

The sole responsibility for the content published on this document lies with the authors. It does not necessarily reflect the opinion of the Innovation and Networks Executive Agency (INEA) or the European Commission (EC). INEA or the EC are not responsible for any use that may be made of the information contained therein.

WP3

D3.3 Optimisation Tool for Transmission Network Planning

User guide & test cases



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 Wangwei Kong, Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña, Prof Pierluigi Mancarella. The University of Manchester.
OFFICIAL REVIEWER(S)	Dajana Vrbičić Tenđera, HOPS and Tomislav Capuder, University of Zagreb

DOCUMENT HISTORY

VERSION	AUTHORS	DATE	CHANGES
0.1	Dr Wangwei Kong, Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	08/08/2022	Initial structure of the document containing main contents
0.2	Dr Wangwei Kong, Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	19/08/2022	Adding new figures and test case example
0.3	Dr Wangwei Kong, Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	27/08/2022	Adding references, appendices, default data and potential errors and solutions in tool installation
0.4	Dr Wangwei Kong, Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	14/09/2022	Adding missing content and restructuring document
0.9	Dr Andrey Churkin, Dr Eduardo Alejandro Martínez Ceseña	26/09/2022	Additional corrections before sending document to reviewers
0.95	Dajana Vrbičić Tenđera, and Dr Tomislav Capuder	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

1. AIM	5
2. INTRODUCTION	5
3. HIGH-LEVEL DESCRIPTION OF THE TRANSMISSION NETWORK PLANNING TOOL.....	5
4. GETTING STARTED	7
5. TOOL INSTALLATION	8
5.1. Installing the Python components of the tool	8
5.2. Installing the Julia components of the tool.....	9
6. TOOL EXECUTION.....	11
7. INPUT AND OUTPUT DATA.....	14
8. TEST CASE	15
9. BIBLIOGRAPHY	19
APPENDIX A – INSTALLATION INSTRUCTIONS FOR DEVELOPERS	19
APPENDIX B – POTENTIAL INSTALLATION ERRORS	21
APPENDIX C – MODIFIED CASE5.M FILE DATA	23

List of figures

Figure 1: Example of extreme scenarios considered by the planning tool.....	5
Figure 2: Example of scenario tree used by the planning tool	6
Figure 3. Download ZIP file of the tool	8
Figure 4. Example of CLI using Anaconda prompt	8
Figure 5. Julia prompt interface	10
Figure 6. Installing PyJulia through a Python REPL.....	10
Figure 7. Help information with commands of the tool	11
Figure 8. Help information of the screening model of the tool	12
Figure 9. Help information of the investment model of the tool.....	13
Figure 10. Topology of the 5-bus system	16
Figure 11. Example of CLI using Anaconda prompt	20
Figure 12. Download the tool using Git.....	20
Figure 13. Tool download through prompt using git clone	21
Figure 14. Example of windows system Environment Variables.....	22

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

GLPK – “GNU Linear Programming Kit” software

MILP – Mixed-integer linear programming

OPF – Optimal Power Flow

SCACOPF – Security-constrained AC 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 3.3 “Optimisation tool for transmission network planning” within the “Advanced Tools Towards cost-efficient decarbonisation of future reliable Energy SysTems” (ATTEST) project. The tool determines optimised investment plans for transmission system 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 D3.3 “Optimisation tool for transmission network planning”, which includes the installation and execution of the tool. D3.3 is a flexible transmission expansion planning optimisation tool that considers the use of flexibility from emerging bulk generation technologies (e.g., storage) in combination with support from the demand side. D3.3 optimises network investment pathways for transmission networks across multiple future energy scenarios, e.g., 2020, 2030, 2040 and 2050. The investments are optimised considering both capital expenditures (assets built) and associated operation costs based on long-term uncertainties from future energy scenarios. The operation costs can be associated with generation, reliability, and flexibility services procurement.

A functional description of the tool is provided, as well as a general description of the inputs and outputs, computational requirements, and interactions with other tools within ATTEST. The applications of the tool are illustrated with an example.

3. High-level description of the transmission network planning tool

The proposed transmission network planning tool aims to propose investment in assets (e.g., line reinforcements) and non-asset-based solutions (e.g., from flexible sources) considering an uncertain future. Two extreme forecasts for demand growth are considered, the active and slow economy scenarios, as shown in Figure 1.

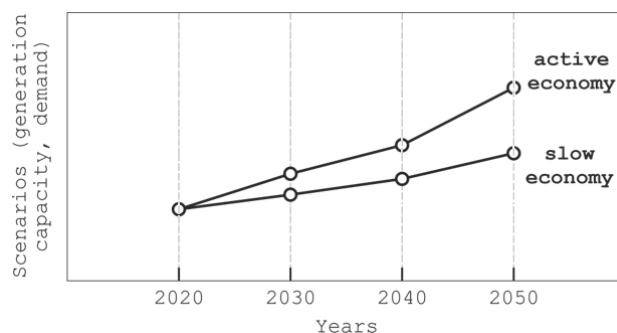


Figure 1: Example of extreme scenarios considered by the planning tool

To meet the abovementioned aim, the tool optimises the proposed investment schemes, i.e., planning investment decisions during selected years, across the scenarios. Taking the example from Figure 1, the investment schemes include investment decisions in the year 2020, and two recommended investment decisions for 2030, 2040 and 2050. For this purpose, the tool was formulated as a multi-stage mathematical programming model, as discussed in detail in (Kong, et al., 2021).

