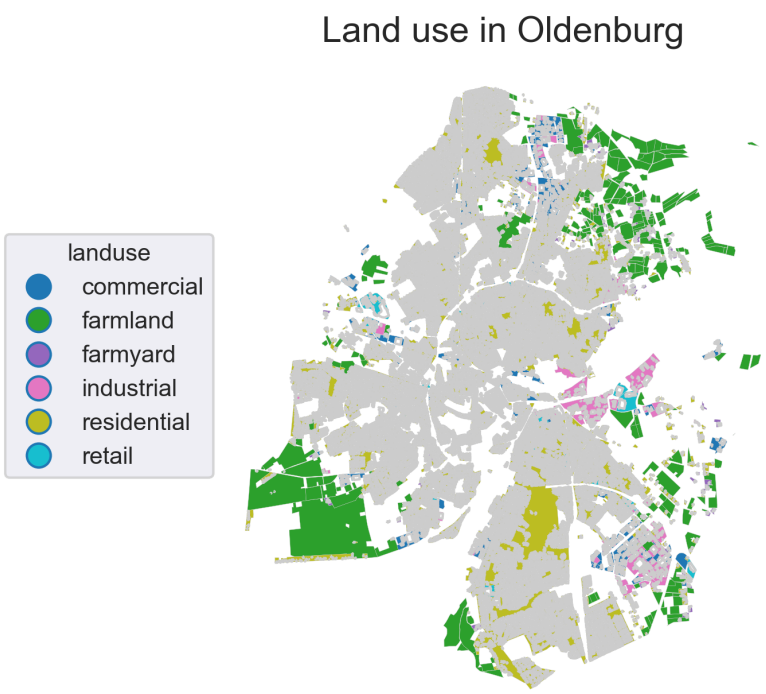

Fig. 1: Extracted OpenStreetMap *landuse* datasets for the city of Oldenburg. Credits: OpenStreetMap contributors.

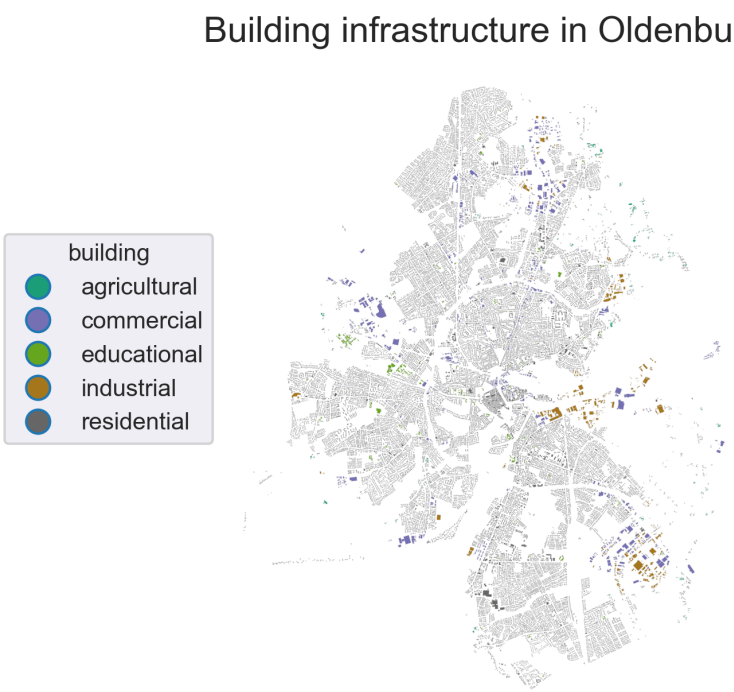

Fig. 2: Extracted OpenStreetMap *building* datasets for the city of Oldenburg. Credits: OpenStreetMap contributors.

