ENS Project Lab Report

Panda Power Project

Advisor	Prof. Dr. Thomas Hamacher
	Chair of Renewable and Sustainable Energy Systems
Submitted by	Jiahe Chu, Karim Kaaniche, Martin Oviedo, Rohith Sureshbabu, Taeyoung Kim
Submitted on	Munich, 06.02.2023

Contents

1. Introduction

1.1. Project Overview

In this project lab, we successfully developed a toolbox using *pandapower* to do network simulations in different scenarios. The toolbox contains a front-end, a database, and an analysis part.

- The front-end allows users to define network topologies and different load/generation scenarios via Excel, then uses python scripts to translate the Excel files to *pandapower* networks.
- The database contains six pre-defined topologies and seven pre-defined scenarios, which users can use directly.
- The analysis part does the network calculation and analysis, then summarizes the results to an Excel output file with a dashboard and plots the network.

The toolbox also supports time series analysis and optimal power flow calculation. A basic overview of the code structure is shown in Figure 1. All the source codes are available in our GitLab repository: https://gitlab.lrz.de/jiahechu/propens-pandapower

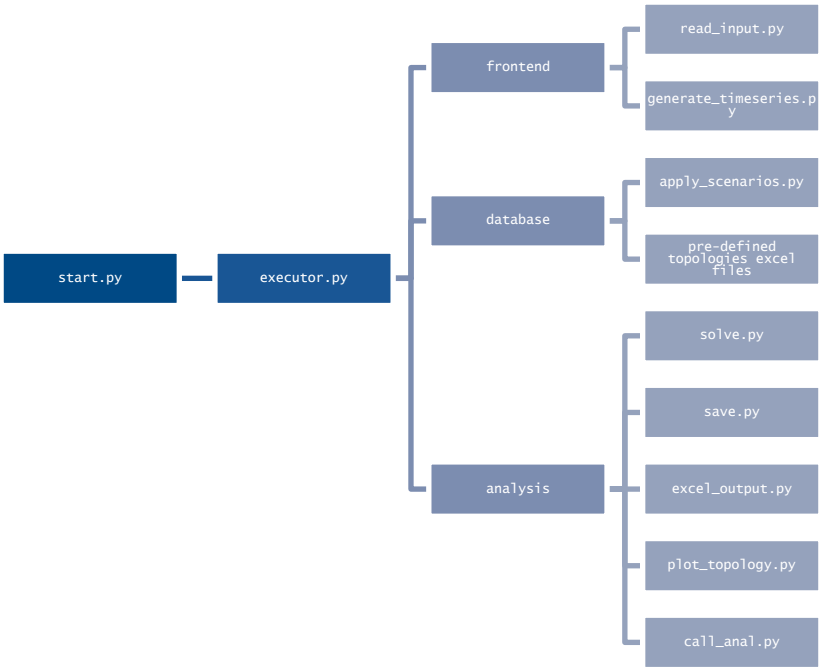


Figure 1 Basic code structure of the toolbox

1.2. Responsibility

Front-end: Jiahe Chu

Database: Karim Kaaniche, Rohith Sureshbabu

Analysis: Martin Oviedo, Taeyoung Kim

1.3. Time Schedule

Table 1 Time Schedule

Week	Front-end	Database	Analysis
Nov. 9 – Nov. 15	implement input function	finish scenarios and topologies list	excel/data output formatting; - analysis function
Nov. 16 – Nov. 22	Help implement scenarios and topologies	implementation of scenarios and topologies	excel/dashboard output formatting; analysis function and print it in excel
Nov. 23 – Nov. 29	design example input	implementation of scenarios and topologies	link output to front-end configuration
Nov. 30 – Dec. 6	implement generate time series function	implementation of scenarios and topologies	link output to front-end configuration, clean code
Dec. 7 – Dec. 13	implement test cases	implement test cases, clean code	clean and standardize code
Dec. 14 – Dec. 20	Debugging	implement test cases, clean code	clean and standardize code, debugging
Jan. 11 – Jan. 17	Debugging, Implement optimal power flow	Documentation, readme	Debugging
Jan. 18 – Jan. 24	Write report	Write report	Write report
Jan. 25 – Jan. 31	Prepare presentation	Prepare presentation	Prepare Presentation

2. Front-end

2.1. User Interface

The input data will be given via the user interface, which consists of two parts: Excel input files and a python starting script. First, the network and scenarios will be defined in the Excel files, then the python script will be called to execute the toolbox.

2.1.1. Excel Input Files

Figure 2 shows the basic structure of the Excel input files, it consists of three different parts: topology, scenarios, and if the user chooses to do time series calculation: time steps file. The topology and time series both contain only one Excel file, the scenarios consist of multiple Excel files for multiple load/generation situations.

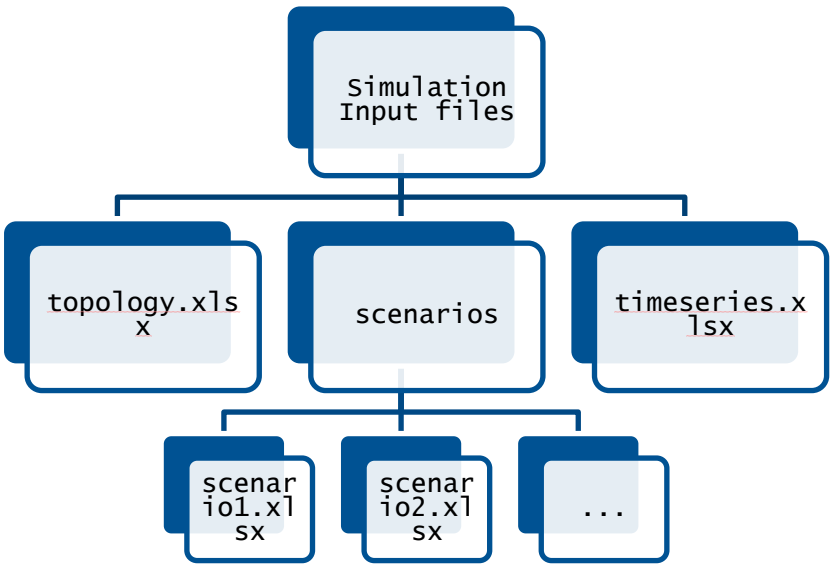


Figure 2 Structure of the simulation input files

The topology file and scenarios files contain the *pandapower* network components. They are split as follows:

- Topology (defines the graphical topology of the network):
    - bus
    - line
    - switch

    - dc line
    - impedance
    - shunt
- Scenarios (define the load/generation situation of the network):
    - load
    - motor
    - asymmetric load
    - static generator
    - asymmetric static generator
    - external grid
    - transformer
    - transformer3w
    - generator
    - storage
    - ward
    - extended ward
    - pwl_cost (for optimal power flow)
    - poly_cost (for optimal power flow)

One topology can have multiple corresponding scenarios. Take Kerber Landnetze Freileitung 2 as an example, Figures 3 and 4 show the graphical network topology and the topology input Excel file structure. The topology Excel file takes only information of buses and lines. For this topology, 3 different scenarios are defined: basic scenario, PV scenario, and PV-storage scenario. Figure 5 shows the scenario excel files of three scenarios, the orange part is the additional information of the corresponding scenarios compared to the basic scenario. In the basic scenario, only loads (households), transformer station, and external grid are added to the network. The PV scenario has an additional PV system in each household. In the PV-Storage scenario, the private batteries are also defined beside the PV systems.