At a high level, the tool divides the problem into a series of targeted subproblems, namely a screening model that identifies feasible investment alternatives, an investment model that focuses on minimising investment costs (part 1), and a second investment model that, besides minimising investment costs, also aims to minimise operation costs (part 2).

1. *Screening model*

Optimising a planning strategy for transmission networks can be computationally expensive considering the potentially massive number of investments available, some of which can become attractive under uncertain future conditions. To mitigate this issue, the planning tool utilises a screening model to identify and pre-select potentially attractive candidate investments.

The screening model is a simplified transmission planning model, a DC OPF model with a linear cost function for network investments. The model is applied to different conditions across scenarios to identify which network components would normally be reinforced. The relevant options are then selected to create an investment catalogue, i.e., reinforcements of transmission lines and transformers.

2. *Investment model - part 1*

This investment model takes the outputs of the screening model (i.e., a list of attractive investment options) to identify the least-cost investments across a stochastic future using a MILP formulation. It is important to note that flexibility explored in ATTEST, such as the flexible power exchange at TSO/DSO interface, has little value under deterministic or low-uncertainty conditions, e.g., two deterministic scenarios depicted in Figure 1 (Martinez Cesena, et al., 2016). Accordingly, the model expands the scenarios to capture uncertainty by creating trees like those shown in Figure 2. It is important to note that the trees are used internally by the tool to properly estimate the value of flexibility, but the inputs and outputs of the tool only include the two original extreme scenarios.

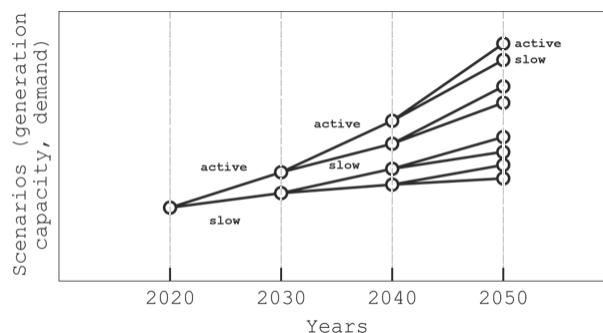


Figure 2: Example of scenario tree used by the planning tool

Part 1 of the investment model focuses on minimising investment costs while ignoring the operation costs. A SCACOPF model, developed in WP4 (Alizadeh, et al., 2021), is applied for **the peak time period** to identify the requirements in new investments.

3. Investment model - part 2

Part 2 of the investment model focuses on minimising both investment costs and operation costs. For this purpose, the objective function of the model is extended to include operational costs, and the ACOPF model is applied for **24h of a typical day** to estimate typical operational costs.

4. Getting started

Now that a high-level description of the tool has been provided, the next step is to install and use the tool. For that purpose, some preparations are required.

It is worth noting that, within the ATTEST project, all the developed tools can be accessed through a dedicated platform developed within WP6. Thus, most users do not need to install Tool 3.2 separately. However, this document provides a complete guide for advanced users who may want to develop the open-source transmission network planning tool or use it for stand-alone applications. The basic installation instructions for typical users are provided below, whereas the detailed instructions for developers are provided in “Appendix A – Installation instructions” of this deliverable.

To install the tool, a Python environment and a Command Line Interface (CLI), also known as a console or command prompt, are needed. The CLI should have git functionalities. There are different options available, but we recommend installing the anaconda package (<https://www.anaconda.com/>) which includes a Python environment (Spyder), CLI and other relevant functionalities, such as version control for git (required to access developer functionalities).

1. **Python environment** – Assuming the Anaconda package is installed, the Spyder Python environment can be accessed from:

Start --> All Programs --> Anaconda3 --> Spyder

2. **CLI** – Anaconda also provides a CLI (i.e., Anaconda Prompt). To access the CLI 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 available:

https://github.com/wangweik/ATTEST_Tool3.2

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.

5. Tool Installation

To use the transmission network planning tool for stand-alone applications, the related Python and Julia packages have to be installed as presented in detail below.

5.1. Installing the Python components of the tool

1. Please make sure to install a CLI and Python environment and to have access to the GitHub repository (see Section 4 Getting started).
2. Access the GitHub repository (https://github.com/wangweik/ATTEST_Tool3.2). Select the “Code” option from the repository, and then select “Download ZIP” (Figure 3. Download ZIP file of the tool Figure 3) to save a copy of the model. Afterward, unzip the file in the location where the tool should be installed. This will create the ATTEST_Tool3.2 folder.

