eGo Documentation

Release 0.3.0

open_eGo-Team

Sep 07, 2018

Contents

Note: Note, the data source of eGo relies on the Open Energy Database. - The registration for the public accessible API can be found on openenergy-platform.org/login.

CHAPTER 1

Overview

1.1 The eGo tool

The python package eGo is a toolbox and application which connects the tool **eTraGo** - with an optimization of flexibility options for transmission grids based on PyPSA and **eDisGo** - which analysis and optimization distribution grids.

1.1.1 The open_eGo project

This software project is part of the research project open_eGo.

1.1.2 The OpenEnergy Platform

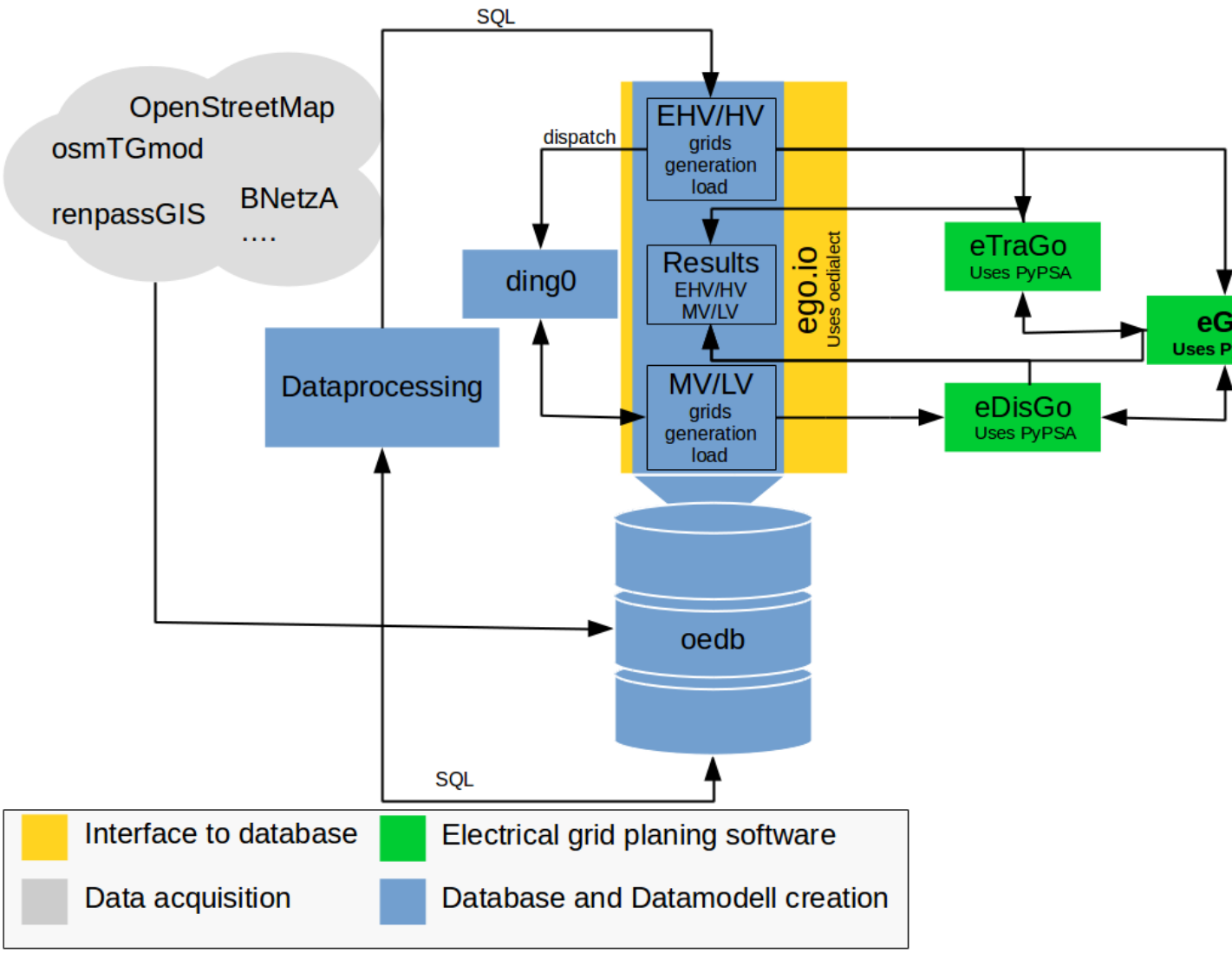
Within this project we developed the OpenEnergy Platform which this software is using in order to get and store the in- and output data. Before you start to calculate a registration on the platform is needed. For more see openenergy-platform and login.

The OpenEnergy platform mainly addresses students, researchers and scientists in the field of energy modelling and analytics as well as interested persons in those fields. The platform provides great tools to make your energy system modelling process transparent. All data of the open_eGo project are stored at this platform. Learn more about the database access.