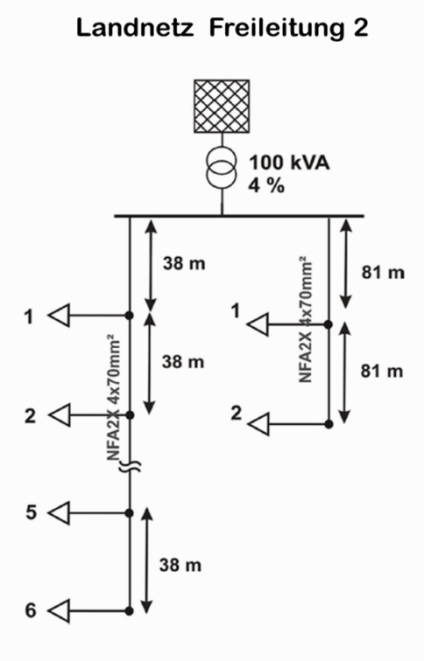


Figure 3 Topology of Kerber Landnetze Freileitung 2

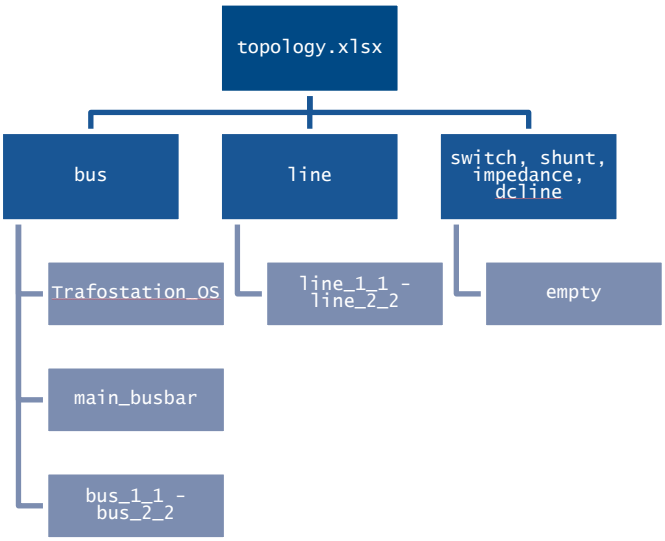
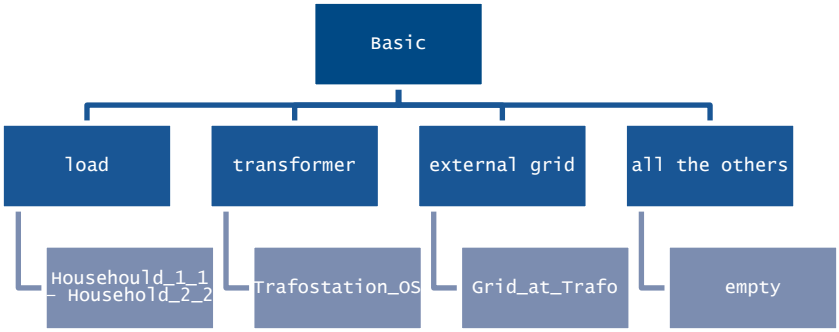


Figure 4 Topology input Excel file structure of Kerber Landnetze Freileitung 2



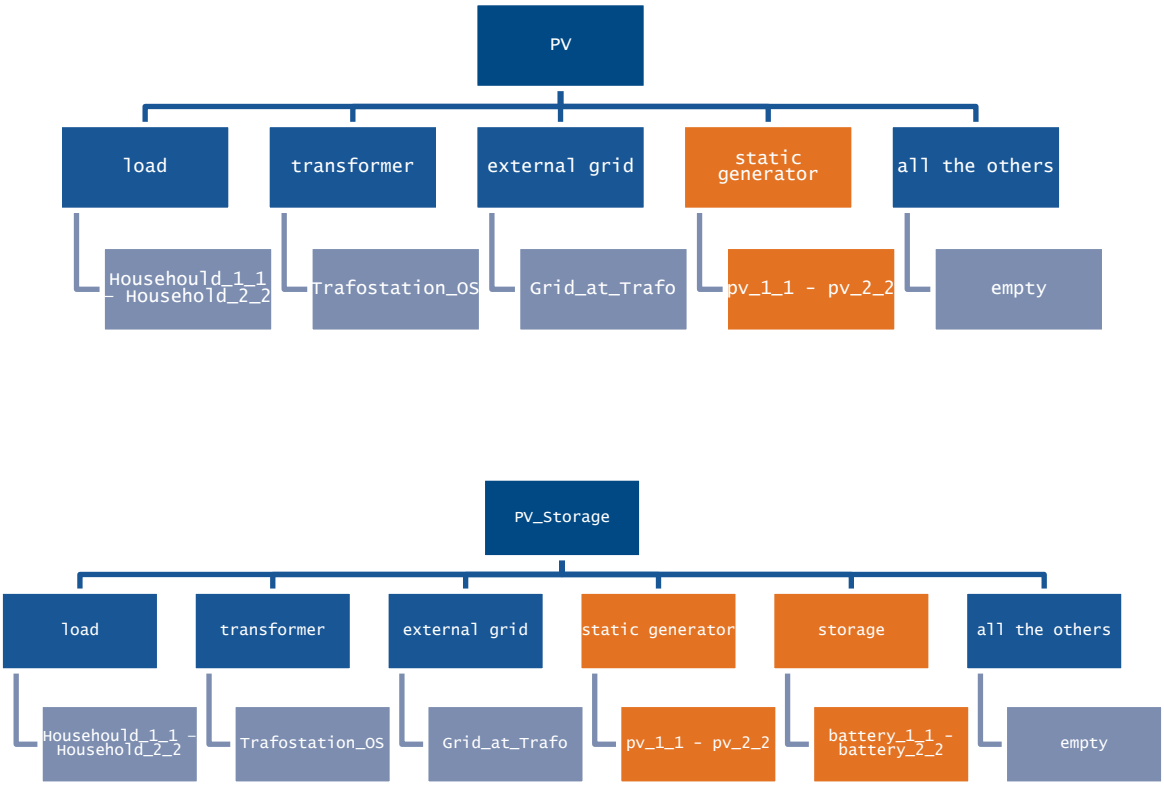


Figure 5 Excel file structure for scenario basic, PV and PV-Storage

Users can choose if the program does the time series analysis or optimal power flow calculation for the scenario in its corresponding Excel file. If the user decided to do a time series analysis, an additional time steps Excel file needs to be provided. It should contain the time series data for all the parameters which vary between time steps. The path which contains the time steps Excel file should be given as a string in the corresponding scenario Excel file.