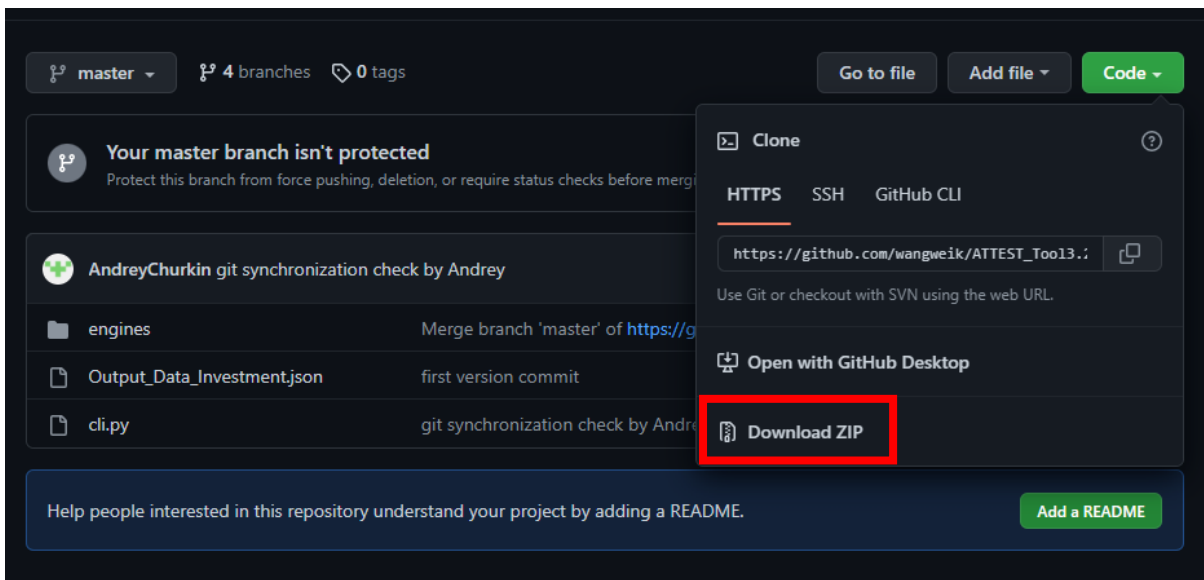


Figure 3. Download ZIP file of the tool

3. Open a CLI (see Figure 4)

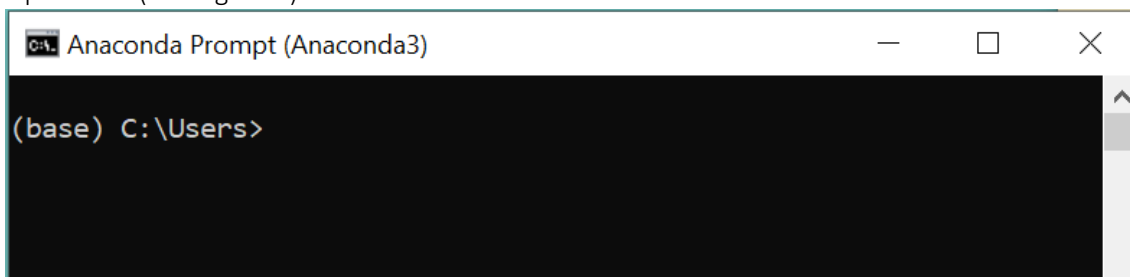


Figure 4. Example of CLI using Anaconda prompt

4. Navigate using the CLI to the location where the software is unzipped (this will be the location of the ATTEST_Tool3.2 folder). To access this location, the 'cd' command can be used:

cd C:\Location_of_software

Note that the command 'dir' can be used to see the folders available, and the command 'cd ..' can be used to exit a folder:

dir

cd ..

5. Use the console to enter the folder where the model is located.

cd ATTEST_Tool3.2

6. At this stage, the model is ready to be used. However, the Python software on a computer may not have by default all the packages required by the tool. If this is the case, when calling the Python tool (this will be presented in Section 6), the console will display that a package was not found. For example, if the console is displaying that the 'pyomo' package is missing, the package has to be installed using the CLI commands. The two typical commands are "pip install" and "conda install", as illustrated below with the installation of the pyomo package. However, some packages have bespoke installation commands which can be found online.

pip install pyomo

conda install pyomo

7. It is likely that the GLPK package will require additional installation. The bespoke command required for this is:

conda install -c conda-forge glpk

8. Another aspect to consider is incompatibility with some of the newer packages. For example, the model is not compatible with the latest release of pandas. Accordingly, an older version of pandas should be installed with:

pip install pandas==1.4.1

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

pip list

conda list

5.2. Installing the Julia components of the tool

1. The developed model includes an interface between Python and Julia, therefore, PyJulia should be installed.
 - a. Install Julia using the command:

pip install --user julia

- b. Install PyJulia: Julia needs to be installed into the same folder where Anaconda is installed to allow PyCall to work without path finding errors.
Download installer for the Julia programming language:

<https://julialang.org/downloads/>

After Julia is installed, open the Julia prompt (Figure 5):

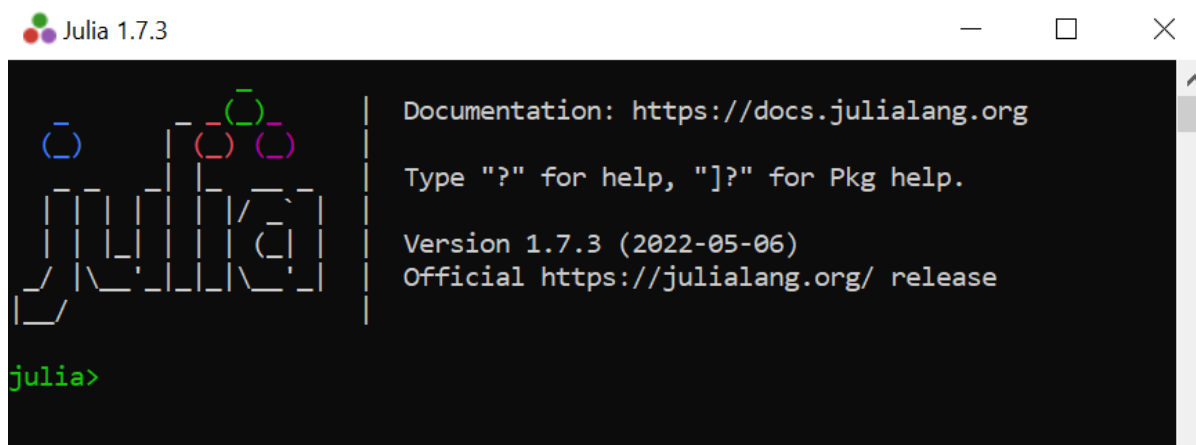


Figure 5. Julia prompt interface

Activate the packages installation by the command:

using Pkg

Then, Julia packages can be installed, for example, the PyCall package:

Pkg.add("PyCall ")

- c. Install Julia packages required by Pyjulia: launch a Python REPL and run the following code (Figure 6). A confirming message will be printed showing PyCall is built successfully.

```
In [2]: import julia
In [3]: julia.install()
Precompiling PyCall...
Precompiling PyCall... DONE
PyCall is installed and built successfully.
```

Figure 6. Installing PyJulia through a Python REPL

Detailed instructions for installing Pyjulia can be found at:

<https://pyjulia.readthedocs.io/en/latest/>

2. Required Python packages:
 - a. Python =3.8
 - b. Pyomo = 6.2
 - c. networkx = 2.6.3

- d. dataclasses = 0.8
 - e. numpy = 1.20.3
 - f. pandas = 1.4.1
 - g. pandapower = 2.9.0
 - h. odfpy = 1.4.1
 - i. GLPK = 5.0
 - j. Click
 - k. openpyxl
3. Required Julia packages:
- a. PyCall = 1.92.5
 - b. Ipopt = 1.0.2
 - c. JLD = 0.13.2
 - d. JuMP = 1.1.1
 - e. JSON = 0.21.3
 - f. OdsIO = 0.6.1
 - g. Dates
 - h. LinearAlgebra

The list of potential issues is provided in “Appendix B – Potential installation errors” of this deliverable. It is also possible to contact the developers through the ATTEST project of the Gitlab repository.

6. Tool execution

1. The users can run the tool in Python “path/ATTEST_Tool3.2” by simply using the command:

Python cli.py

- a. Help information

To access the help information of Tool 3.2, run the command:

Python cli.py --help

All the command lines are displayed with explanation, as shown in Figure 7.

```
>python cli.py --help
Usage: cli.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  run-all          Run all models
  run-investment   Run the investment model
  run-screening    Run the screening model
```

Figure 7. Help information with commands of the tool

- b. Screening model

To run the screening model only, run the command:

Python cli.py run-screening

Executing the screening model requires the following input data:

Required input data	Default input data
input directory:	[path for input file]
output directory:	[path for output file]
country name:	[UK]
test case name:	[m_file_name]
time-series data:	[xlsx_file_name]
contingency data:	[ods_file_name]
peak_hour:	[19]
Number of years:	[4]

Press enter when defining each input data. Then, the tool will present next input data for defining or start to execute the program. Note that if the input information is empty, default data is used for the simulations.

To access the help information for running the screening model, run the command:

Python cli.py run-screening --help

A detailed explanation of each input data is provided (Figure 8):

```
>python cli.py run-screening --help
Usage: cli.py run-screening [OPTIONS]

Run the screening model

Options:
  --input_dir TEXT           Specify the directory for input data
  --output_dir TEXT         Specify the directory for output data
  --country TEXT            Specify the country:"PT", "HR", or "UK"
  --test_case TEXT          Specify the test_case without ".m".
  --xlsx_file_name TEXT     Specify the xlsx_file_name without ".xlsx".
  --ods_file_name TEXT      Specify the ods_file_name without ".ods".
  --peak_hour INTEGER       Specify the number between 1 and 24, i.e., 19 for 7
                             p.m.
  --no_year INTEGER         Specify the number of years: 1 - [2020], 2 -
                             [2020,2030], 3 - [2020,2030,2040], 4 -
                             [2020,2030,2040,2050]
  --help                    Show this message and exit.
```

Figure 8. Help information of the screening model of the tool

c. Investment model

To run the investment model only, run the command:

Python cli.py run-investment

Executing the investment model requires the following input data:

Required input data	Default input data
input directory:	[path for input file]
output directory:	[path for output file]
country name:	[UK]
test case name:	[m_file_name]
time-series data:	[xlsx_file_name]
contingency data:	[ods_file_name]
peak_hour:	[19]
Number of years:	[4]
run_both	[True]

Press enter when finishing each input data. Then, the tool will present next input data for defining or start to execute the program. Note that if the input information is empty, the default data is used for the simulations.

To access the help information for running the investment model, run the command (Figure 9):

Python cli.py run-investment --help

```
>python cli.py run-investment --help
Usage: cli.py run-investment [OPTIONS]

Run the investment model

Options:
  --input_dir TEXT          Specify the directory for input data
  --output_dir TEXT        Specify the directory for output data
  --country TEXT           Specify the country:"PT", "HR", or "UK"
  --test_case TEXT        Specify the test_case without ".m".
  --xlsx_file_name TEXT    Specify the xlsx_file_name without ".xlsx".
  --ods_file_name TEXT     Specify the ods_file_name without ".ods".
  --peak_hour INTEGER      Specify the number between 1 and 24, i.e., 19 for 7
                           p.m.
  --no_year INTEGER        Specify the number of years: 1 - [2020], 2 -
                           [2020,2030], 3 - [2020,2030,2040], 4 -
                           [2020,2030,2040,2050]
  --run_both BOOLEAN       Define investment setting, True = [considering both
                           investment cost and operation cost], False =
                           [considering investment cost only]
  --help                   Show this message and exit.
```

Figure 9. Help information of the investment model of the tool

One extra input data flag, “run_both”, is required by the investment model. This flag indicates whether to run the investment model considering investment costs only (False) or considering both investment and operation costs (True).