1.1.3 Model overview

eTraGo

The python package eTraGo provides an optimization of flexibility options for transmission grids based on PyPSA. A speciality in this context is that transmission grids are described by the 380, 220 and 110 kV in Germany. Conventionally the 110kV grid is part of the distribution grid. The integration of the transmission and 'upper' distribution grid is part of eTraGo.


SQL
OpenStreetMap
osmTGmod
renpassGIS
BNetzA
....
dispatch
EHV/HV
grids
generation
load
Results
EHV/HV
MV/LV
MV/LV
grids
generation
load
ego.io
Uses oedialect
ding0
Dataprocessing
eTraGo
Uses PyPSA
eDisGo
Uses PyPSA
oedb
SQL
Interface to database
Electrical grid planing software
Data acquisition
Database and Datamodell creation

The focus of optimization are flexibility options with a special focus on energy storages and grid expansion measures.

eDisGo

The python package eDisGo provides a toolbox for analysis and optimization of distribution grids. It is closely related to the python project Ding0 as this project is currently the single data source for eDisGo providing synthetic grid data for whole Germany. Learn more here.

Dataprocessing

For the open_eGo project several python packages are developed which are feeded by the input data of the data processing. The dataprocessing is writen in SQL and Python. Learn more here.

ego.io

The `ego.io` is a SQLAlchemy Interface to the OpenEnergy database (oedb). The oedb tables as ORM objects are defined here and small helpers for I/O tasks are contained. Learn more here.

Dingo

The DIstribution Network GeneratOr (Ding0) is a tool to generate synthetic medium and low voltage power distribution grids based on open (or at least accessible) data. Learn more here.

1.1.4 Supported by

This project is supported by the German Federal Ministry for Economic Affairs and Energy (BMWI).

1.1.5 License

1.1.6 Partner

1.2 Installation

eGo is designed as a Python package therefore it is mandatory to have Python 3 installed. If you have a working Python3 environment, use pypi to install the latest eGo version. We highly recommend you to use a virtual environment. Use following pip command in order to install eGo:

```
$ pip3 install eGo --process-dependency-links
```

1.2.1 Using virtual environment

Firstly, you create a virtual environment and activate it:

```
$ virtualenv venv --clear -p python3.5
$ source venv/bin/activate
$ cd venv
```

Inside your virtual environment you can install eGo with the pip command.

1.2.2 Linux and Ubuntu

The package eGo is tested with Ubuntu 16.04 and 18.04 inside the virtual environments of virtualenv. The installation is shown above.

1.2.3 Windows or Mac OSX users

For Windows and/or Mac OSX user we highly recommend to install and use Anaconda for your Python3 installation. First install anaconda inclusing python 3.5 or higher version from https://www.anaconda.com/download/ and open an anaconda prompt as administrator and run:

```
$ conda install pip
$ conda config --add channels conda-forge
$ conda install shapely
$ pip3 install eGo --process-dependency-links
```

The full documentation can be found on this page. We use Anaconda with an own environment in order to reduce problems with packages and different versions on our system. Learn more about Anacona environments.

1.2.4 Setup database connection

The package `ego.io` gives you a python SQL-Alchemy representations of the **OpenEnergy-Database** (oedb) and access to it by using the oedialect a SQL-Alchemy binding Python package for the REST-API used by the OpenEnergy Platform (OEP). Your API access / login data will be saved in the folder `.egoio` in the file `config.ini`. You can create a new account on openenergy-platform.org/login.

oedialect connection

```
[oedb]
dialect  = oedialect
username = <username>
database = oedb
host     = openenergy-platform.org
port     = 80
password = <token>
```

Local database connection

```
[local]
username = YourOEDBUserName
database = YourLocalDatabaseName
host = localhost or 127.0.0.1
port = 5433
pw = YourLocalPassword
```

Old developer connection

```
[oedb]
username = YourOEDBUserName
database = oedb
host = oe2.iws.cs.ovgu.de
port = 5432
pw = YourOEDBPassword
```

1.3 Getting started

1.3.1 How to use eGo?

1. Check and prepare your eGo setting in `ego/scenario_setting.json`
2. Start your calculation with predefined results tools and run under `eGo/ego` the main file with `>>> python3 ego_main.py`

```
>>> python3 ego_main.py
>>> ...
>>> INFO:ego:Start calculation
>>> ...
```

1.3.2 Examples

```
# import the eGo tool
from ego.tools.io import eGo

# Run your scenario
ego = eGo(jsonpath='scenario_setting.json')

# Analyse your results on extra high voltage level (etrago)
ego.etrago_line_loading()
```

1.3.3 Example Cluster of Germany

1.4 Theoretical background

Contents

1.4.1 Models overview

1.4.2 eTraGo's background

Learn more aboute eTraGo's theoretical background of methods and assumptions here.

