ATTEST

Ref. Ares(2023)7150968 - 20/10/2023

WP3

D3.2 Optimisation Tool for Distribution Network Planning

User guide & test cases

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 864298.

DOCUMENT CONTROL PAGE

DOCUMENT	User guide
TYPE	Report
DISTRIBUTION LEVEL	Public
DUE DELIVERY DATE	30/09/2022
DATE OF DELIVERY	30/09/2022
VERSION	V1.0
DELIVERABLE RESPONSIBLE	University of Manchester
AUTHOR(S)	Dr Andrey Churkin, Dr Wangwei Kong, Dr Eduardo Alejandro Martínez Ceseña, Prof Pierluigi Mancarella
OFFICIAL REVIEWER(S)	Miguel A. Sanz-Bobi, COMILLAS, Tomislav Capuder, University of Zagreb

DOCUMENT HISTORY

VERSION	AUTHORS	DATE	CHANGES
0.1	Dr Wangwei Kong, Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	05/09/2022	Initial structure of the document containing main contents
0.9	Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	26/09/2022	Detailed description of the tool, examples of input/output and the command line interface, case study of a distribution network
0.95	Dr Miguel A. Sanz-Bobi, Dr Tomislav Capuder, Dajana Vrbičić Tenđera	29/09/2022	Internal review
1.0	Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	30/09/2022	Finalising the document. Incorporating suggestions and modifications from the reviewers.

Table of Contents

List of figures

List of abbreviations and acronyms

AC – Alternating Current

ATTEST – "Advanced Tools Towards cost-efficient decarbonisation of future reliable Energy SysTems" project

CLI – Command-line Interface

DC – Direct Current

DSO – Distribution System Operator

OPF – Optimal Power Flow

TSO – Transmission System Operator

WP – Work Package

1. Aim

This document provides a step-by-step guide to installing and using the Python-based model developed for Deliverable 3.2 “Optimisation tool for distribution network planning” within the Advanced Tools Towards cost-efficient decarbonisation of future reliable Energy SysTems (ATTEST) project. The tool determines optimised investment plans for distribution systems, considering different future energy scenarios and integration of demand-side flexibilities.

2. Introduction

The ATTEST project aims to develop an open-source toolbox comprising a suite of innovative tools to support TSOs / DSOs synergic operation, optimal maintenance of assets, and coordinated planning of both transmission and distribution systems for 2030 and beyond.

This document presents a user guide for the deliverable D3.2 “Optimization tool for distribution network planning”, which includes the installation and execution of the tool. The distribution network planning tool comprises multi-stage stochastic optimisation models that enable exploiting the potential of flexible resources. The stochastic formulation (non-recombining scenario trees) is complemented by a simulation-based optimisation framework to produce adaptive path-dependent network reinforcement strategies. One of the major challenges of distribution network planning is the high computation cost of evaluating a potentially massive number of investments under uncertain future conditions. To tackle this challenge, the proposed distribution network planning tool adopts a recursive algorithm for optimisation. The recursive function considers a reduced search space by terminating all infeasible investment strategies (Martinez Cesena & Mancarella, 2016). Thus, the computation time is minimised compared to a full exhaustive search.

The principles of path-dependent network planning can be visualised as shown in Figure 1. The planning tool receives two scenarios (corresponding to active and slow economies) that form an envelope of possible future system development across multiple years, e.g., 2020, 2030, 2040, and 2050. Then, these extreme scenarios are transformed into a scenario tree, where at each year, possible future trajectories branch into a more active one and a slower one. In this manner, the planning problem represents a portfolio of investment decisions and can capture the effects of non-asset-based solutions, e.g., activating flexibility resources. The recursive function iteratively verifies the feasibility of different investment decisions, starting with the initial year, usually 2020, and moving forward in time. The investments are optimised considering capital expenditures (assets built) and security standards.