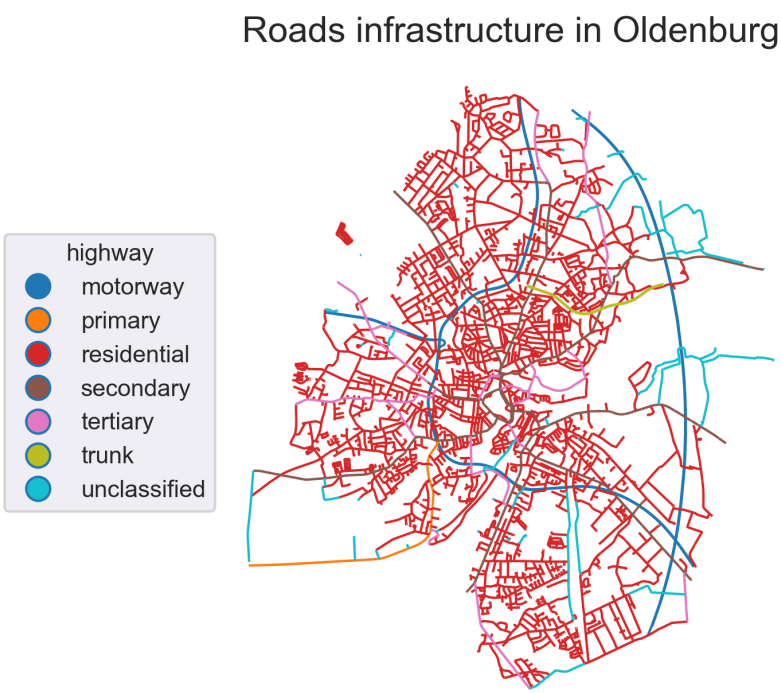

Fig. 3: Extracted OpenStreetMap *highway* datasets for the city of Oldenburg. Credits: OpenStreetMap contributors.

data. *Figure 4* shows the generated quarter-hourly time series of the aggregated load and PV power supply profile for investigated case study.

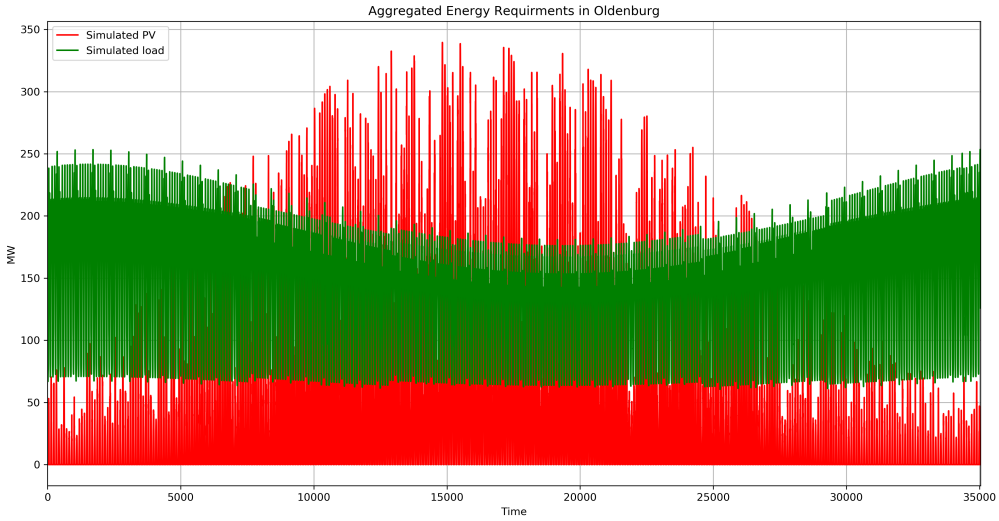

Fig. 4: Simulated electricity consumption (green) and solar power generation (red) for the city of Oldenburg.

2.3 Module III: Flexibisation optimisation

The spatial-temporal simulation outputs from Module I and II are time series of electricity demand and supply. Theses generated datasets will be used by the eomof-solph model as inputs to the linear optimisation problem. This module aims to determine the minimum system costs at the given spatial urban scale while matching simultaneously the simulated electricity demand. In addition, it aims to allocate and optimise distributed storage and other flexibility options in urban energy systems.