- d. All models

To run both screening model and investment model, run the command:

Python cli.py run-all

The help information can be accessed through the comment:

Python cli.py run-all --help

- 2. Users can also import the access method in a Python file, which allows to access the package without the CLI. If users installed the library, then they just need to use it in Python with the codes:

from engines.screening_model import run_main_screening

from engines.investment_model import run_main_investment

Input data for the screening model and investment model are indicated in the above tables.

7. Input and output data

- 1. Input data:

The input data for D3.3 includes network data (from WP2), contingency data (from WP4), cost information (from WP2), and asset life assessment index (from WP5).

- a. Network information, including future energy scenarios and generation costs, is stored in “engines/tests/json”. The flexibility service costs are assumed to be 107.24 €/MWh.

Example:

Transmission_Network_PT_2030_Active_Economy.m

- b. Time-series data, including demand, generation, and flexibility, is stored in “engines/tests/excel”. Note if flexibility data are missing, the tool will by default use 10% of the peak load as the flexibility for each load bus, i.e., if the peak load is 100MW, then the upward flexibility is 10MW and the downward flexibility is 10MW.

Example:

Transmission_Network_PT_2030_Active_Economy_24hGenerationLoadData.xlsx

- c. Contingency data is stored in “engines\SCOPF_R5\input_data”. If contingency data is not found, the screening model will use N-1 contingencies. However, this may cause load curtailments due to network islanding. Therefore, a list of contingencies is strongly recommended for input data.

Example:

case_template_PT.ods

- d. Intervention lists and costs for branch and transformer investments are stored in “engines/tests/json”. Note if the intervention input data is missing, a set of default data will be used.

Example:
Intervention.json

- e. Life assessment indicators for network physical assets, i.e., branches and transformers, are stored in “engines/tests/csv”.

Example:
mpc_life_assessment_index.json

2. Output data:

The output data from D3.3 includes results from both the screening model and the investment model. For the outputs of the investment model, if only part 1 is run, the results will be only for part 1, excluding the operation costs. If both part 1 and part 2 are executed, the results will be two files.

- a. Results from the screening model are stored in “engines/results”

Example:
screen_result_PT_Transmission_Network_PT.json

- b. Results from the investment model are stored in “engines/results”

Example:
investment_result_PT_Transmission_Network_PT_pt1.json
investment_result_PT_Transmission_Network_PT_pt2.json

The format of the information, and the use of the tool, are illustrated below with a test case.

8. Test case

The example network is a modified 5-bus system, which is described in detail in “Appendix C – Modified Case5.m file data” of this deliverable. UK’s data for future energy scenarios for 2030, 2040, and 2050 are used for the 5-bus test case. The topology for the example network is presented in Figure 10.

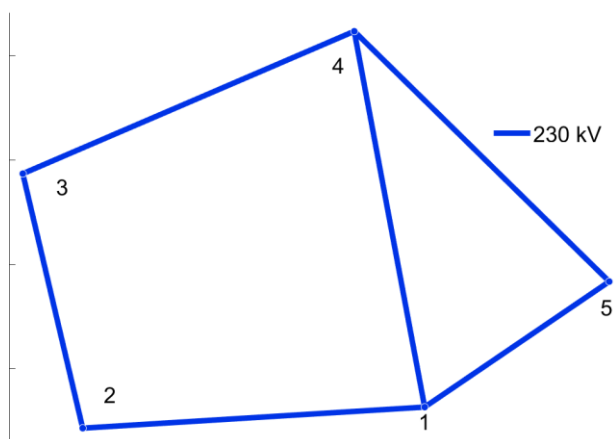


Figure 10. Topology of the 5-bus system

Investment catalogue:

An investment catalogue including both capacity and the corresponding costs for branches and transformers are required as an optional input. If no such data is found, a set of default data based on assumptions will be used.

"line_list": [50,100,150,200,250,300,500]

"line_cost": 1000 euro/MVA/km

"transformer_list": [140, 280, 450, 800]

"transformer_cost": 9903 euro/MVA

List of contingencies:

contingency	From	To
1	4	5

Screening model results:

The screening results are present in the way

“branch/transformer number”: [capacity investment for branch/transformer]

For example, “1”: [10] means that increase the capacity of branch1 of 10MVA. Note that here “branch number” includes both branches and transformers, but their investments are distinguished by the pre-defined investment catalogues.