2.1.2. Python Starting Script

After preparing the input Excel files, the simulation can be started by calling the Python starting script.

Figure 6 shows an overview of the starting script. The starting script consists of two dictionaries: input setup and output setup. In the input setup, names and paths which contain Excel input files of the network and scenarios need to be given as strings. If the user chooses to use a pre-defined scenario (detailed description in chapter 3: Database), it must

be also specified in the input setup. The output setup contains the path of the folder, in which the output file (detailed description in chapter 4: Analysis) is going to be saved, and what plots are going to be done.

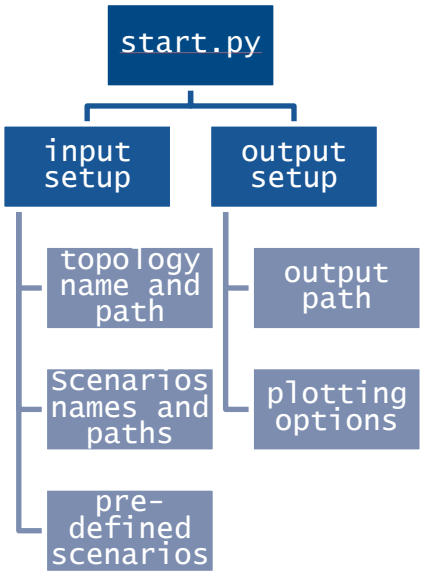


Figure 6 Structure of the Python starting script

After defining the input and output setup, users can simply run the starting script and the toolbox is going to be executed.

2.2. Code Implementation

The toolbox is executed via an executor. In the executor, all the functions are going to be called step by step to carry out the simulation (Figure 1). The front-end part is called first. Two relevant functions for the front-end are the read input function and generate time series function.

The read input function takes the paths of the network topology Excel file and a scenario Excel file as input. In the function, the two Excel files are combined into one file. Then the combined Excel is read by *pandapower* and returns a *pandapower* network as function output. For each scenario, the read input function is called once in the executor and passes a *pandapower* network to the next step.

The generate time series function is called additionally after the read input function for the scenarios, in which time series analysis is set to true. The function takes a *pandapower* network and the path of the time steps Excel file as input. In the function, *pandapower* controllers for time series analysis are created according to the time steps Excel file and are added to the given *pandapower* network. Then the function returns the *pandapower* network with time series controllers as output.

3. Database

Database is in general defined as a collection of data that is electronically collected and accessed.

In this project the database consists of a collection of different electrical network topologies. This also considers the predefined scenarios that are relevant to these topologies. With the help of the front-end, these topologies and scenarios are combined and are returned as a Pandapower network. This is further taken to analysis division for power flow calculations.

3.1. Topologies

Topologies are the different architectures that we can have for a Network. In our case we are considering electrical Networks.

While going through the Pandapower documentation, it was found out that there are some predefined networks. After an observation about the available topologies, the next step was to understand how to work with these topologies. In other words, how this information should be well explored.

Exploring the topologies is to discover the main structure of the topologies, for example how many lines are there, or how many buses, or if there are some switches...

This was with the help of the Pandapower documentation. The idea was to find the appropriate function for example to find how many buses or how man lines can be found in a topology, or also how can we add a bus.

In order to have a variety of topologies, the focus was to see what can differ from one network to another.

The first observation was for example, that there are some networks that are complex and other that they are simple. Some topologies are built with branches and some without. With this good variety the user will have mostly no need to add a new topology, and they can use one of predefined ones. However, the user still has the possibility to configure the topology or create a new one.

Six topologies were selected:

- Four loads with branches out (Figure 7)

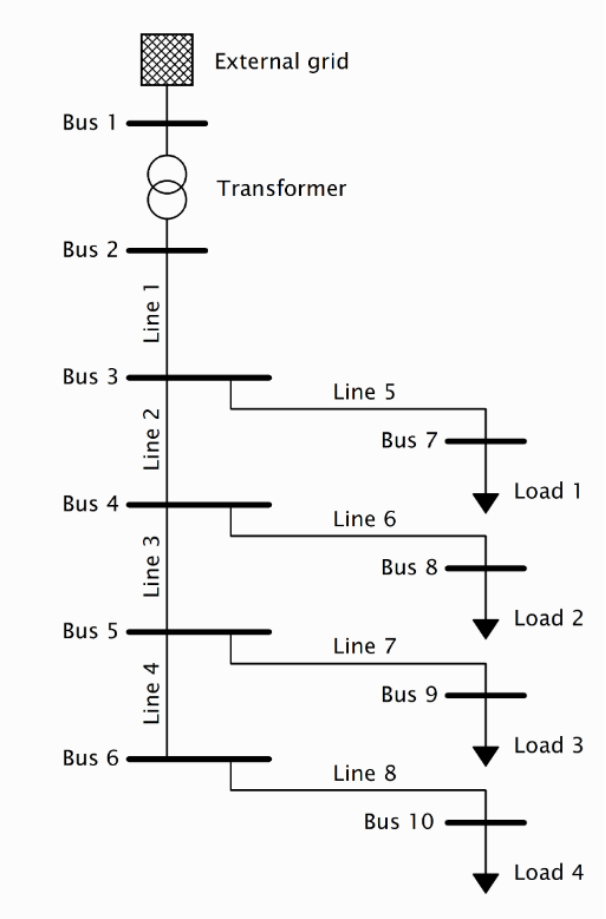


Figure 7 Four loads with branches out

- Simple example network (Figure 8)**Error! Reference source not found.**

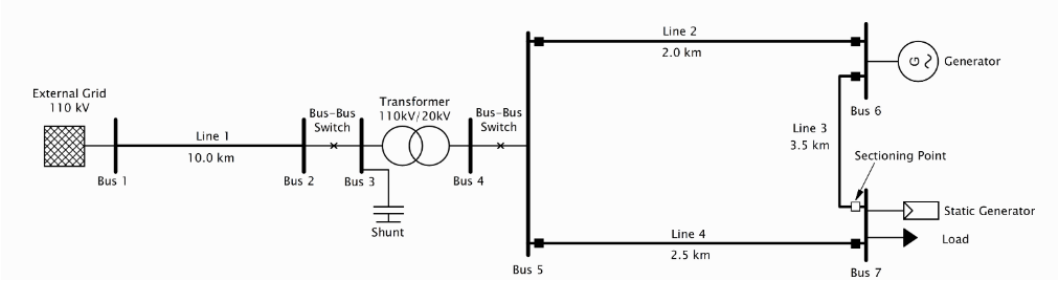


Figure 8 Simple example network

- Multi-Voltage Level Example Network (Figure 9)