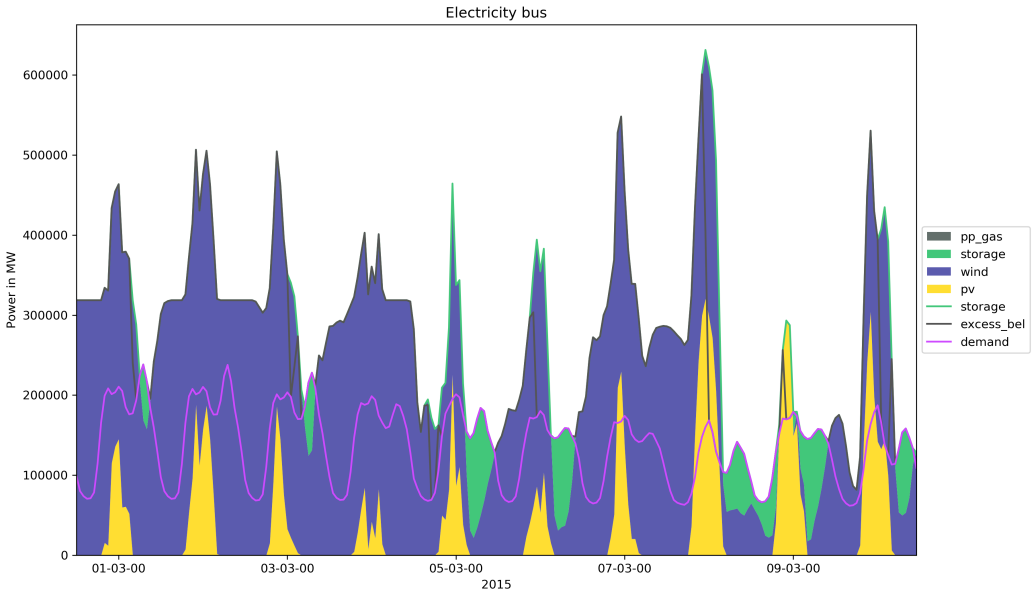

Fig. 5: Example result of optimal energy requirements, which minimize investement cost simulated for the city of Oldenburg.

CHAPTER

THREE

FLEXIGIS MODULE

3.1 Makefile

Contains make rules for the automation of all flexiGIS modules.

- Download:

```
wget -nv -O $(OSM_raw_data) $(OSM_raw_data_URL)
```

Download the OSM raw data from https://download.geofabrik.de

- filter_data:

```
osmosis \
    --read-pbf file=$(OSM_raw_data) \
    --tag-filter accept-ways building=* --used-node \
    --bounding-polygon file=$(polyfile) \
    --buffer outPipe.0=building \
    --read-pbf file=$(OSM_raw_data) \
    --tag-filter accept-ways highway=* --used-node \
    --bounding-polygon file=$(polyfile) \
    --buffer outPipe.0=highway \
    --read-pbf file=$(OSM_raw_data) \
    --tag-filter accept-ways landuse=* --used-node \
    --bounding-polygon file=$(polyfile) \
    --buffer outPipe.0=landuse_1 \
    --read-pbf file=$(OSM_raw_data) \
    --tag-filter accept-relations landuse=* --used-node \
    --bounding-polygon file=$(polyfile) \
    --buffer outPipe.0=landuse_2 \
    --merge inPipe.0=landuse_1 inPipe.1=landuse_2 \
    --buffer outPipe.0=landuse_all \
    --merge inPipe.0=landuse_all inPipe.1=building \
    --buffer outPipe.0=landuse_building \
    --merge inPipe.0=landuse_building inPipe.1=highway \
    --write-pbf file=$(OSM_merged_data)
```

Merge and Filter the OSM raw geo-urban datasets using Osmosis

- export_data:

```
export PGPASSWORD=$(postgres_password); createdb -U $(postgres_user) -h
↪$(postgres_host) $(postgres_database);
export PGPASSWORD=$(postgres_password); $(osm2pgsql_bin) -r pbf --
↪username=$(postgres_user) --database=$(postgres_database) --host=
↪$(postgres_host) --port=$(postgres_port) -s \
-C $(osm2pgsql_cache) --hstore --number-processes $(osm2pgsql_num_
↪processes) $(OSM_merged_data);
```

