pandapower - an Open Source Framework for Automated Evaluations of Future Power Systems

Leon Thurner
Department of Energy Management
and Power System Operation
University of Kassel, Germany
Email: leon.thurner@uni-kassel.de

Alexander Scheidler
Martin Braun
Fraunhofer Institute for Wind Energy
and Energy System Technology
Kassel, Germany

***Abstract*—pandapower is new open source power system analysis tool that is well suited for applications with a high degree of automation. This paper introduces pandapower and demonstrates a typical work flow of defining and analysing electric power systems. As an exemplary application of pandapower, a simple algorithm that analyses the hosting capacity of a distribution system is introduced. In addition to that, case study results of hosting capacity investigations that were carried out in cooperation with major network operators in Germany and Switzerland are presented.**

I. Introduction

The Indian Ministry of New and Renewable Energy Sources has issued the goal to raise the installed power of renewable energy resources from currently below 43 GW to 175 GW by the year 2022 [1]. Germany has already undergone a similarly steep ascent in renewable generation installation by increasing the installed power of renewable energy sources from under 10 GW in the year 1999 to over 100 GW in the year 2016 [2]. This massive increase of distributed generation units constitutes a paradigm shift in electric power generation and distribution and raises the complexity of power system operation, analysis and planning. This is especially relevant in distribution systems, where a majority of DG units are installed. The evaluation of future power system scenarios often requires probabilistic methods with a high demand for automation. This paper introduces the new open source tool pandapower, which focuses on easy to use power system analysis with a high degree of automation (Section II). As an exemplary application of pandapower, we present a probabilistic analysis of the photovoltaic (PV) hosting capacity in distribution systems based on pandapower (Section II). Finally, we present case study results of hosting capacity investigations that were carried out in cooperation with major network operators in Germany and Switzerland (Section V).

II. pandapower

pandapower is a new open source power system analysis tool available under a BSD license. It is implemented in Python, guaranteeing free availability and flexible expansion with other open source libraries. Since Python itself is open source, applications in pandapower can be parallelized without any license constraints. This makes it well suited for scientific applications which often rely on high performance computing.

pandapower comes with an extensive library of electric elements, such as ZIP loads, lines, transformer or switches (see Section II-B). Based on these network models, pandapower allows to carry out power flow, optimal power flow, state estimation and short circuit calculations as well as topological graph search (see Section II-C).

A. Data Structure

pandapower is based on a tabular data structure, where every element type is represented by a table that holds all parameters for a specific element and a result table which contains the element specific results of the different analysis methods. The tabular data structure is based on the Python library pandas [3]. It allows the storage of variables of any data type, so that electrical parameters can be stored together with status variables and meta-data, such as names or descriptions. The tables can be easily expanded and customized by adding new columns without influencing the pandapower functionality. All inherent pandas methods can be used to efficiently read, write and analyse the network and results data. A pandapower network (in the following abbreviated as 'net') is a Python dictionary that holds all information about the network. It includes element and a result tables for each element type, such as lines, transformers, switches etc. (see Section II-B). Element tables hold all input parameters that are specified by the user, while the result table is used by power flow or optimal power flow functions to store the results. Besides the element tables the net data structure also includes standard type data and network wide parameters like frequency, network name or rated apparent power for the per unit system.

B. Element Models

The electric models and equivalent circuits, representing the different elements, are described in this Section. Detailed formulas for the calculation of the electric parameters are also available in the pandapower documentation.[1]

1) Bus: Buses represent the nodes in the network. All other elements are connected to one or more buses. With the element based network model, it is possible to connect multiple loads or generators to the same bus. The rated voltage of the buses defines the voltage levels and constitutes the reference framework for the per unit system.

2) Load: Loads are used to model electric consumption. pandapower includes a ZIP model that allows modeling loads with constant power, constant current or constant impedance.

[1] https://pandapower.readthedocs.io

3) Static Generator: Static generators are used to model constant power injection. Since all nodal power is given from a consumer viewpoint, the power generation has to be defined with a negative active power generator. This might seem unintuitive for generator type elements, but the consistent convention makes the definition of power values unambiguous even for elements where the signing is not obvious, such as external grids, shunts or buses.

4) Generator: Generator elements model power generation units with a fixed active power injection at a fixed voltage magnitude in the power flow calculation. If reactive power limits are defined for the generator, the voltage set point might not always be reached in a power flow calculation.