1.4.3 eDisGo Cluster methods

1.4.4 Economic calculation

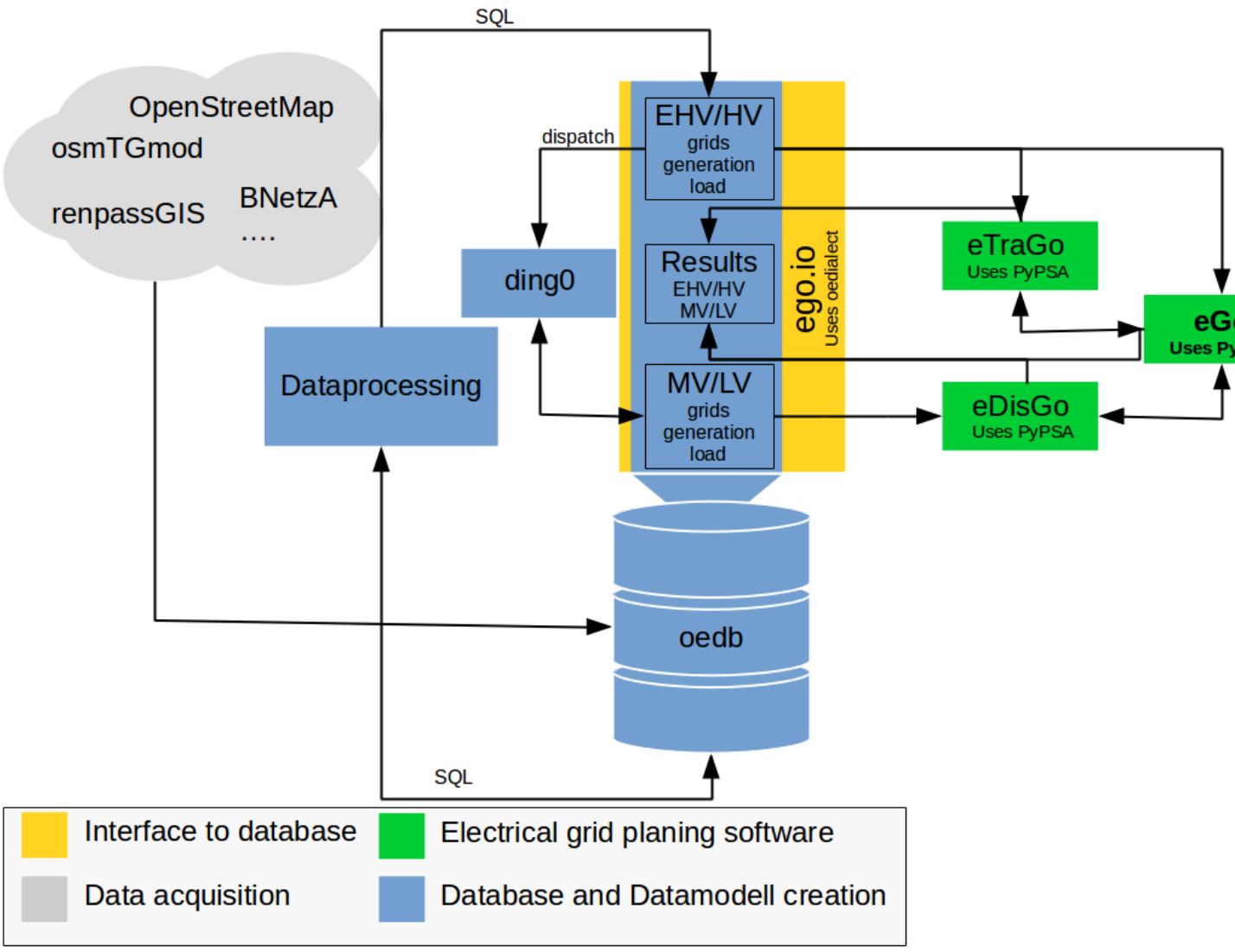
1.4.5 References

1.5 Developer notes

1.5.1 Installation

Note: Installation is only tested on (Ubuntu 16.04) linux OS.

Please read the Installation Guideline ego.doc.installation.


SQL
OpenStreetMap
osmTGmod
renpassGIS
BNetzA
....
dispatch
EHV/HV
grids
generation
load
Results
EHV/HV
MV/LV
MV/LV
grids
generation
load
ding0
Dataprocessing
ego.io
Uses oedialect
eTraGo
Uses PyPSA
eDisGo
Uses PyPSA
oedb
SQL
Interface to database
Electrical grid planing software
Data acquisition
Database and Datamodell creation

1. Use virtual environment

Create a virtualenvironment and activate it:

```
$ virtualenv --clear -p python3.5  ego_dev``
$ cd ego_dev/
$ source bin/activate
```

2. Get eGo

Clone eGo from github.com by running the following command in your terminal:

```
$ git clone https://github.com/openego/eGo
```

With your activated environment *cd* to the cloned directory and run `pip3 install -e eGo --process-dependency-links` . This will install all needed packages into your environment.