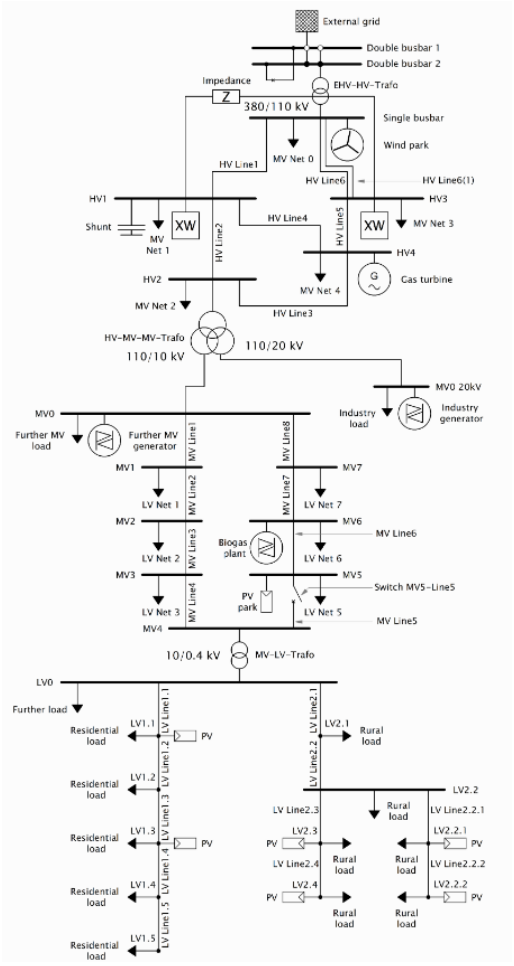


Figure 9 Multi-Voltage Level Example Network

- Kerber Landnetz Freileitung 1 (Figure 10)

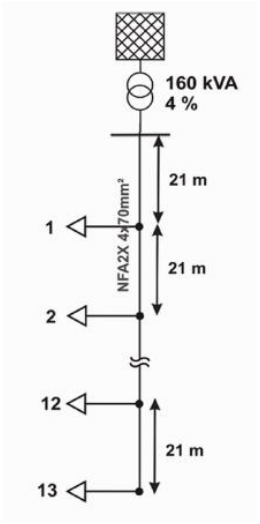


Figure 10 Kerber Landnetz Freileitung 1

- Kerber Landnetz Freileitung 2 (Figure 11)

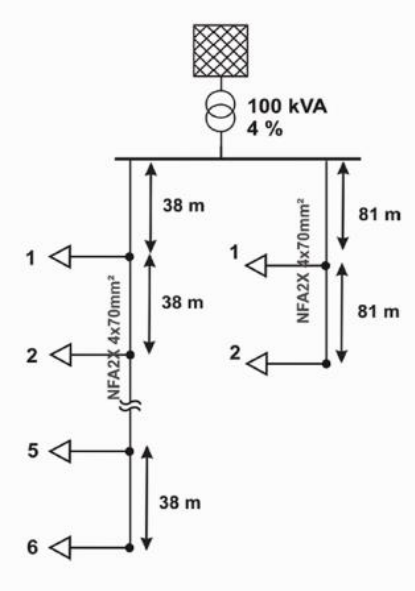


Figure 11 Kerber Landnetz Freileitung 2

- Four load branch (Figure 12)

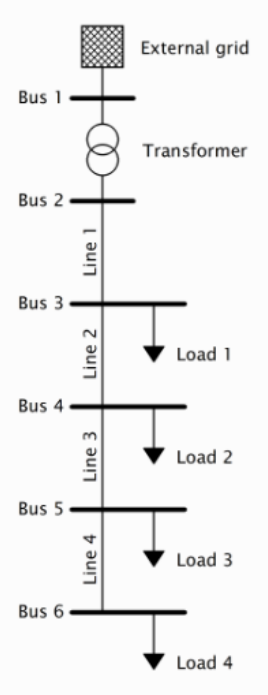


Figure 12 Four load branch

The needed Information was sorted out from the different topologies: needed Information to work with and later for the analysis. This work was done with some code (see Figure 13) to get the needed information.

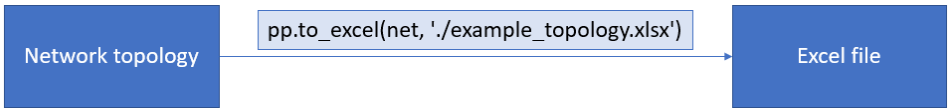


Figure 13 Explanatory diagram for obtaining the relevant information about the network topology

The obtained excel file is modified according to the needed information. It could be that some excel files have some extra information that are not relevant for our cases, they could be removed, in other words adapt the obtained file in order to have the needed front -end data.

3.2. Scenarios

Scenarios are the different combinations of network components that can be altered by the user flexibly in a real network with respect to a particular topology. As explained earlier in the front-end (2), each topology corresponds to itself multiple scenarios of load and generation. This part of the toolbox is the place where the user has freedom to experiment with these components and how it plays a role in the final power flow analysis.

Once the topology of the network is input into the Pandapower network in the form of excel, the scenarios need to be fed into them. Depending on the topology that is being used for the analysis, the predefined scenario used might vary, and afterwards if needed the user can scale the individual parameters and can vary the network scenarios or can leave it as it is. For Example, if a User wants to double the capacity of Solar PV plant, he/she can double the active and reactive power as per the needs of their network.

The different components/parameters of network elements that can vary to constitute the network scenarios are:

- Static Generator:
    - Scaling **Active and reactive power** – 0 to 200 %
        - PV
        - Wind
        - Conventional Power Plants
- Load
    - Scaling **Active and reactive power** – 0 to 200 %
- Transformer
    - Scaling **Apparent power** – 0 to 200 %
- Lines

- Scaling the **Maximum thermal current** – 0 to 195 %
- Increasing the **number of parallel lines** – greater than or equal to 2