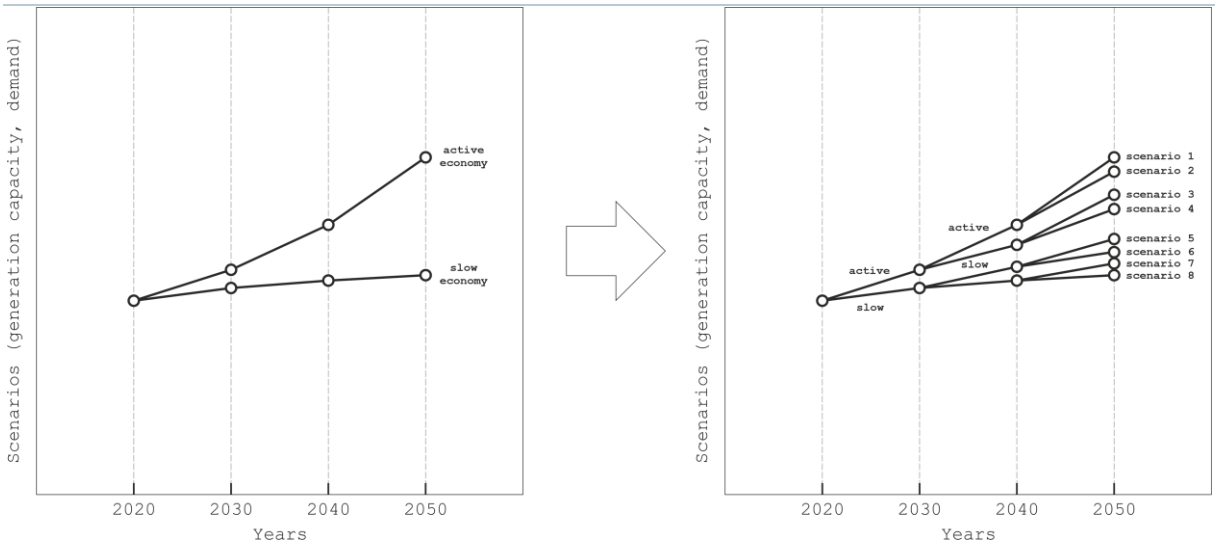

Figure 1. Visualization of the path-dependent network reinforcement: (left) two input scenarios provided by ATTEST test cases corresponding to active and slow economies, (right) scenario tree generated in the distribution planning tool.

However, the formulated multi-year investment approach is a hard combinatorial problem, which becomes computationally infeasible for a large number of possible investments, i.e., when multiple lines can be built with a variety of capacity update options. In this regard, the planning tool first runs a screening model. This model identifies potential interventions using a DC OPF formulation, compares these results with a given investment catalogue, and produces clusters of investments for the planning tool. Therefore, for the sake of computational effectiveness, the planning tool evaluates clusters of investment decisions, i.e., groups of line updates, but not individual line updates.

At a high level, the structure of the planning tool consists of the two main components:

1. *Screening model*
   The screening model is a case of the linearised DC OPF formulation. This model is solved individually for each node of the scenario tree (each combination of scenarios and years). However, the path dependence is not considered in these simulations, i.e., investment decisions in one year do not affect decisions in the upcoming years. Such simplified modelling enables fast identification of potential network updates. The identified investment options are grouped into clusters of investments, which will be used in the second planning stage. The number of clusters produced depends on the number of distinguishable solutions (line capacity updates) in different scenarios and years. This number can reach the total number of nodes in the scenario tree. However, as will be further explained in the document, a special parameter is introduced to limit the maximum number of clusters.
2. *Recursive planning algorithm*
   The planning algorithm iteratively evaluates the feasibility of the identified clusters of investments for all nodes of the scenario tree. An exact AC OPF model is used for the simulations, and the path dependence is introduced to produce a consistent portfolio of investment for different scenarios and capture the effects of flexibility. The algorithm starts with the initial year, for example, 2020, and moves forward in time to identify feasible investments while minimising costs. The output of the algorithm is a multi-year portfolio of optimal investments for the two given scenarios, active and slow economies.