3. Get your Database login data

Learn more here.

4. Create Dingo grids

Install ding0 from github.com and run the `example_parallel_multiple_grid_districts.py` script, which can be found under `ding0/ding0/examples/`.

```
$ git clone https://github.com/openego/ding0.git
$ pip3 install -e ding0
$ python3 ding0/ding0/examples/example_parallel_multiple_grid_districts.py
```

Learn more about Dingo. Before you run the script check also the configs of Dingo and eDisGo in order to use the right database version. You finde this files unter `ding0/ding0/config/config_db_tables.cfg` and `~.edisgo/config/config_db_tables.cfg`. Your created ding0 grids are stored in `~.ding0/...`

eDisGo and eTraGo

Please read the Developer notes of eDisGo and eTraGo.

1.6 What's new

See what's new as per this release!

Releases

- *Release v0.3.0 (September 07, 2018)*
- *Release v0.2.0 (July 18, 2018)*
- *Release v0.1.0 (March 29, 2018)*
- *Release v0.0.1 (February 02, 2018)*

1.6.1 Release v0.3.0 (September 07, 2018)

Power Flow and Clustering. eGo is now using eTraGo non-linear power flows based on optimization results and its disaggregation of clustered results to an original spatial complexities. With the release of eDisGo speed-up options, a new storage integration methodology and more are now available.

Added features

- Update of dependencies
- Implementing of Ding0 grid parallelization
- Redesign of scenario settings and API simplifications
- Adding and using the Power Flow of eTraGo in eGo
- Testing and using new dataprocessing Version v0.4.3, v0.4.4 and v0.4.5
- make eGo installable from pip via `pip3 install eGo -- process-dependency-links`
- Implementing eDisGo's storage distribution for MV and LV grids
- Improved logging and the creation of status files
- Maximal calculation time for ding0 grids can be set by user
- eDisGo results import and export (all eGo-relevant data from eDisGo can be re-imported after a run)
- Storage-related investment costs are also allocated to MV grids
- Update of cluster plots
- Plot of investment costs per line and bus
- Update of `ego.iplot` for an interactive visualization

1.6.2 Release v0.2.0 (July 18, 2018)

Fundamental structural changes of the eGo tool are included in this release. A new feature is the integration of the MV grid power flow simulations, performed by the tool eDisGo.. Thereby, eGo can be used to perform power flow simulations and optimizations for EHV, HV (*eTraGo*) and MV (*eDisGo*) grids.

Moreover, the use of the Dataprocessing versions `''v0.4.1''` and `''v0.4.2''` is supported. Please note, that this release is still under construction and only recommended for developers of the *open_eGo* project.

Furthermore, overall cost aggregation functions are available.

Added features

- **Cleaned and restructured eGo classes and functions**
  - Move classes of eGo from results.py to io.py
  - Move serveral function
- **Introduce new files for *eDisGo* handling**
  - edisgo_integration.py
  - mv_cluster.py
- Introduce new file storages.py for eTraGo

- Updated eTraGo 0.6 and integrated eTraGo's new functions and features to eGo
- Updated eDisGo 0.0.3 version which includes the features of a parallelization for custom function and other important API changes.
- Started to implemented pep8 style to eGo Code
- Implemented logging function for the whole model
- Using the Restfull-API for the OpenEnergy Database connection, buy using ego.io v0.4.2. A registration is needed and can be done on openenergy-platform.org/login
- Remove functionalities from `ego_main.py` to the eGo class
- Fixed eTraGo scenario import of `etrago_from_oedb()`

Notes

- As an external user you need to have an account on the openenergy-platform.org/login
- In future versions, all MV grids (*ding0* grids) will be queried from your database. However, in this version all MV grids have to be generated with the tool ding0 and stored in *eGo*'s *data* folder.
- Total operational costs are missing in this release

1.6.3 Release v0.1.0 (March 29, 2018)

As this is the second release of eGo. This Release introduce the results class and is still under construction and not ready for a normal use.

Added features

- Update of interface between eTraGo and eDisGo (specs)
- New structure of eGo module / resulte class
- Restructuring of functions
- Add import function of eTraGo results form oedb

Notes

- The 'direct_specs' function is not working and needs to be set to `false`

1.6.4 Release v0.0.1 (February 02, 2018)

As this is the first release of eGo. The tool eGo use the Python3 Packages eTraGo (Optimization of flexibility options for transmission grids based on PyPSA) and eDisGo (Optimization of flexibility options and grid expansion for distribution grids based on PyPSA) for an electrical power calculation from extra high voltage to selected low voltage level.