{"0": [], "1": [], "2": [50, 100, 150], "3": [], "4": [], "5": []}

Investment model part 1 results:

{"Country": "UK", "Case name": "case5",

"Scenario 1": {

"Total investment cost (EUR-million)": 0.0, "Branch investment cost (EUR-million)": 0.0,

"Flexibility investment cost (EUR-million)": 0.0, "Net Present Operation Cost (EUR-million)": 0,

"2020": {

"Operation cost (EUR-million/year)": 0,

"Branch investment (MVA)": [0, 0, 0.0, 0, 0, 0],

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

"2030": {

"Operation cost (EUR-million/year)": 0,

```

    "Branch investment (MVA)": [0, 0, 0.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2040": {
    "Operation cost (EUR-million/year)": 0,
    "Branch investment (MVA)": [0, 0, 0.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2050": {
    "Operation cost (EUR-million/year)": 0,
    "Branch investment (MVA)": [0, 0, 0.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"Scenario 2": {
    "Total investment cost (EUR-million)": 0.05, "Branch investment cost (EUR-million)": 0.05,
    "Flexibility investment cost (EUR-million)": 0.0, "Net Present Operation Cost (EUR-million)": 0,
"2020": {
    "Operation cost (EUR-million/year)": 0,
    "Branch investment (MVA)": [0, 0, 0.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2030": {
    "Operation cost (EUR-million/year)": 0,
    "Branch investment (MVA)": [0, 0, 0.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2040": {
    "Operation cost (EUR-million/year)": 0,
    "Branch investment (MVA)": [0, 0, 0.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2050": {
    "Operation cost (EUR-million/year)": 0,
    "Branch investment (MVA)": [0, 0, 50.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]}

```

Investment model part 2 results:

```

{"Country": "UK", "Case name": "case5",

```

```

"Scenario 1": {

```

```

"Total investment cost (EUR-million)": 0.15, "Branch investment cost (EUR-million)": 0.15,
"Flexibility investment cost (EUR-million)": 0.0,
"Net Present Operation Cost (EUR-million)": 39749.25235651885,
"2020": {
  "Operation cost (EUR-million/year)": 1283.95331139313,
  "Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],
  "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2030": {
  "Operation cost (EUR-million/year)": 1658.2366686458026,
  "Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],
  "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2040": {
  "Operation cost (EUR-million/year)": 2409.3160007390693,
  "Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],
  "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
"2050": {
  "Operation cost (EUR-million/year)": 3230.6244760178633,
  "Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],
  "Flexibility investment (MW)": [0, 0, 0, 0, 0]}},
"Scenario 2": {
  "Total investment cost (EUR-million)": 0.15, "Branch investment cost (EUR-million)": 0.15,
  "Flexibility investment cost (EUR-million)": 0.0,
  "Net Present Operation Cost (EUR-million)": 39749.25235651885,
  "2020": {
    "Operation cost (EUR-million/year)": 1283.95331139313,
    "Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
  "2030": {
    "Operation cost (EUR-million/year)": 1492.2323400664268,
    "Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],
    "Flexibility investment (MW)": [0, 0, 0, 0, 0]},
  "2040": {
    "Operation cost (EUR-million/year)": 1958.9503170133917,

```

```
"Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],  
"Flexibility investment (MW)": [0, 0, 0, 0, 0]},  
"2050": {  
  "Operation cost (EUR-million/year)": 2570.949394651704,  
  "Branch investment (MVA)": [0, 0, 150.0, 0, 0, 0],  
  "Flexibility investment (MW)": [0, 0, 0, 0, 0]}}
```

9. Bibliography

Alizadeh, M. I., Usman, M. & Capitanescu, F., 2021. A direct approach to stochastic multi-period AC security constrained optimal power flow. *arXiv:2104.15038*, pp. 1-9.

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

Martinez Cesena, E. A., Capuder, T. & Mancarella, P., 2016. Flexible distributed multi-energy generation system expansion planning under uncertainty. *IEEE Transactions on Smart Grid*, 7(1), pp. 348 - 357.

Appendix A – Installation instructions for developers

1. Please make sure that a CLI and Python environment are installed and that access to the GitHub repository is obtained (Sections 4 and 5).
2. Open a CLI (see Figure 41)

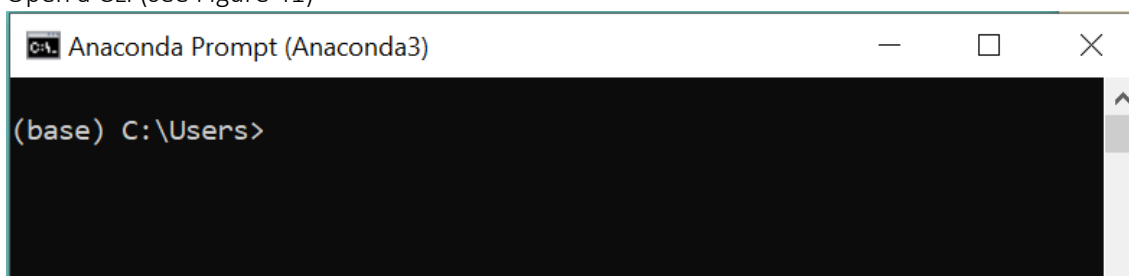


Figure 11. Example of CLI using Anaconda prompt

- Navigate, using the CLI to the location where the software should be installed (this will be the location of the ATTEST_Tool3.2 folder). To access this location, the change directory, 'cd', command can be used:

```
cd C:\Location_of_software
```

Note that the command 'dir' can be used to see the folders available, and the command 'cd ..' can be used to exit a folder:

```
dir
```

```
cd ..
```

- Access the Github repository (https://github.com/wangweik/ATTEST_Tool3.2). Select the "Code" option from the repository, select "HTTPS" and copy the URL (see Figure 12).