- Storage
  - Scaling **Maximum storable energy** – 0 to 100 %

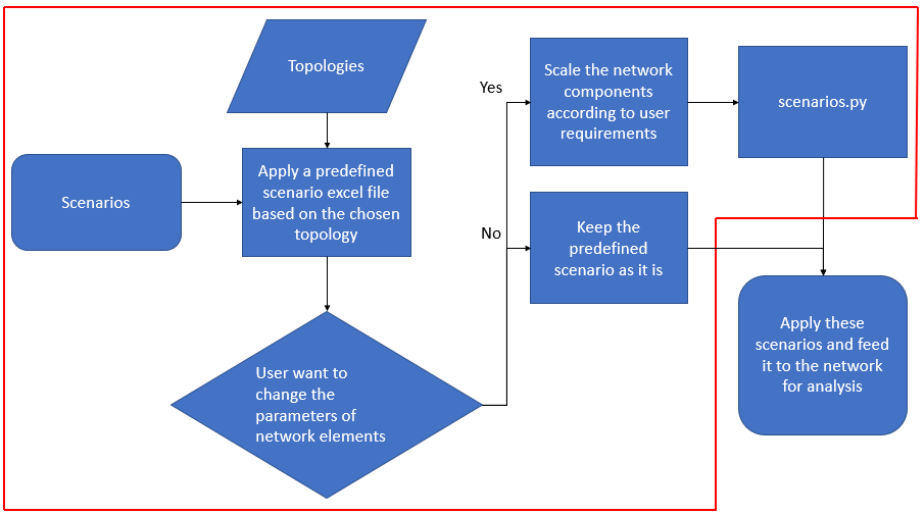


Figure 14 Logic flow of scenario script

A Flowchart showing the logic of the Scenario script is shown above. Shown below is the flowchart of the part of the Pandapower program that carries out the scenario part. In the file 'scenarios.py' which is the database of all predefined scenarios, various functions are defined such that each function takes in an input from the user. Depending on the type of parameter that is dealt with, the user gives a value to alter them. These parameters are already mentioned above. After altering the parameters using the user inputs, these are again fed into the Pandapower network. This is then taken into the executor for the power flow analysis after the input is applied to the network.

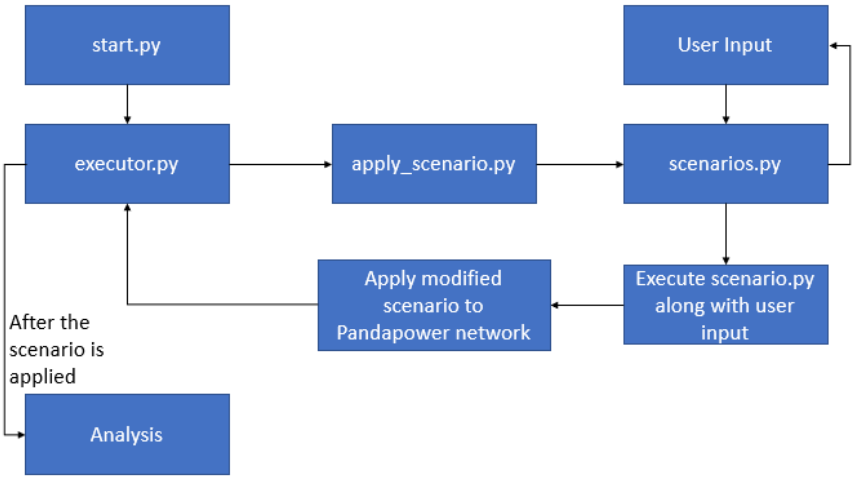


Figure 15 Data flow of scenario in the program

Additionally, the scenario was tried with the switches to perform the power flow analysis based on multiple network zones, but due to the complexity of manipulating the switches, they were ignored for our program.

4. Analysis

This part of the toolbox consists of receiving the network data, solve it, sorting the generated results, and creating the output. The input, process, and output of each step of the analysis is shown in Figure 16. The circles represent the functions and the arrows the files or data that is sent to the next process. The following subsections describes each step in a more comprehensive manner. The output setup contains two essential information; firstly, the output path where the Excel's spreadsheet is going to be saved, by default it is at the folder named 'results' in the toolbox; secondly, the setup tells the toolbox what plots should be done, two of them would be obtained directly after solving the network, and the last would be the topology, that can be save as an image.

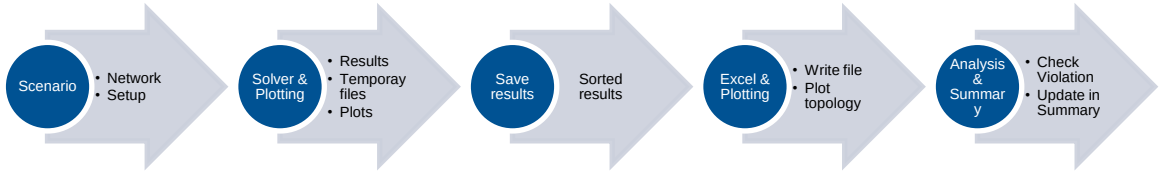


Figure 16: Analysis' Workflow

4.1. Solver

A more detailed workflow of the solver is shown in Figure 17. The first step is calling the solver, it receives the network, and the configuration of what problem should be solved. The latter includes whether the problem is a Power Flow (PF) or an Optimal Power Flow (OPF), whether it is a linearized approximation or the full non-linear, and finally, whether it is a time series analysis or it is a one-time step iteration.

After defining the problem that should be solved, the solver calls the parameters needed for the results. These parameters are obtained by calling the 'output parameters' functions, it reads the network and provides a list of pre-defined parameters from each element in the network. Thus, it is possible to add an element or modify the output just by changing the list of parameters where the 'output parameters' function is defined.

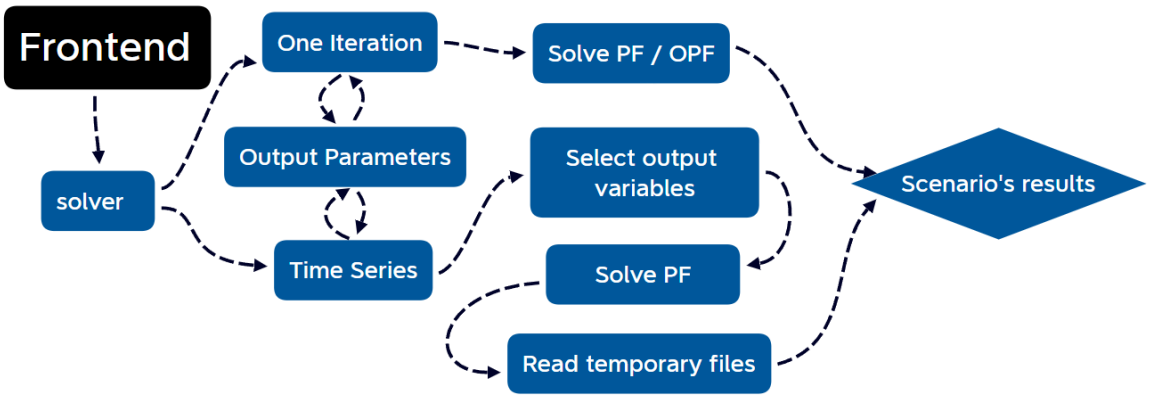


Figure 17: Solver's Workflow

If only one iteration will be calculated, the network can be solved directly. In case the scenario includes a time series calculation, then more steps are required. The first step is to add the output parameters to the time series' writer (pandapower' function to work with time series). The reason is that in *pandapower's* time series tool, pandapower does not save any results by default. Thus, we configurate the tool in this step. Next, we solve the network and *pandapower* saves the output parameters' results in temporary files. Finally, we read the files with the results, where each parameter of each element is a spreadsheet, as there are two dimensions i.e. time steps and the number of elements. Finally, the solver returns the scenario's results and plots the results according to the setup. Only one iteration scenarios are plotted, and the plots are saved according to the output path. There are two available interactive plots. One plot include the load, generation, lines impedances and voltage levels; and the second is a heat map of the branches loading percent and of the buses voltages.