Added features

- Interface between eTraGo and eDisGo
- Plots with folium
- First result structure

1.7 eGo

1.7.1 Overview of modules

ego.tools package

Submodules

ego.tools.economics

This module collects useful functions for economic calculation of eGo which can mainly distinguished in operational and investment costs.

ego.tools.economics.**annuity_per_period**(*capex*, *n*, *wacc*, *t*, *p*)
Calculate per given period

Parameters

- **capex** (*float*) – Capital expenditure (NPV of investment)
- **n** (*int*) – Number of years that the investment is used (economic lifetime)
- **wacc** (*float*) – Weighted average cost of capital
- **t** (*int*) – Timesteps in hours
- **p** (*float*) – interest rate

ego.tools.economics.**edisgo_convert_capital_costs**(*overnight_cost*, *t*, *p*, *json_file*)
Get scenario and calculation specific annuity cost by given capital costs and lifetime.

Parameters

- **json_file** (*:obj:dict*) – Dictionary of the scenario_setting.json file
- **_start_snapshot** (*int*) – Start point of calculation from scenario_setting.json file
- **_end_snapshot** (*int*) – End point of calculation from scenario_setting.json file
- **_p** (*numeric*) – interest rate of investment
- **_t** (*int*) – lifetime of investment

Returns **annuity_cost** – Scenario and calculation specific annuity cost by given capital costs and lifetime

Return type numeric

Examples

$$PVA = (1/p) - (1/(p * (1+p)^t))$$

ego.tools.economics.**etrago_convert_overnight_cost**(*annuity_cost*, *json_file*, *t=40*, *p=0.05*)

Get annuity cost of simulation and calculation total `overnight_costs` by given capital costs and lifetime.

Parameters

- **json_file** (*:obj:dict*) – Dictionary of the `scenario_setting.json` file
- **_start_snapshot** (*int*) – Start point of calculation from `scenario_setting.json` file
- **_end_snapshot** (*int*) – End point of calculation from `scenario_setting.json` file
- **_p** (*numeric*) – interest rate of investment
- **_T** (*int*) – lifetime of investment

Returns **overnight_cost** – Scenario and calculation total `overnight_costs` by given annuity capital costs and lifetime.

Return type numeric

Examples

$$PVA = (1/p) - (1/(p * (1+p)^t))$$
$$K_{ON} = K_a * PVA * ((t/(period+1))$$

ego.tools.economics.**etrago_operating_costs**(*network*)

Function to get all operating costs of eTraGo.

Parameters **network_etrago** (`etrago.tools.io.NetworkScenario`) – eTraGo network object compiled by `etrago.appl.etrago()`

Returns **operating_costs** – DataFrame with aggregate operational costs per component and voltage level in [EUR] per calculated time steps.

Return type pandas.Dataframe

Example

```
>>> from ego.tools.io import eGo
>>> ego = eGo(jsonpath='scenario_setting.json')
>>> ego.etrago.operating_costs
```

component	operation_costs	voltage_level
biomass	27.0	ehv
line losses	0.0	ehv
wind_onshore	0.0	ehv

ego.tools.economics.**etrago_grid_investment**(*network*, *json_file*)
Function to get grid expantion costs from eTraGo

Parameters

- **network_etrago** (etrago.tools.io.NetworkScenario) – eTraGo network object compiled by etrago.appl.etrago()
- **json_file** (*:obj:dict*) – Dictionary of the scenario_setting.json file

Returns **grid_investment_costs** – Dataframe with voltage_level, number_of_expansion and capital_cost per calculated time steps

Return type pandas.Dataframe

Example

```
>>> from ego.tools.io import eGo
>>> ego = eGo(jsonpath='scenario_setting.json')
>>> ego.etrago.grid_investment_costs
```

voltage_level	number_of_expansion	capital_cost
ehv	27.0	31514.1305
hv	0.0	0.0

ego.tools.economics.**edisgo_grid_investment**(*edisgo*, *json_file*)
Function aggregates all costs, based on all calculated eDisGo grids and their weightings :param edisgo: Contains multiple eDisGo networks :type edisgo: ego.tools.edisgo_integration.EDisGoNetworks

Returns Dataframe containing annuity costs per voltage level

Return type None or pandas.DataFrame

ego.tools.economics.**get_generator_investment**(*network*, *scn_name*)
Get investment costs per carrier/ generator.

ego.tools.edisgo_integration

ego.tools.io

This file contains the eGo main class as well as input & output functions of eGo in order to build the eGo application container.

class ego.tools.io.**egoBasic**(**args*, ***kwargs*)
Bases: object

The eGo basic class select and creates based on your scenario_setting.json file your definded eTraGo and eDisGo results container. And contains the session for the database connection.

Parameters **jsonpath** (json) – Path to scenario_setting.json file.