5) External Grid: The external grid element represents a voltage source with fixed voltage magnitude and voltage angle. Adherence with the complex voltage set point in the power flow calculation is achieved by setting the generator bus as a slack node.

6) Shunt: Shunts are network elements that can be used to model capacitor banks or choke coils. Shunts are specified by their power values at rated voltage. The shunt model includes a step model which allows to subdivide the shunt power.

7) Line: The line element is used to model cables or overhead lines with a π equivalent circuit. It has two different categories of parameters: parameters that depend on one specific line (e.g. line length or the buses which it connects) and parameters that depend on the type of line which is used (e.g., the impedance and capacity per kilometer or maximal thermal current). `pandapower` comes with a standard type library that allows the creation of lines using the element specific parameters and loading the type specific parameters from a standard type data base. The user can either define individual standard types or use the predefined `pandapower` basic standard types for convenient creation of networks. The line loading is calculated as the ratio of the current through the line to the maximal thermal current of the line.

8) Two-Winding Transformer: The transformer element is used to model power transformers that connect two different voltage levels. Transformers are commonly modeled with a T equivalent circuit [4]. However, for the sake of completeness, `pandapower` also allows to model transformers with a π-transformer model.

Just as for lines, the transformer model differentiates between element specific parameters, such as the connected buses or the number of parallel transformers, and type parameters, such as the short circuit voltage, open loop losses, rated voltage and rated power. Transformers can then be created through the standard type library by specifing only the connected buses and a standard type, from which the electric parameters are loaded. The transformer element includes a tap changer model that allows to change the transformer ratio and angle shift with the position of the tap. The transformer loading can either be calculated in relation to the rated power or the rated current of the transformer.

9) Three-Winding Transformer: The three-winding transformer Three-winding transformers can be modeled by three two-winding transformers in wye connection [4]. The three-winding transformer model in `pandapower` does this conversion internally. Just as for two-winding transformers, there is a standard type library and a tap changer model for three-winding transformers.

10) Switch: The switch element allows modeling of ideal switches. It supports switches between two buses or between a bus and a branch element (line or transformer). A closed bus-bus switch galvanically couples two buses without a voltage drop, which is achieved by internally fusing the buses. Branch elements that are connected to a bus through an open switch are in open loop operation, which means their open loop losses are still considered.

11) DC Transmission Line: A DC transmission line transmits active power between two buses. The transmission is reduced by absolute transformation losses at the stations and relative transmission losses on the line. The transformer stations at both ends of the line can be used for reactive power injection similar to the generator element.

12) Network Equivalents: `pandapower` includes three models that are commonly used as network equivalents in network reductions: impedance, ward and extended ward equivalents. An impedance element connects two buses with an impedance. The impedance can be unsymmetrical, so that forward impedance and backward impedance have different values. The rated voltage of the connected buses can also differ. The ward equivalent is a combination of a constant apparent power consumption and a constant power load. The extended ward equivalent includes an additional voltage source with internal impedance.

C. Network Analysis

`pandapower` provides a wide range of electric and topological network analysis functionality.

1) Power Flow: The `pandapower` power flow algorithm is originally based on PYPOWER, but has been improved with respect to robustness, runtime and usability. The Newton-Raphson solver [5] is accelerated with just-in-time compilation trough the Python library numba. The power flow can be initialized from a DC power flow or from previous results to improve convergence. A connectivity check allows the power flow to converge even if some areas are not connected to an external grid.

2) Optimal Power Flow: `pandapower` allows solving AC and DC optimal power flow (OPF) problems through an interface to the PYPOWER interior point solver [6]. Costs, flexibilities and constraints can be configured through the `pandapower` data structure.

3) Short Circuit Calculation: The short circuit module allows calculation of short-circuit currents according to IEC 60909 [7]. The correction factors and calculation rules specified in the standard are applied automatically when carrying out a short circuit calculation. The module allows to calculate symmetrical three-phase and unsymmetrical two-phase short circuit currents.

4) State Estimation: `pandapower` includes an implementation of a state estimation with weighted-least-square approach. The state estimation module supports voltage, active power and reactive power measurements at buses, lines and transformers. It includes a bad data detection based on a χ^2 test and a normalized residual test that allows to detect and remove bad measurements [8].