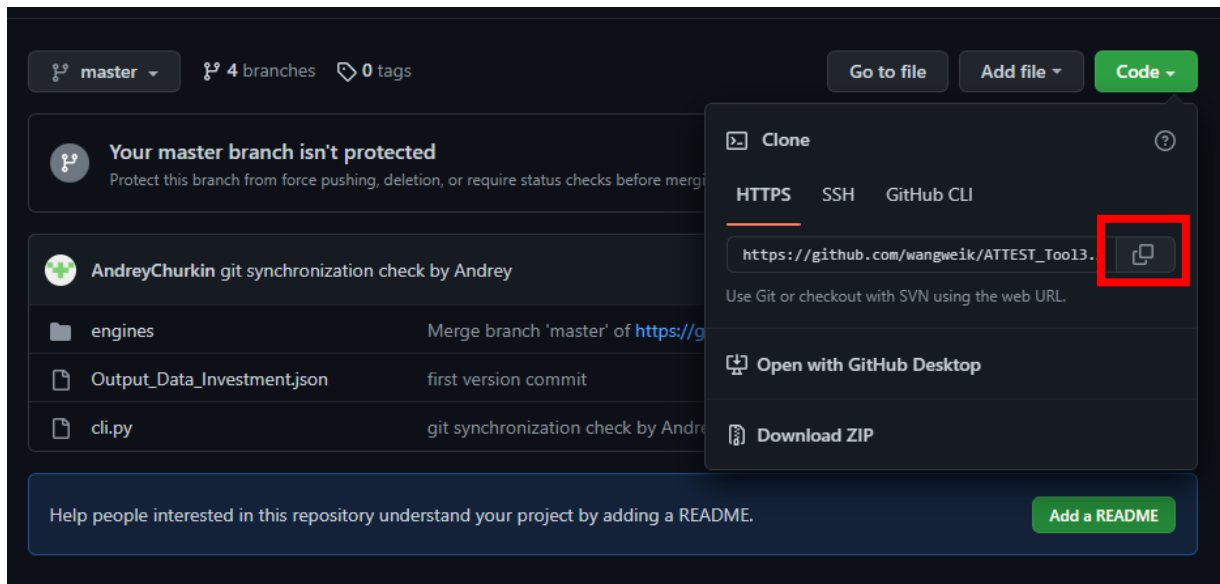



Figure 12. Download the tool using Git

In the CLI, use the "git clone" command with the provided URL (also copied below). The console will then install the software in the desired location.

```
git clone https://github.com/wangweik/ATTEST_Tool3.2.git
```

The console will then clone the model (Figure 13), which will continue to be linked to the GitHub repository. This version provides full access to the GitHub functionalities, including options to update the tool.



```
Anaconda Prompt (anaconda3)
(base) C:\Users\kong>cd C:\Users\kong\UoM
(base) C:\Users\kong\UoM>git clone https://github.com/wangweik/ATTEST_Tool3.2.git
Cloning into 'ATTEST_Tool3.2'...
remote: Enumerating objects: 773, done.
remote: Counting objects: 100% (176/176), done.
remote: Compressing objects: 100% (114/114), done.
remote: Total 773 (delta 107), reused 112 (delta 62), pack-reused 597
Receiving objects: 97% (750/773), 3.57 MiB | 3.54 MiB/s
Receiving objects: 98% (758/773), 3.57 MiB | 3.54 MiB/s
Receiving objects: 100% (773/773), 4.59 MiB | 3.71 MiB/s, done.
Resolving deltas: 100% (473/473), done.
```

Figure 13. Tool download through prompt using git clone

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

cd ATTEST_Tool3.2

6. The console will now display the name of the current branch, typically the master branch. The branches are different versions of the model, which 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. Update the model by pulling all the changes from GitHub:

git pull

8. At this stage, the model has been installed with developer functionalities. Please complete the installation by following the steps in Section 5.2.

The list of potential issues is given in “Appendix B – Potential installation errors” of this deliverable. It is also possible to contact the developers through the ATTEST project of the Gitlab repository.

Appendix B – Potential installation errors

1. The above-mentioned programmes are not installed in the same folder path, or error in installing GLPK, or error in not finding PyCall while using PyJulia.

Solution 1: create a virtual environment in Python for the tool, then install all the required packages to the virtual environment: The commands for creating and activating virtual environment are:

conda create -n new_env_name Python=3.8

conda activate new_env_name

Note that “new_env_name” is the name for environment, Python version = 3.8.

Potential problems: can be solved by running prompt as admin; using prompt instead of command window.