Returns

- **json_file** (*:obj:dict*) – Dictionary of the scenario_setting.json file
- **session** (sqlalchemy.orm.session.Session) – SQLAlchemy session to the OEDB

class ego.tools.io.**eTraGoResults**(**args*, ***kwargs*)
Bases: *ego.tools.io.egoBasic*

The eTraGoResults class creates and contains all results of eTraGo and it's network container for eGo.

Returns

- **network_etrago** (etrago.tools.io.NetworkScenario) – eTraGo network object compiled by etrago.appl.etrago()
- **etrago** (pandas.Dataframe) – DataFrame which collects several eTraGo results

class ego.tools.io.**eDisGoResults**(**args*, ***kwargs*)
Bases: *ego.tools.io.eTraGoResults*

The eDisGoResults class create and contains all results of eDisGo and its network containers.

edisgo
Contains basic informations about eDisGo

Returns

Return type pandas.DataFrame

class ego.tools.io.**eGo**(*jsonpath*, **args*, ***kwargs*)
Bases: *ego.tools.io.eDisGoResults*

Main eGo module which includs all results and main functionalities.

Returns

- **network_etrago** (etrago.tools.io.NetworkScenario) – eTraGo network object compiled by etrago.appl.etrago()
- **edisgo.network** (ego.tools.edisgo_integration.EDisGoNetworks) – Contains multiple eDisGo networks
- **edisgo** (pandas.Dataframe) – aggregated results of eDisGo
- **etrago** (pandas.Dataframe) – aggregated results of eTraGo

total_investment_costs
Contains all investment informations about eGo

Returns

Return type pandas.DataFrame

total_operation_costs
Contains all operation costs information about eGo

Returns

Return type pandas.DataFrame

plot_total_investment_costs(*filename=None*, *display=False*, ***kwargs*)
Plot total investment costs

plot_power_price(*filename=None*, *display=False*)
Plot power prices per carrier of calculation

plot_storage_usage(*filename=None*, *display=False*)
Plot storage usage by charge and discharge

plot_edisgo_cluster(*filename=None*, *display=False*, ***kwargs*)
Plot the Clustering of selected Dingo networks

plot_line_expansion(***kwargs*)
Plot line expantion per line

plot_storage_expansion(***kwargs*)
Plot storage expantion per bus

iplot
Get iplot of results as html

ego.tools.io.**results_to_excel**(*ego*)
Wirte results of ego.total_investment_costs to an excel file

ego.tools.io.**etrago_from_oedb**(*session*, *json_file*)
Function which import eTraGo results for the Database by the `result_id` number.

Parameters

- **session** (sqlalchemy.orm.session.Session) – SQLAlchemy session to the OEDB
- **json_file** (`dict`) – Dictionary of the `scenario_setting.json` file

Returns **network_etrago** – eTraGo network object compiled by `etrago.appl.etrago()`

Return type `etrago.tools.io.NetworkScenario`

ego.tools.io.**recover_resultsettings**(*session*, *json_file*, *orm_meta*, *result_id*)
Recover scenario_setting from database

ego.tools.mv_cluster

This file contains all functions regarding the clustering of MV grids

ego.tools.mv_cluster.**analyze_attributes**(*ding0_files*)
Calculates the attributes wind and solar capacity and farthest node for all files in ding0_files. Results are written to ding0_files

Parameters **ding0_files** (`str`) – Path to ding0 files

ego.tools.mv_cluster.**cluster_mv_grids**(*no_grids*, *cluster_base*)
Clusters the MV grids based on the attributes, for a given number of MV grids

Parameters

- **ding0_files** (`str`) – Path to ding0 files
- **no_grids** (*int*) – Desired number of clusters (of MV grids)

Returns Dataframe containing the clustered MV grids and their weightings

Return type pandas.DataFrame

ego.tools.plots

Module which collects useful functions for plotting eTraGo, eDisGo and eGo results.

ego.tools.plots.**carriers_colore**()
Return matplotlib colore set per carrier (technologies of generators) of eTraGo.

Returns **colors** – List of carriers and matplotlib colores

Return type `dict`

ego.tools.plots.**ego_colore**()
Get the four eGo colores

Returns **colors** – List of eGo matplotlib colores

Return type dict

ego.tools.plots.**plot_storage_expansion**(*ego*, *filename=None*, *dpi=300*, *column='overnight_costs'*, *scaling=1*)
Plot line expantion

Parameters

- **ego** (*ego.tools.io.eGo*) – eGo eGo inclueds on eTraGo and eDisGo
- **filename** (*list*) – Filename and/or path of location to store graphic
- **dpi** (*int*) – dpi value of graphic
- **column** (*str*) – column name of eTraGo's line costs. Default: overnight_costs in EURO. Also available s_nom_expansion in MVA or annualized investment_costs in EURO
- **scaling** (*numeric*) – Factor to scale storage size of bus_sizes

Returns https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show

Return type plot matplotlib.pyplot.show

ego.tools.plots.**plot_line_expansion**(*ego*, *filename=None*, *dpi=300*, *column='overnight_costs'*)
Plot line expantion