4.2. Save

In this section, the scenario's results are received, and each element's results are sorted according to the spreadsheet's format. Then, the sorted results of each element (which are data frames) are added to each scenario's results. In the end, when all the elements of the current scenario have been sorted and saved, the scenario is added to the dictionary of scenarios, where each scenario's results are stored. The mentioned process is depicted in Figure 18.

4.3. Create Spreadsheet & Plot topology

In this section the spreadsheet's output is created. The function is called after all scenarios were calculated and saved. For that, the function receives the dictionary with all scenarios

results, and iterate over each scenario, and element. The elements are organized according to its type e.g., generation includes: ‘generators’, ‘static generators’, and ‘external grid’.

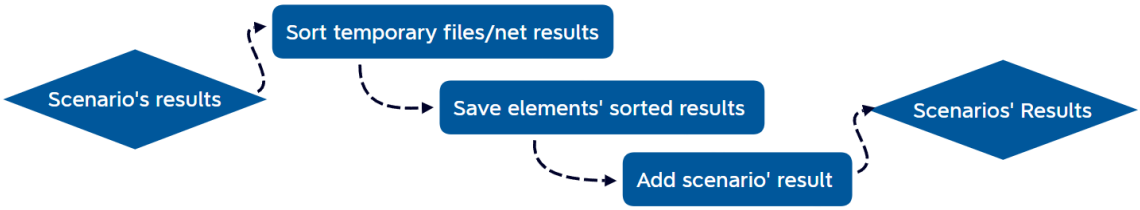


Figure 18: Save's Workflow

Then, they are written in the excel sheet according to its type; that in order to have the same type of elements in one sheet, for easier visualization and post processing. In the end, all the data is stored in one large data frame, which is written in the ‘data’ sheet. Figure 19 shows the workflow for this section. The data sheet has two functions: 1) all the results are accessible in a tabular format, hence in case the user wants to do further post-processing, just one table should be read; 2) it simplifies the macros for creating the dashboard. Finally, after the spreadsheet is created, in order to observed the topology, the toolbox will show a simple plot of the network.

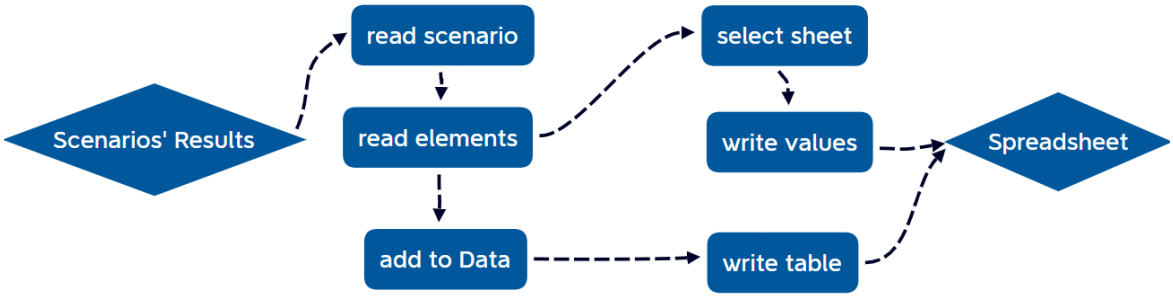


Figure 19: Create Spreadsheet's Workflow

4.3.1. Results

This subchapter includes the output that is obtained from running the tool. At the beginning, the spreadsheet includes seven sheets, that are: Summary, Demand, Generation, Buses, Lines, Trafos, and Data. Moreover, the dashboard sheet is created if the user executes the excel’s macro to do it.

The first sheet is the summary, where the button to create the dashboard is located, as can be seen in Figure 20. Further, the sheet includes buttons to display or hide the sheet with pivot tables needed for the dashboard. And after performing the analysis function on the result, the over- and undervoltage violation, and over loaded components will be written. Further explanation of Analysis function will be at section 4.4.

	Time Step	Bus Index	Scenario	Under Voltage [p.u]	Time Step	Line Index	Scenario	Loading Percentage[%]	Time Step	Trafo Index	Scenario	Loading Percentage[%]
0	0	5	basic	0.9787	---	---	---	---	---	---	---	---
1	0	6	basic	0.9772								
2	0	7	basic	0.9764								
3	0	1	pv	0								
4	0	2	pv	0								
5	0	3	pv	0								
6	0	4	pv	0								
7	0	5	pv	0								
8	0	6	pv	0								
9	0	7	pv	0								
10	1	4	pv	0.9789								
11	1	5	pv	0.9763								
12	1	6	pv	0.9746								
13	1	7	pv	0.9737								

Figure 20: Summary Sheet of the Spreadsheet Output

Once the Dashboard has been created, the user can visualize all the scenarios results in it, this can be done by selecting the scenario on the first box on the left of the dashboard. Figure 21 shows an example of a Dashboard, and in this case the scenario displayed is called ‘pv’. Further, the next box allows the user to select what zones of the pandapower network should be displayed in the dashboard, in this case the user did not add any name to the zones; therefore, ‘---’ was added by default. The last box that is shown in Figure 21 is ‘fuel’, this allows the users to filter the generation that is displayed. In the center of the sheet are two charts, on the top the generation labeled by zones is shown, and on the bottom the generation labeled by fuel. Finally, the last two charts on the right show the total generation share by zones and fuel, respectively.

Besides the generation, the dashboard also includes the results of the transmission’s lines and transformers. Figure 22 shows the last four charts and the last box to filter the results. The last box allows the users to select a specific time step. The only box that does not filter all the results is the third, which filters the generation’s fuel source, then all the rest are applied to all the charts as soon as the users selects a scenario, zone, or time step. Regarding the charts in this part, the chart in the middle shows the total losses in the lines and transformers, the one on the right show the share of the losses considering whether they happen in the lines or transformers. The last charts show the maximum, average and minimum loading percent of the lines and transformer in the network.