Export the Filtered OSM data to Postgres Server using osm2pgsql

- abstract_data:

```
python flexigis_road.py -U $(postgres_user) -P $(postgres_port) -H
↪$(postgres_host) -D $(postgres_database)
python flexigis_buildings.py -U $(postgres_user) -P $(postgres_port) -H
↪$(postgres_host) -D $(postgres_database)
python plot_polygons.py
```

Execute abstraction module on filtered OSM dataset

- demand_supply_simulation:

```
python flexigis_simulate.py
```

Simulates urban energy requirement using abstracted dataset

- drop_database:

```
dropdb --username=$(postgres_user) --port=$(postgres_port) --host=
↪$(postgres_host) $(postgres_database)
```

Provides option to delete or retain database

- example:

```
python example.py
python plot_polygons.py
python flexigis_simulate.py
```

Runs a test example data abstraction and simulation using filtered OSM data stored as csv file

3.2 flexigis_buildings.py

Get Building and Landuse geo-referenced data. Output geo-referenced building data are categorized into Agricultural, Commercial, Educational, Industrial and Residential. Outputs data are stored as CSV files.

3.3 flexigis_road.py

Get geo-referenced Highway data. Highway categories such as motor ways, squares, bus stop, links, etc. Outputs are exported to csv files.

3.4 weather_data.py

Download ECMWF ERA5 weather data from CDS using the feedinlib interface. Returns a netcdf file of the downloaded weather data.

3.5 data_format.py

Prepare weather data parameters in feedinlib format. Returns csv files for renewable feedin calculation

3.6 feedin.py

Calculate the renewable feedin time series, using the feedinlib package.

3.7 flexigis_simulate.py

Simulates Urban Energy Requirements, outputs are stored stored as CSV file.

3.8 flexigis_optimize.py

This script is use to model an energy system, and derive the optimal RE feedin capacity and storage investment. The optimisation is carried out using the oemof solph package. The below codes are taken from the oemof-example codes github repository. The optimization settings are described below by the following parameters:

- optimize wind, pv, gas_resource and storage
- set investment cost for wind, pv and storage
- set gas price for kWh

See links: oemof-example and oemof-plotting_examples

3.9 plot_polygons.py

Plot georeferenced data (landuse, building, highway) using geopandas.

plot_polygons.**plot_building**(*df_building*, *legend_box*, *fig_size*, *font_size*, *face_color*, *destination*)

Plot polygons for buildings.

Parameters

- **df_building** (*DataFrame*) – building georeferenced data (geodataframe)
- **legend_box** (*tuple(float)*) – tuple of floats, eg (0.0, 0.05, 0.01, 0.7)
- **fig_size** (*tuple(int)*) – figure size
- **font_size** (*int*) – title font size
- **face_color** (*str*) – backgroud color (eg, white, black)
- **destination** (*str*) – plot destination path

plot_polygons.**plot_landuses**(*df_building*, *df_landuse*, *legend_box*, *fig_size*, *font_size*, *face_color*, *destination*)

Plot polygons for landuse.

Parameters

- **df_building** (*DataFrame*) – dbuilding georeferenced data (geodataframe)
- **df_landuse** (*DataFrame*) – landuse georeferenced data
- **legend_box** (*tuple(float)*) – tuple of floats, eg (0.0, 0.05, 0.01, 0.7)
- **fig_size** (*tuple(int)*) – figure size
- **font_size** (*int*) – title font size
- **face_color** (*str*) – backgroud color (eg, white, black)
- **destination** (*str*) – plot destination path

plot_polygons.**plot_roads**(*df_highway*, *legend_box*, *fig_size*, *font_size*, *face_color*, *destination*)

Plot lines (highway).