Parameters

- **ego** (*ego.tools.io.eGo*) – eGo eGo inclueds on eTraGo and eDisGo
- **filename** (*list*) – Filename and or path of location to store graphic
- **dpi** (*int*) – dpi value of graphic
- **column** (*str*) – column name of eTraGo's line costs. Default: overnight_costs in EURO. Also available s_nom_expansion in MVA or annualized investment_costs in EURO

Returns https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show

Return type plot matplotlib.pyplot.show

ego.tools.plots.**plot_grid_storage_investment**(*costs_df*, *filename*, *display*, *var=None*)

ego.tools.plots.**power_price_plot**(*ego*, *filename*, *display*)
Plot power price of calculated scenario of timesteps and carrier

:param ego ego.io.eGo:

Returns https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show

Return type plot matplotlib.pyplot.show

ego.tools.plots.**plot_storage_use**(*ego*, *filename*, *display*)
Plot storage use by charge and discharge values

:param ego ego.io.eGo:

Returns https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show

Return type plot matplotlib.pyplot.show

ego.tools.plots.**get_country**(*session*, *region=None*)
Get Geometries of scenario Countries

ego.tools.plots.**prepareGD**(*session*, *subst_id=None*, *version=None*)
Get MV grid districts for plotting

ego.tools.plots.**plot_edisgo_cluster**(*ego, filename, region=['DE'], display=False, dpi=600*)
Plot the Clustering of selected Dingo networks

:param ego `ego.io.eGo`: self class object of eGo() :param filename: file name for plot `cluster_plot.pdf` :type filename: str :param region: List of background countries e.g. ['DE', 'DK'] :type region: list :param display: True show plot false print plot as `filename` :type display: boolean

Returns https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show

Return type plot `matplotlib.pyplot.show`

ego.tools.plots.**igeoplot**(*ego*, *tiles=None*, *geoloc=None*, *args=None*)
Plot function in order to display eGo results on leaflet OSM map. This function will open the results in your main web browser

:param network_etrago:: class: eTraGo network object compiled by: meth: *etrago.appl.etrago* :type network_etrago:: class: *etrago.tools.io.NetworkScenario* :param tiles: Folium background map style *None* as OSM or *Nasa* :type tiles: str :param geoloc: Listwhich define center of map as (lon, lat) :type geoloc: : obj: *list*

Returns **plot** – HTML file with .js plot

Return type html

ego.tools.plots.**colormapper_lines**(*colormap*, *lines*, *line*, *column='s_nom'*)
Colore Map for lines

ego.tools.results

This module include the results functions for analyze and creating results based on eTraGo or eDisGo for eGo.

ego.tools.results.**create_etrago_results**(*network*, *scn_name*)
Create eTraGo results

Parameters

- **network** (`NetworkScenario`) – eTraGo `NetworkScenario` based on PyPSA Network. See also pypsa.network
- **scn_name** (*str*) – Name of used scenario

Returns **generator** – Result of generator as DataFrame in `ego.etrago.generator`

Return type pandas.DataFrame

ego.tools.results.**results_per_voltage**(*network*)
Get eTraGo results per voltage level

Parameters **network** (`etrago.tools.io.NetworkScenario`) – eTraGo `NetworkScenario` based on PyPSA Network. See also pypsa.network

ego.tools.results.**ego_results_to_oedb**(*ego*)
Function to upload results into oedb database

ego.tools.specs

This files contains all eGo interface functions

ego.tools.specs.**get_etragospecs_direct**(*session*, *bus_id*, *etrago_network*, *scn_name*, *grid_version*, *pf_post_lopf*, *max_cos_phi_renewable*)

Reads eTraGo Results from Database and returns and returns the interface values as a dictionary of corresponding dataframes

Parameters

- **session** (*sqlalchemy.orm.session.Session*) – Handles conversations with the database.
- **bus_id** (*int*) – ID of the corresponding HV bus
- **etrago_network** (etrago.tools.io.NetworkScenario) – eTraGo network object compiled by `etrago.appl.etrago()`
- **scn_name** (*str*) – Name of used scenario 'Status Quo', 'NEP 2035' or 'eGo 100'

Returns Dataframes used as eDisGo inputs

Return type `dict` of pandas.DataFrame

ego.tools.storages

This module contains functions for storage units.

ego.tools.storages.**etrago_storages**(*network*)

Sum up the pysical storage values of the total scenario based on eTraGo results.

Parameters **network** (etrago.tools.io.NetworkScenario) – eTraGo `NetworkScenario` based on PyPSA Network. See also pypsa.network

Returns **results** – Summarize and returns a `DataFrame` of the storage optimaziation.