A more detailed description of the proposed tool can be found in (Kong, et al., 2021). The remainder of this document is structured as follows. Sections 3 and 4 provide a detailed description of the steps needed to install the planning tool. Then, Section 5 introduces the main commands and options for interacting with the tool and launching the planning model. The format and location of input data are discussed in Section 6. Finally, Section 7 demonstrates the performance of the tool for an illustrative planning problem, a 3-bus case study. Description of this case study is given in the Appendix.

A test case from the ATTEST data base is shown as an example for executing the developed distribution network planning tool.

3. Getting started

Note that within ATTEST project, all developed tools are being accessed through a platform developed within WP6. Thus, users would not necessarily need to go through the installation of Tool 3.1. However, this document provides a complete guide for users who are willing to use the open-source distribution network planning tool solely.

The following components are required to successfully run the tool, including a Python compiler and a Command Line Interface (CLI), also known as console or command prompt:

1. Python compiler – The model was developed using Spyder and VScode. The former is provided by Anaconda3:
   https://conda.io/docs/user-guide/install/index.html
2. CLI – Anaconda also provides a CLI (i.e., Anaconda Prompt). To access the CLI in in windows, go to:
   Start –> All Programs –> Anaconda3 –> Anaconda Prompt
3. Git – Git is a free and open-source distributed version control system. Available at: https://git-scm.com/downloads
4. The Python model – The Python model is a set of scripts and programs that can be cloned to the user's computer from the Git repository.

Typical questions:

Why is the Python compiler needed?

The Python compiler allows to edit the model and create new case studies.

Why is a CLI needed?

The CLI allows to install, run and update the model. These applications will be illustrated below.

4. Tool Installation

1. Request access rights to the ATTEST Tool 3.1 online repository on GitHub using the following link:

   https://github.com/jnmelchorg/pyensys

   Note that this Git repository, called Pyensys, includes tools developed by the University of Manchester for different projects, including ATTEST. Therefore, if cloning Pyensys, a user will get of copy of multiple tools, some of which are not relevant to distribution network planning. Nonetheless, cloning these tools together to a user's computer does not cause any conflicts or errors. Users can clone this entire repository and use only the tools and commands designed for distribution network planning.

2. Install and open a CLI, as shown in Figure 2. Alternatively, users can launch the tool via Command Prompt (cmd.exe), as shown in Figure 3.

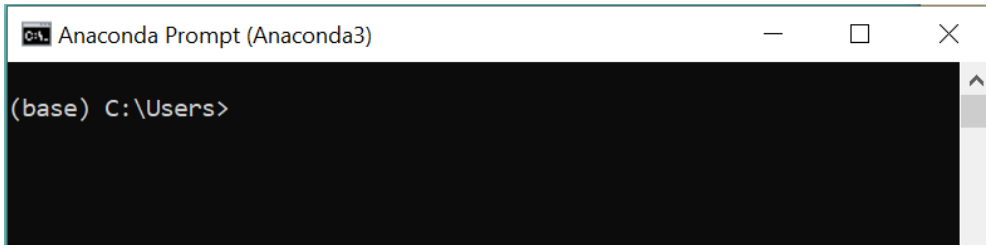

Figure 2. Example of CLI using Anaconda Prompt

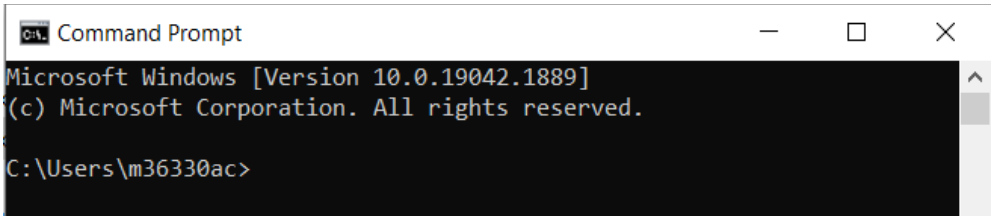

Figure 3. Example of CLI using Windows Command Prompt

3. Navigate, using the CLI to the location where the ATTEST Tool 3.1 model should be installed. The change directory, 'cd', command can be used:

   cd C:\ Location_of_software\ ATTEST_Tool3.1

   Note that the command 'dir' can be used to see the folders available, and the command 'cd ..' can be used to go back one folder:

   dir

   cd ..

4. Now, clone the tool from the Git repository to the computer using the "Git clone" command:

   Git clone https://github.com/jnmelchorg/pyensys.git

The models will be automatically copied to the specified directory, and version control will be activated via GitHub functionalities.

5. Use console to go into the folder where the model is located.

cd C:\ Location_of_software\ ATTEST_Tool3.1

6. The console will now display the name of the current branch, typically the master branch. The branches are different versions of the model that allow users to modify the model and track changes while keeping the original version of the model (master version) available. To see all available branches, use the command:

git branch -a

To access a specific branch, use the following command:

git checkout branch_name

7. Before installing the tool, users can select a virtual environment in Python. A list of existing environments can be called by:

conda env list

A new environment for testing the tools can be created with the command:

conda create -n new_enviroment_name Python=3.9

Then, the environment can be activated by:

conda activate new_enviroment_name

8. Now, when the Pyensys models have been cloned to the computer and a Python environment has been selected, it is time to install the planning tool. The installation can be initiated using the following command:

Python setup.py install

Note that such installation will not allow users to modify the codes of the tool. If needed, users can initiate installation in the development mode using the command:

pip install -e .

9. At this stage, the model has been installed. However, the model also requires several Python packages that may not yet be installed. In this case, when running a Python model, the console will display an error stating that a package was not found. For example, if the console is displaying that the 'pyomo' package is missing, this package has to be installed before launching the tool. The most common package installation commands are 'pip install' and "conda install":

pip install pyomo

conda install pyomo

To check all the Python package versions, use the command:

pip list

conda list

10. Below is the list of required Python packages:
    a. click==8.1.3
    b. Cython==0.29.32
    c. networkx==2.8.5
    d. numpy==1.23.1
    e. openpyxl==3.0.10
    f. pandapower==2.10.1
    g. pandas==1.4.3
    h. Pyomo==6.4.1
    i. pyparsing==3.0.9
    j. pypsa==0.20.0
    k. tables==3.7.0

5. Tool execution

1. Users can run the tool in the folder "path/ATTEST_Tool3.1" with the command:

Python cli.py

Or simply by calling the Pyensys models:

pyensys

If launched successfully, the tool will print out its interface, displaying available commands and options, as shown in Figure 4.

```
Usage: pyensys [OPTIONS] COMMAND [ARGS]...

  Prepare pyene simulation

Options:
  --init BOOLEAN            Take the settings from __init__
  --hydro INTEGER           Number of hydropower plants
  --profile / --no-profile
  --help                    Show this message and exit.

Commands:
  run
  run-ac              AC power flow
  run-dist_invest     Call ATTEST's distribution network planning tool
  run-dist_path       Call ATTEST's distribution network planning tool
  run-e               Prepare energy balance simulation
  run-en              Prepare energy balance and network simulation
  run-example-hydro   Prepare energy balance and network simulation
  run-n               Prepare electricity network simulation
  run-res             Prepare Network Simulation
  run_pyene           run simulation specified in excel file
  test                Hidden development functionality
```

Figure 4. Example of the tool's interface with available commands and options

Note that only two commands correspond to the ATTEST distribution network planning tool, namely "run-dist_invest" and "run-dist_path". The "run-dist_invest" command, short for "run distribution investment models", launches the complete set of models with full planning functionalities and user's control over the input data. That is, the screening model will be solved for a selected case study to identify clusters of investments. Then, a scenario tree will be generated and the recursive planning model will launched to find path-dependent feasible investment plans. Another command, "run-dist_path", is designed for developers and more advanced users who want to test the investment model for different scenarios and investment options. This command requires a predefined scenario tree with possible investment decisions in JSON format.