Parameters

- **df_highway** (*DataFrame*) – highway georeferenced data (geodataframe)
- **legend_box** (*tuple(float)*) – tuple of floats, eg (0.0, 0.05, 0.01, 0.7)
- **fig_size** (*tuple(int)*) – figure size
- **font_size** (*int*) – title font size
- **face_color** (*str*) – backgroud color (eg, white, black)
- **destination** (*str*) – plot destination path

Plot geo-referenced data (landuse, building, highway) using geopandas.

3.10 flexigis_utils.py

Helper functions for FlexiGIS data abstraction.

flexigis_utils.**compute_area**(*dataset*, *width*)

Compute area for each line feature and return a dataframe object.

Parameters

- **dataset** (*DataFrame*) – OSM planet data
- **width** (*dict*) – unique highway category as *key* and the width in meters as the *value*

Returns
dataframe containing an "area" attribute

Return type
DataFrame

flexigis_utils.**data_to_file**(*dataset*, *name='name'*)

Write a dataframe to a csv/shape file.

Parameters

- **dataset** (*DataFrame*) – OSM planet data
- **name** (*str*) – file name of the output csv file (eg. *table_name*)

flexigis_utils.**dbconn_from_args**()

Parse database credentials as environmental variables.

Get database connection from command-line arguments or environment ariables. Reuse environment variables from libpq/psql (see http://www.postgresql.org/docs/9.1/static/libpq-envars.html)

flexigis_utils.**get_csv_categories**(*destination*, *name='category_name'*)

Get csv files from a folder (temp folder).

Parameters

- **destination** (*str*) – Path to categorised buildings csv files.
- **name** (*str*) – The name of building type based on landuse category.

Returns
dataframe of buildings in a landuse category

Return type
pandas.DataFrame

flexigis_utils.**get_data_from_buildings**(*data_building*, *mask_data*)

Extract building for a landuse category.

Parameters

- **data_building** (*GeoDataFrame*) – building data.
- **mask_data** (*GeoDataFrame*) – building/landuse intersects.

Returns
dataframe of buildings in a landuse category

Return type
pandas.DataFrame

flexigis_utils.**get_features**(*mask_landuse*, *res_intersects*, *select='category'*)

Get features polygons.

Parameters

- **mask_landuse** (*GeoDataFrame*) – categories of building based on landuse
- **res_intersects** (*GeoDataFrame*) – intersect between building and landuse

Returns
GeoDataFrame of building/landuse intersects.

Return type
pandas.GeoDataFrame

flexigis_utils.**get_intersects**(*data_building*, *data_landuse*)

Get intersects for between building category and landuse.

Parameters

- **data_building** (*GeoDataFrame*) – building data
- **data_landuse** (*GeoDataFrame*) – landuse data

Returns
GeoDataFrame of intersects between landuse and building.

Return type
pandas.GeoDataFrame

flexigis_utils.**get_polygons**(*data*, *select='feature'*)

Get building classification.

Parameters

- **data** (*DataFrame*) – OSM planet data
- **select** (*str*) – building data feature (eg. *apartment*, *farmhouse*)

Returns
GeoDataFrame of building category type

Return type
pandas.GeoDataFrame

flexigis_utils.**mask_landuse_data**(*data_landuse*, *res_intersects*)

Mask intersections with landuse data.

Parameters

- **data_landuse** (*GeoDataFrame*) – landuse data
- **res_intersects** (*GeoDataFrame*) – intersect between building and landuse

Returns
GeoDataFrame of building/landuse intersects.

Return type
pandas.GeoDataFrame

flexigis_utils.**shape_legend**(*node*, *ax*, *handles*, *labels*, *reverse=False*, ***kwargs*)

Plot legend manipulation. This code is copied from the oemof example script see link here: https://github.com/oemof/oemof-examples/tree/master/oemof_examples/oemof.solph/v0.3.x/plotting_examples

Contains helper functions for FlexiGIS data abstraction.