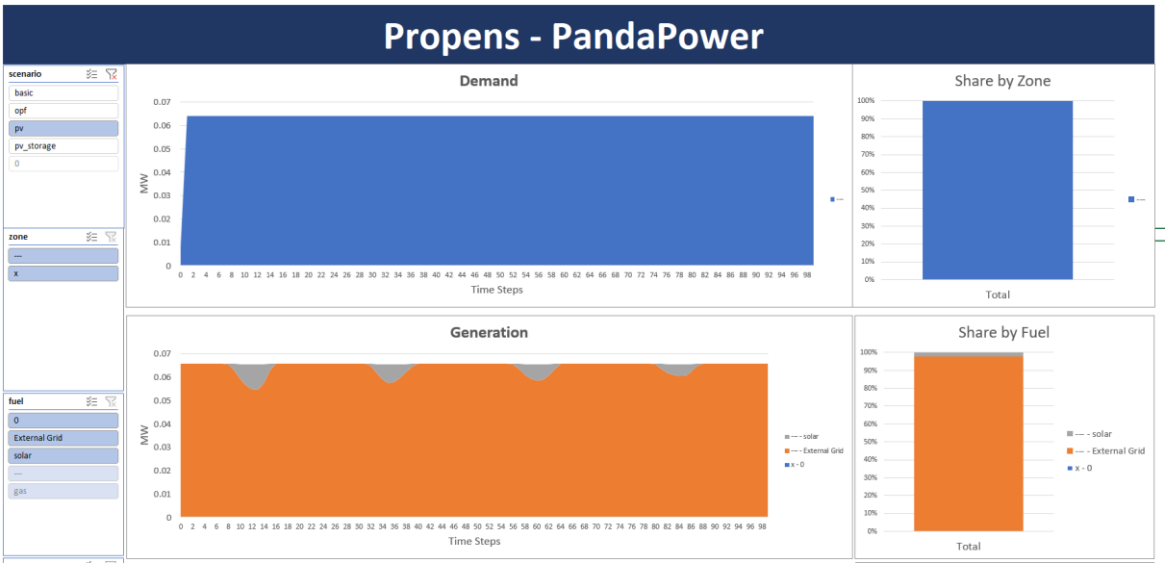


Figure 21: Dashboard View 1/2

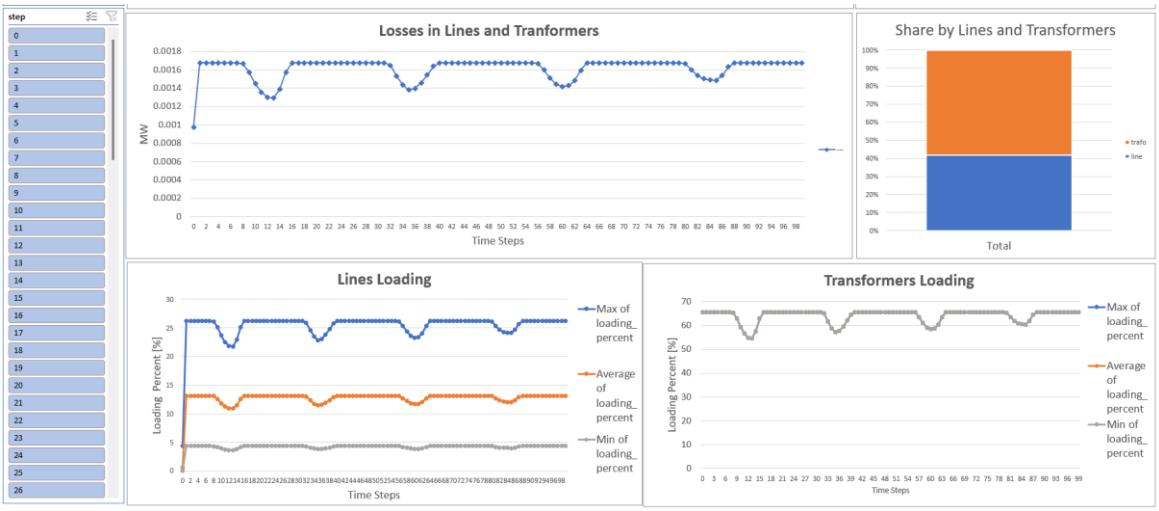


Figure 22: Dashboard View 2/2

4.4. Analysis Function

The module anal_fun.py is built based on the class, for easier modification. At first it was building simple power flow analysis without taking care of time step. After successfully implementing time series calculation by co-working with front-end/database team, analysis function also needed to be updated. By inheriting the parent class of *pd_Analysis* into *pd_ts_Analysis*, the transition happened smoothly. Unfortunately, most of functions must be modified differently since the format changed while developing further. Currently, *pt_ts_Analysis* class is available for both time series analysis and just simple one-time

power flow analysis, due to unified information template of the result file. During the development, the analysis function must be modified whenever the excel output function changes. However currently, this module works completely independent from the creating result to exporting it to excel, since the only input it reads are the outcome that is already created.

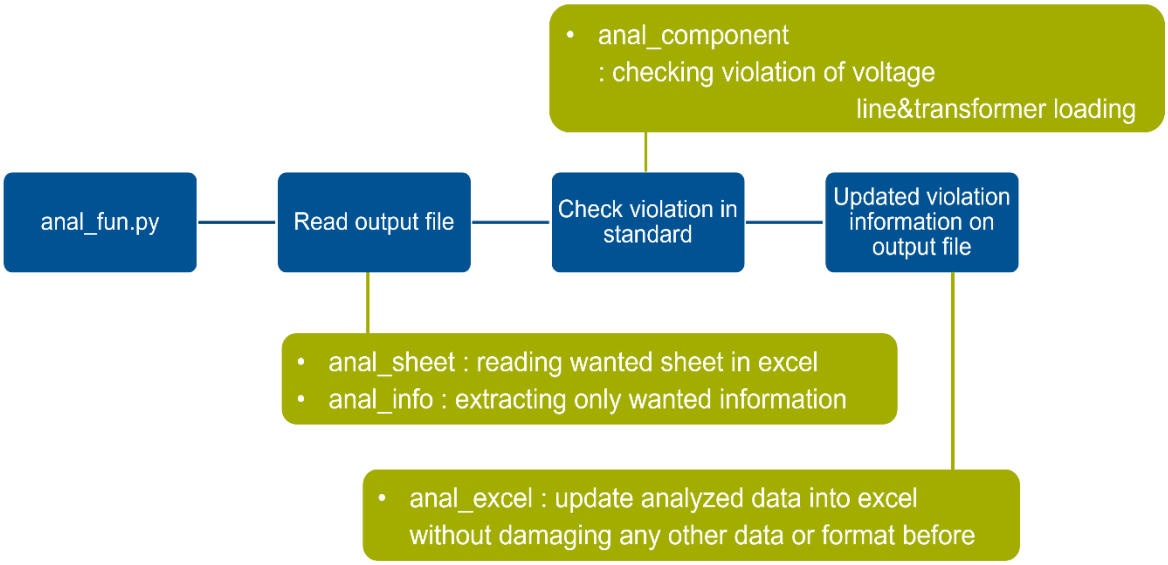


Figure 23: Work Flow of Analysis Function

In Figure 23, the scheme of analysis function is depicted. After the results are obtained, then the excutor.py will call the analysis function. When the function is called, then firstly, *anal_sheet* function will try read the excel file from the result folder. And then *anal_info* function will extract related information only and return it as data frame(pandas) format.