2. The investment model can be launched by the command:

pyensys run-dist_invest

Users can read the description of available options by using the "--help" command:

pyensys run-dist_invest --help

An example of the "--help" command output is given in Figure 5.

```
Options:
  --output_dir TEXT         Full path of the outputs. By
                            default:c:\users\m36330ac\documents\mega\eduardo
                            alejandro martinez cesena\wp3\python\from
                            nicolas\pyensys\pyensys\tests\outputs\output.json.
  --case TEXT               Location and name of m file. By
                            defaultc:\users\m36330ac\documents\mega\eduardo
                            alejandro martinez cesena\wp3\python\from
                            nicolas\pyensys\pyensys\tests\matpower\case3.m.
  --line_capacities FLOAT   List of line capacities [MVA]. By default: [0.045,
                            0.075, 0.1125, 0.15, 0.225, 0.3, 0.5, 0.75, 1.0,
                            2.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 80.0,
                            100.0, 250.0, 500.0].
  --TRS_capacities FLOAT    List of transformer capacities [MVA]. By default:
                            [1.0, 2.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0,
                            80.0, 100.0, 250.0, 500.0].
  --line_Costs TEXT         Line costs [£/km]. By default it is an empty list
                            [] and the model assumes 60000*line capacity. The
                            costs can alsobe added here, e.g., using the line
                            capacities above: [1800, 3000, 4500, 6000, 9000,
                            12000, 20000, 30000, 40000, 80000, 200000, 400000,
                            800000, 1200000, 1600000, 2000000, 2400000,
                            3200000, 4000000, 10000000, 20000000].
  --TRS_costs TEXT          Transformer costs [£]. By default it is an empty
                            list [] and the model assumes 7000*transformer
                            capacity. The costs can also be added here, e.g.,
                            using the transformer capacities above: [7000,
                            14000, 35000, 70000, 140000, 210000, 280000,
                            350000, 420000, 560000, 700000, 1750000, 3500000].
  --cont_list TEXT          List of line contingencies. By default is its empty
                            [].
  --line_length TEXT        Length of each branch [km], by default it is empty
                            [] and the model assumes all lines are 1km long
                            i.e.,[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                            1, 1, 1, 1, 1, 1].
  --growth TEXT             Dictionary with demand growth [%] for selected
                            years (e.g., 2020, 2030 and 2040) and scenarios
                            (e.g., active and slow). The first year is taken as
                            the current year and should be assign a growth of
                            0%. By default: {'Active': {'2020': 0, '2030':
                            1.89, '2040': 3.0},'Slow': {'2020': 0, '2030': 1.1,
                            '2040': 2.0}}.
  --DSR TEXT                Dictionary with DSR [%] for selected years (e.g.,
                            2020, 2030 and 2040) and scenarios (e.g., active
                            and slow). The format of the information must match
                            --growth. By default: {'Active': {'2020': 0,
                            '2030': 0.05, '2040': 0.05}, 'Slow': {'2020': 0,
                            '2030': 0.02, '2040': 0.02}}.
  --cluster TEXT            List of investment clusters [MVA]. By default it is
                            set to None and the model will calculate the
                            adequate investment options.
  --oversize INTEGER        Option to intentionally oversize investments by
                            selecting the next available value from the
                            --line_capacities. By default it is set to 0 (no
                            oversize).
  --Max_clusters INTEGER    Constraint on the maximum number of clusters
                            considered. By default it is set to 5.
  --scenarios TEXT          List of scenarios to model. By default it is left
                            empty [] and the model will consider all available
                            scenarios, e.g., [0, 1, 2, 3].
  --help                    Show this message and exit.
```