Solution 2: add or change path of environmental variables as shown in Figure 14

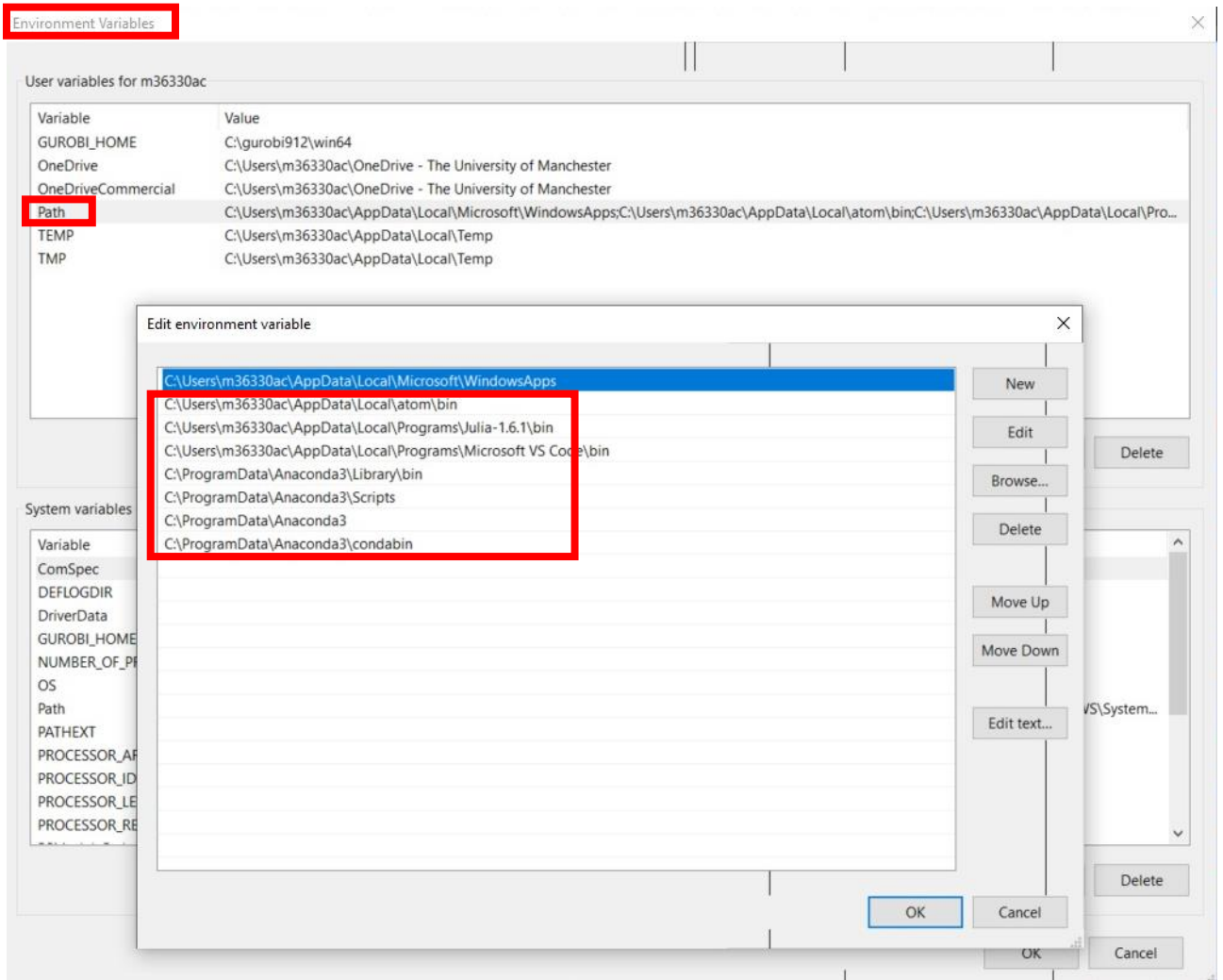


Figure 14. Example of windows system Environment Variables

2. Pyomo installation:
Using command to install pyomo:

pip install pyomo

3. Missing packages:
If some packages are missing, using following commands to install the missing package:

pip install package_name

conda install package_name

Some of the packages cannot be installed using conda, “pip install” will work fine, for example, using command to install pyomo:

pip install pyomo

4. Julia package version error:

While running the tool, there a potential error from Julia:

JULIA: LoadError: MethodError: no method matching Model(; add_bridges=false)

This is due to the reason that the version of JuMP is not up to date, use the following command to update JuMP in Julia prompt:

Using Pkg

Pkg.update("JuMP")

If the package is not updated, then using the following command to update all the packages:

Pkg.update()

To check the versions of Julia packages, using the command:

Pkg.status ()

Appendix C – Modified Case5.m file data

```
function mpc = case5
```

```
%% 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 2 110 10 0 0 1 1 0 230 1 1.1 0.9;
 2 1 200 58.61 0 0 1 1 0 230 1 1.1 0.9;
 3 2 230 78.61 0 0 1 1 0 230 1 1.1 0.9;
 4 3 200 31.47 0 0 1 1 0 230 1 1.0 1.0;
 5 2 100 10 0 0 1 1 0 230 1 1.1 0.9;
```

```
];
```

```
%% 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 = [
```



```

1 40 0 1300 -1300 1 100 1 2400 0 0 0 0 0 0 0 0 0 0 0;
3 323.49 0 390 -390 1 100 1 100 0 0 0 0 0 0 0 0 0 0 0;
4 0 0 150 -150 1 100 1 250 0 0 0 0 0 0 0 0 0 0 0;
5 466.51 0 450 -450 1 100 1 600 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 2 0.00281 0.0281 0.00712 500 0 0 0 0 1 -360 360;
1 4 0.00304 0.0304 0.00658 300 0 0 0 0 1 -360 360;
1 5 0.00064 0.0064 0.03126 100 0 0 0 0 1 -360 360;
2 3 0.00108 0.0108 0.01852 250 0 0 0 0 1 -360 360;
3 4 0.00297 0.0297 0.00674 200 0 0 0 0 1 -360 360;
4 5 0.00297 0.0297 0.00674 350 0 0 0 0 1 -360 360;
];

%%----- 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 10 0 0;
2 0 0 2 15 0 0;
2 0 0 2 20 0 0;
2 0 0 2 30 0 0;
];

```