Currently, analysis has been done for 3 parts. Voltage violation in the buses, overloading of the line and transformers. The standard value of violation can be arbitrary chosen. From the result in Figure 20, we choose the voltage violation as ±2% from the nominal value, 100% for Transformer loading and 100% for Line loading, to check if all analysis function is worked properly. This value can be easily changed by user based on their needs. In case there's no violation, '---' line will be inserted, which can be seen in the Figure 20, next to the voltage violation summary.

Voltage violations check the value of the voltage in each bus are higher or lower than the standard values in per unit. It first creates empty data frame(pandas) and add the data in case of violation is happened. The information written is the time step, the index of the bus, and the value of the voltage. If there are both under and over voltage violation happened,

in left side there will be undervoltage information will be written and overvoltage information at the right-hand side.

Same procedure will happen for the over loading violation of the lines and the transformers. However, in overloading case, we only check if the loading is exceeding 100% or not, because there's no under loading issue in power line, in case of safety.

After each violation criteria are checked, then finally *anal_excel* function will be called and it will update the values in the result file. During this time, all the other features of result file, such as Dashboard function and macros from section 4.3.1 will remain untouched. The result of the summary can be found in Figure 20.

5. Testing & Error Management

5.1. Test Environment

To ensure the code's lifetime and reliability, tests were implemented for the toolbox. The used test environment is the unit test. For each function in the source code, a test case is implemented as a class with multiple test functions. The test coverage summary is shown in Table 2. (Stand 01.02.2023)

Table 2 Summary of test coverage

Module	*statements*	*missing*	*excluded*	*coverage*
src__init__.py	0	0	0	100%
src\analysis__init__.py	0	0	0	100%
src\analysis\excel_output.py	72	9	0	88%
src\analysis\parameters.py	85	19	0	78%
src\analysis\plot.py	25	17	0	32%
src\analysis\save.py	51	5	0	90%
src\analysis\solver.py	42	8	0	81%
src\analysis\time_series_func.py	42	12	0	71%
src\frontend__init__.py	0	0	0	100%
src\frontend\generate_timeseries.py	33	1	0	97%
src\frontend\read_input.py	19	0	0	100%
src\scenarios__init__.py	0	0	0	100%
src\scenarios\apply_scenario.py	6	0	0	100%
src\scenarios\scenarios.py	45	9	0	80%
test__init__.py	0	0	0	100%
test\test_excel_output.py	36	1	0	97%
test\test_generate_timeseries.py	14	1	0	93%
test\test_parameters.py	42	1	0	98%
test\test_read_input.py	52	1	0	98%
test\test_save.py	59	1	0	98%
test\test_scenarios.py	41	1	0	98%
test\test_solver.py	90	1	0	99%
test\test_time_series_func.py	41	1	0	98%
Total	**795**	**88**	**0**	**89%**

5.2. Error Management

To make debugging easier for users while using the toolbox, an error management system is developed in the executor. The system gives the user error messages by adding “try … except …” statements while calling each function.

Figure 24 shows the overview of the error management system. The blue blocks are all functions called in the executor. The red blocks are returned error messages from the system. If the previous function runs successfully, the program will go to the next function (next blue block). If an error is raised in the previous function, the program will give the corresponding error message (red block beside), then stop the toolbox. Using this system, the user can know which part of the code went wrong and have a more detailed direction to debug.

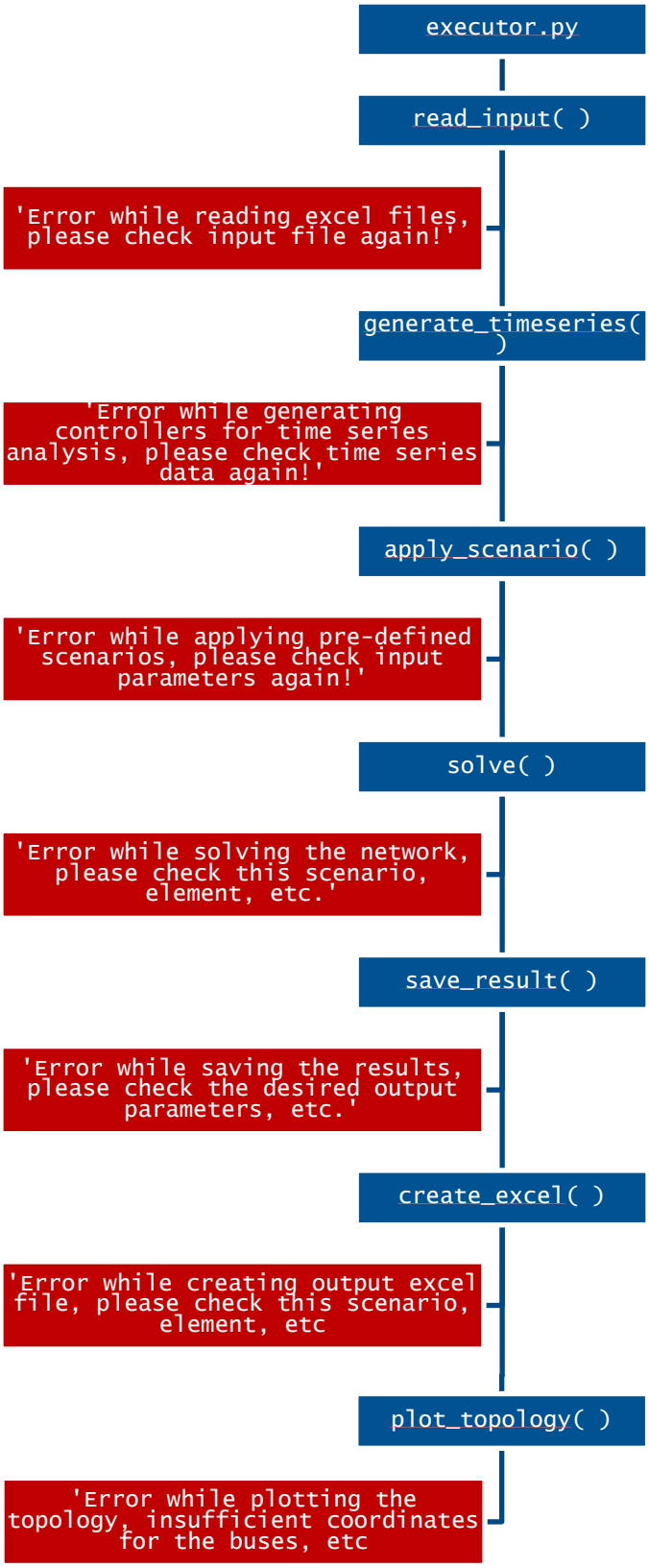


Figure 24: Error Management System

List of Figures

List of Tables