Figure 5. Example of the help information for the investment planning tool

Most of the options specify the input data for the planning model are self-explanatory (considering the information in the help menu), for example:

- "output_dir" sets the path for the output directory where results of the planning tool will be saved;
- "case" specifies the location of the MATPOWER file for the case study;

- "line_capacities" and "TRS_capacities" are the lists of available investment options for lines and transformers, given in MVA;
- "line_Costs" and "TRS_costs" indicate the costs of the investment options for lines and transformers;
- "growth" defines the load growth forecast for each year, for two scenarios corresponding to active and slow economies.
- "DSR" introduces the levels of flexibility for future years as a percentage of the total load.

However, a few options might need additional clarification. The "*Max_clusters*" option sets an important parameter for the screening model, limiting the number of investment clusters prepared for the recursive investment model. It is not recommended to increase this number significantly due to the computationally demanding combinatorial problem. An empirical limit for this parameter is about 10 clusters. Higher number of clusters will result in the recursive calculation of millions of OPF problems, which is not practical. The "*cont_list*" parameter allows users to introduce contingencies into the planning problem. However, most of the ATTEST case studies represent radial distribution networks with no feasible contingencies. Therefore, this list is empty by default. The "*line_length*" parameter represents assumptions of the lines' length in a distribution network. This parameter affects the costs of investments and can be tuned by users. The default length is assumed 1 km for each line. Finally, the "*oversize*" option can be used to select higher capacity investments than the optimal ones (the next available option will be selected for each line from the investment catalogue). This option might be needed for producing robust investment strategies and avoiding potential congestion and voltage issues.

6. Input and output data

1. Input data:
   Note that when calling the "run-dist_invest" command, the specified case study (.m file) has to be placed in a location specified by the user. If the location is not specified, the model will look for the file in:

   …pyensys\pyensys\tests\matpower

   Similarly, when running the "run-dist_path" command, if a path is not specified, the model will look for the input (JSON file) in the following location:

   …pyensys\pyensys\tests\json

2. Output data:
   The output data of the network planning model includes costs and recommended investments across two extreme scenarios. To facilitate integration with other tools within ATTEST, the format of the outputs file is exactly the same as the format of the outputs of the transmission

planning tool in T3.1. The output file can be placed in a location selected by the user but, by default, the files are placed in:

…pyensys\pyensys\tests\outputs\output.json

The format of the information, and the use of the tool, are illustrated below with an example.

7. Test cases

7.1. Network for demonstration: 3-bus test system

The performance of the planning tool and the format of input/output data is demonstrated based on a simple 3-bus test system. The system is described in the "case3.m" file. Network data can also be found in "Appendix A: case3.m data" of this deliverable. Topology of the network is illustrated in Figure 6.

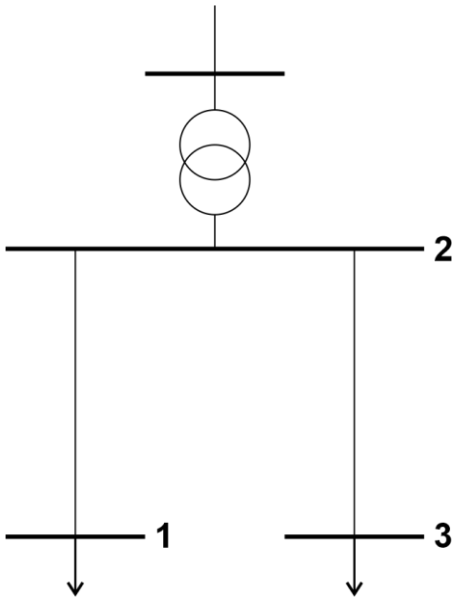

Figure 6. Topology of the 3-bus test system

It is considered that loads of the network will continuously increase in the future according to the default increase multipliers assumption. Three time periods (snapshots of years) are considered in the planning problem: 2020, 2030, and 2050. Only a maximum of three investment clusters are enabled for this simulation. Then, the planning problem is solved by using the "run-dist_invest" command.