CHAPTER

FOUR

PUBLICATIONS LIST

- Alaa Alhamwi, W. Medjroubi, T. Vogt, C. Agert, *GIS-based urban energy systems models and tools: Introducing a model for the optimisation of flexibilisation technologies in urban areas*, Applied Energy, Volume 191, 1-9 (2017), 10.1016/j.apenergy.2017.01.048, https://doi.org/10.1016/j.apenergy.2017.01.048.
- Alaa Alhamwi, W. Medjroubi, T. Vogt, C. Agert, *OpenStreetMap data in modelling the urban energy infrastructure: a first assessment and analysis*, Energy Procedia, Volume 142, 1968–1976 (2017), 10.1016/j.egypro.2017.12.397, https://doi.org/10.1016/j.egypro.2017.12.397.
- Alaa Alhamwi, W. Medjroubi, T. Vogt, C. Agert, *Modelling urban energy requirements using open source data and models*, Applied Energy, Volume 231, 1100-1108 (2018), 10.1016/j.apenergy.2018.09.164, https://doi.org/10.1016/j.apenergy.2018.09.164.
- Alaa Alhamwi, W. Medjroubi, T. Vogt, C. Agert, *FlexiGIS: an open source optimisation platform for the optimisation of flexibility options in urban energy systems*, Energy Procedia, Volume 152, 941-949 (2018), 10.1016/j.egypro.2018.09.097, https://doi.org/10.1016/j.egypro.2018.09.097.
- Alaa Alhamwi, W. Medjroubi, T. Vogt, C. Agert, *Development of a GIS-based platform for the allocation and optimisation of distributed storage in urban energy systems*, Applied Energy, Volume 251, 113360 (2019), 10.1016/j.apenergy.2019.113360, https://doi.org/10.1016/j.apenergy.2019.113360.
- Joseph Ranalli and Alaa Alhamwi, *Configurations of renewable power generation in cities using open source approaches: With Philadelphia case study*, Applied Energy, Volume 269, 115027 (2020), 10.1016/j.apenergy.2020.115027, https://doi.org/10.1016/j.apenergy.2020.115027
- Alhamwi et. al, *Modelling urban street lighting infrastructure using open source data*, to be submitted to IEEE Access, (2020).

CHAPTER

FIVE

LICENSE

The FlexiGIS is licensed under the BSD-3-Clause, “New BSD License” or “Modified BSD License”. Redistribution and use in source and binary forms, with or without modification, are permitted. For more information concerning the BSD-3C and the description of the terms under which you can use the FlexiGIS code, please see opensource licenses.

The OpenStreetMap (OSM) data is available under the Open Database License (ODbL). A description of the ODbL license is available at the opendatacommons licenses. OpenStreetMap cartography is licensed as CC BY-SA. For more information on the copyright of OpenStreetMap please visit the openstreetmap copyright page. The OpenStreetMap data distributed is available under the Open Database License ODbL.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

FlexiGIS, 13
flexigis_utils, 19
FlexiGISdoc, 9

p

plot_polygons, 18

INDEX

C

compute_area() (*in module flexigis_utils*), 19

D

data_to_file() (*in module flexigis_utils*), 19
dbconn_from_args() (*in module flexigis_utils*), 19

F

FlexiGIS
    module, 1, 4, 7–9, 13, 15–18
flexigis_utils
    module, 19
FlexiGISdoc
    module, 9

G

get_csv_categories() (*in module flexigis_utils*), 19
get_data_from_buildings() (*in module flexigis_utils*), 19
get_features() (*in module flexigis_utils*), 20
get_intersects() (*in module flexigis_utils*), 20
get_polygons() (*in module flexigis_utils*), 20

M

mask_landuse_data() (*in module flexigis_utils*), 20
module
    FlexiGIS, 1, 4, 7–9, 13, 15–18
    flexigis_utils, 19
    FlexiGISdoc, 9
    plot_polygons, 18

P

plot_building() (*in module plot_polygons*), 18
plot_landuses() (*in module plot_polygons*), 18
plot_polygons
    module, 18
plot_roads() (*in module plot_polygons*), 18

S

shape_legend() (*in module flexigis_utils*), 20