Return type pandas.DataFrame

Notes

The `results` dataframe incluedes following parameters:

charge [numeric] Quantity of charged energy in MWh over scenario time steps

discharge [numeric] Quantity of discharged energy in MWh over scenario time steps

count [int] Number of storage units

p_nom_o_sum: numeric Sum of optimal installed power capacity

ego.tools.storages.**etrago_storages_investment**(*network*, *json_file*)

Calculate storage investment costs of eTraGo

Parameters **network** (etrago.tools.io.NetworkScenario) – eTraGo `NetworkScenario` based on PyPSA Network. See also pypsa.network

Returns **storage_costs** – Storage costs of selected snapshots in [EUR]

Return type numeric

ego.tools.utilities

This module contains utility functions for the eGo application.

ego.tools.utilities.**define_logging**(*name*)
Helps to log your modeling process with eGo and defines all settings.

Parameters **log_name** (*str*) – Name of log file. Default: ego.log.

Returns **logger** – Set up logger object of package logging

Return type logging.basicConfig.

ego.tools.utilities.**get_scenario_setting**(*jsonpath='scenario_setting.json'*)
Get and open json file with scenaio settings of eGo. The settings incluede eGo, eTraGo and eDisGo specific settings of arguments and parameters for a reproducible calculation.

Parameters **json_file** (*str*) – Default: scenario_setting.json Name of scenario setting json file

Returns **json_file** – Dictionary of json file

Return type dict

ego.tools.utilities.**fix_leading_separator**(*csv_file*, ***kwargs*)
Takes the path to a csv-file. If the first line of this file has a leading separator in its header, this field is deleted. If this is done the second field of every row is removed, too.

ego.tools.utilities.**get_time_steps**(*json_file*)
Get time step of calculation by scenario settings.

Parameters **json_file** (dict) – Dictionary of the scenario_setting.json file

Returns **time_step** – Number of timesteps of the calculation.

Return type int

1.7.2 scenario_settings.json

With the scenario_settings.json file you set up your calcualtion. The file can be found on github.

scenario_setting.json
This file contains all input settings for the eGo tool.

Object Properties

- **global** (*global*) – Global settings that are valid for both eTraGo and eDisGo
- **eTraGo** (*eTraGo*) – eTraGo settings, only valid for eTraGo run
- **eDisGo** (*eDisGo*) – eDisGo settings, only valid for eDisGo runs

global

Object Properties

- **eTraGo** (*bool*) – Decide if you want to run the eTraGo tool (HV/EHV grid optimization).
- **eDisGo** (*bool*) – Decide if you want to run the eDisGo tool (MV grid optimiztaion).
- **db** (*string*) – Name of your database (e.g.‘‘’‘oedb’‘‘‘).
- **recover** (*bool*) – If true, (previously calculated) eTraGo results are queried from your database (instead of performing a new run).

- **result_id** (*int*) – ID of the (previeously calculated) eTraGo results that are queried if **recover** is set `true`.
- **gridversion** (*string*) – Version of the *open_eGo* input data-sets (e.g. `''v0.4.2''`)

eTraGo
This section of *scenario_setting.json* contains all input parameters for the eTraGo tool. A description of the parameters can be found here.

Please note that some parameters are already included in *global*

eDisGo
This section of *scenario_setting.json* contains all input parameters for the eDisGo tool and the clustering of MV grids.

Object Properties

- **ding0_files** (*string*) – Relative path to the MV grid files (created by ding0) (e.g. `''data/MV_grids/20180713110719''`)
- **choice_mode** (*string*) – Mode that eGo uses to chose MV grids out of the files in **ding0_files** (e.g. `''manual''`, `''cluster''` or `''all''`). If `''manual''` is chosen, the parameter **manual_grids** must contain a list of the desired grids. If `''cluster''` is chosen, **no_grids** must specify the desired number of clusters. If `''all''` is chosen, all MV grids from **ding0_files** are calculated.
- **manual_grids** (*list*) – List of MV grid ID's (*open_eGo* HV/MV substation ID's)
- **no_grids** (*int*) – Number of MV grid clusters (from all files in **ding0_files**, a specified number of representative clusters is calculated)

1.7.3 ego_main.py

This is the application file for the tool eGo. The application eGo calculates the distribution and transmission grids of eTraGo and eDisGo.

Note: Note, the data source of eGo relies on the Open Energy Database. - The registration for the public accessible API can be found on openenergy-platform.org/login.

Run the `ego_main.py` file with:

```
>>> python3 ego_main.py
>>> ...
>>> INFO:ego:Start calculation
>>> ...
```

The eGo App works like:

```
>>> from ego.tools.io import eGo
>>> ego = eGo(jsonpath='scenario_setting.json')
>>> ego.etrago_line_loading()
>>> print(ego.etrago.storage_costs)
>>> ...
>>> INFO:ego:Start calculation
>>> ...
```

Take also a look into the documentation of eTraGo and eDisGo which are part of eGo.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

R

T