First, the screening model is solved. Its interim results (clusters of investments) are stored in "…pyensys\pyensys\tests\json\ATTEST_Inputs.json". These results have the following structure:

```
{
  "problem": {
    "inter-temporal_opf": false,
    "return_rate_in_percentage": 3.0,
    "non_anticipative": true
  },
  "pandapower_mpc_settings": {
    "mat_file_path": "c:\\users\\m36330ac\\documents\\mega\\eduardo alejandro martinez cesena\\wp3\\python\\from nicolas\\pyensys\\pyensys\\tests\\matpower\\case3.m",
    "frequency": 60.0
  },
  "optimisation_profiles_data": {
    "format_data": "attest",
    "data": [
      {
        "group": "buses",
        "data": [
          [
            1,
            2020,
            0,
            10.0,
            5.0
          ],
          [
            1,
            2020,
            2,
            15.0,
            10.0
          ],
          .
          .
          .
          .
          [
            4,
            2040,
            2,
            19.980000000000004,
            13.320000000000004
          ]
        ],
        "columns_names": [
          "scenario",
          "year",
          "bus_index",
          "p_mw",
          "q_mvar"
        ]
      }
```

```
        ]
      },
      "pandapower_optimisation_settings": {
        "display_progress_bar": true,
        "optimisation_software": "pypower"
      },
      "optimisation_binary_variables": [
        {
          "element_type": "line",
          "costs": [
            110000.0,
            160000.0,
            400000.0
          ],
          "elements_positions": [
            [
              1,
              3
            ],
            [
              1,
              3
            ],
            [
              1,
              3
            ]
          ],
          "installation_time": [
            0,
            0,
            0
          ],
          "capacity_to_be_added_MW": [
            [
              2.0,
              0.75
            ],
            [
              2.0,
              2.0
            ],
            [
              5.0,
              5.0
            ]
          ]
        }
      ]
    }
```

The load increase is specified separately for each bus, for every year and scenario, as given in the list “buses, data:”. The investment parameters are listed in “optimisation_binary_variables”, which specifies the positions of lines in the investment clusters, installation time, capacities, and costs of the line updates.

This data is automatically passed to the recursive optimisation model, which starts iteratively verifying the feasibility of pre-selected investment plans for every year and scenario. Once solved, the model will save its output to the file “…pyensys\pyensys\tests\outputs\output.json”. The output has the following structure:


```
{
  "Country": "case3",
  "Case name": "case3",
  "Scenario 1": {
    "Total investment cost (EUR-million)": 0.11,
    "Flexibility investment cost (EUR-million)": 0,
    "Net Present Operation Cost (EUR-million)": 0,
    "2020": {
      "Operation cost (EUR-million/year)": 0,
      "Branch investment (MVA)": [
        0,
        2.0,
        0,
        0.75
      ],
      "Flexibility investment (MW)": [
        0,
        0,
        0,
        0
      ]
    },
    "2030": {
      "Operation cost (EUR-million/year)": 0,
      "Branch investment (MVA)": [
        0,
        0,
        0,
        0
      ],
      "Flexibility investment (MW)": [
        0,
        0,
        0,
        0
      ]
    },
    "2040": {
      "Operation cost (EUR-million/year)": 0,
      "Branch investment (MVA)": [
        0,
        0,
```

```
            0,
            0
          ],
          "Flexibility investment (MW)": [
            0,
            0,
            0,
            0
          ]
        }
      },
      "Scenario 2": {
        "Total investment cost (EUR-million)": 0.11,
        "Flexibility investment cost (EUR-million)": 0,
        "Net Present Operation Cost (EUR-million)": 0,
        "2020": {
          "Operation cost (EUR-million/year)": 0,
          "Branch investment (MVA)": [
            0,
            2.0,
            0,
            0.75
          ],
          "Flexibility investment (MW)": [
            0,
            0,
            0,
            0
          ]
        },
        "2030": {
          "Operation cost (EUR-million/year)": 0,
          "Branch investment (MVA)": [
            0,
            0,
            0,
            0
          ],
          "Flexibility investment (MW)": [
            0,
            0,
            0,
            0
          ]
        },
        "2040": {
          "Operation cost (EUR-million/year)": 0,
          "Branch investment (MVA)": [
            0,
            0,
            0,
```

```
            0
          ],
          "Flexibility investment (MW)": [
            0,
            0,
            0,
            0
          ]
        }
      }
}
```


The information contained in the output is divided into two groups corresponding to “Scenario 1”, active economy, and “Scenario 2”, slow economy. Within each group, users can find detailed results, such as the costs of investments, indices of updated lines, additional capacity built, and flexibility investments.

8. References

Kong, W. et al., 2021. *Optimal Planning Tools for Transmission and Distribution Systems,* s.l.: The University of Manchester.

Martinez Cesena, E. A. & Mancarella, P., 2016. Practical recursive algorithm and flexible open-source applications for planning of smart distribution networks with demand response. *Sustainable Energy Grids and Networks,* Volume 7, pp. 104 - 116.

Appendix A: case3.m data

```
function mpc = case9
%CASE9    AC Optimal Power flow data for 3 bus, 1 generator case.
%   MATPOWER

%% MATPOWER Case Format : Version 2
mpc.version = '2';

%%-----  Power Flow Data  -----%%
%% system MVA base
mpc.baseMVA = 100;

%% bus data
%   bus_i    type    Pd      Qd      Gs      Bs      area    Vm      Va      baseKV  zone
    Vmax    Vmin
mpc.bus = [
   1.0000    2.0000   10.0000    5.0000         0         0    1.0000    1.0000         0  345.0000    1.0000
1.1000    0.9000
```

```
   2.0000   3.0000        0        0        0        0   1.0000   1.0000        0  345.0000   1.0000   1.0000
1.0000
   3.0000   2.0000  15.0000  10.0000        0        0   1.0000   1.0000        0  345.0000   1.0000
1.1000   0.9000
];

%% generator data
%   bus      Pg       Qg              Qmax    Qmin    Vg              mBase   status   Pmax
    Pmin     Pc1      Pc2      Qc1min Qc1max          Qc2min Qc2max          ramp_agc
    ramp_10          ramp_30          ramp_q apf
mpc.gen = [
   2.0000        0        0  30.0000        0   1.0250  100.0000   1.0000   40.0000        0        0        0
0        0        0        0        0        0        0        0        0
];

%% branch data
%   fbus     tbus     r               x               b         rateA    rateB    rateC    ratio
    angle    status   angmin angmax
mpc.branch = [
   1.0000   2.0000   0.1000   0.5000   0.0200  12.0000        0        0        0        0   1.0000 -360.0000
360.0000
   1.0000   2.0000   0.1000   0.5000   0.0200  12.0000        0        0        0        0        0 -360.0000
360.0000
   2.0000   3.0000   0.1000   0.3300   0.0100  19.0000        0        0        0        0   1.0000 -360.0000
360.0000
   2.0000   3.0000   0.1000   0.3300   0.0100  19.0000        0        0        0        0        0 -360.0000
360.0000
];

%%-----  OPF Data  -----%%
%% generator cost data
%   1        startup  shutdown        n        x1       y1       ...       xn       yn
%   2        startup  shutdown        n        c(n-1)   ...      c0
mpc.gencost = [
    2    0    0    2    4    0
];
```