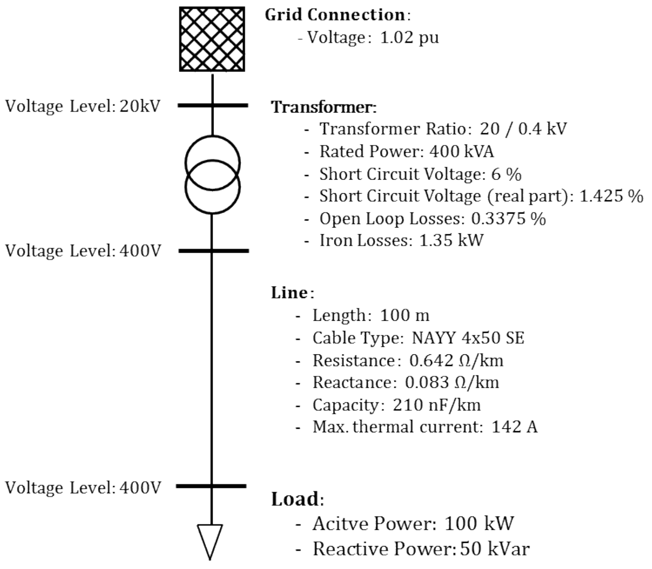
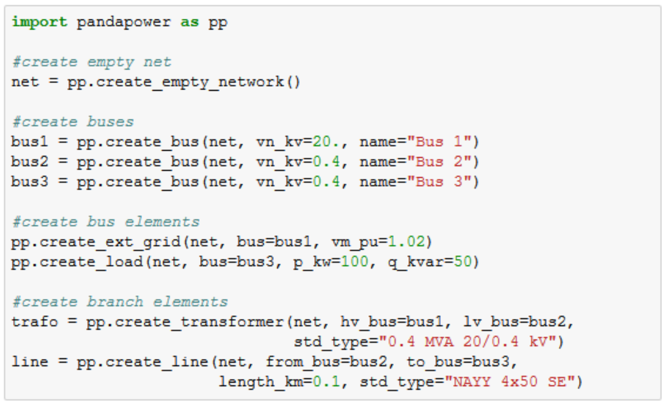


Fig. 1. Minimal Example Network


```
import pandapower as pp

#create empty net
net = pp.create_empty_network()

#create buses
bus1 = pp.create_bus(net, vn_kv=20., name="Bus 1")
bus2 = pp.create_bus(net, vn_kv=0.4, name="Bus 2")
bus3 = pp.create_bus(net, vn_kv=0.4, name="Bus 3")

#create bus elements
pp.create_ext_grid(net, bus=bus1, vm_pu=1.02)
pp.create_load(net, bus=bus3, p_kw=100, q_kvar=50)

#create branch elements
trafo = pp.create_transformer(net, hv_bus=bus1, lv_bus=bus2,
                              std_type="0.4 MVA 20/0.4 kV")
line = pp.create_line(net, from_bus=bus2, to_bus=bus3,
                      length_km=0.1, std_type="NAYY 4x50 SE")
```

Fig. 2. Creating the Minimal Example Network

5) *Topological Graph Searches:* pandapower provides the possibility of graph searches using the Python library NetworkX [9]. The topology module provides the possibility to translate pandapower networks into abstract NetworkX graphs and carry out graph searches to analyze the network structure. Additionally, the module also provides some pre-defined search algorithms to tackle common graph search problems in electric networks, such as finding all supplied buses or identifying buses on main or secondary network feeders.

III. Minimal Example

To demonstrate the pandapower workflow a minimal example is presented in this section. The example shows how networks are created and analyzed using the pandapower API. The code for this example is available as an interactive jupyter notebook on github[2].

A. *Creating a Network*

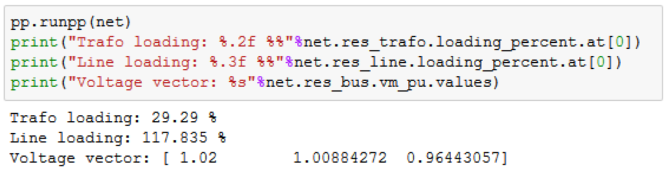
All elements can be added to the network with a corresponding create_* function (e.g., create_bus, create_line, ...). This allows a convenient step-by-step definition of networks. However, it is also possible to introduce new elements by directly editing the element tables.

Note that, in addition to creating networks through the API, over 70 predefined networks can be directly accessed through pandapower. These include the well-known IEEE power system test cases [10], benchmark networks from CIGRE [11] as well as generic medium and low voltage networks.

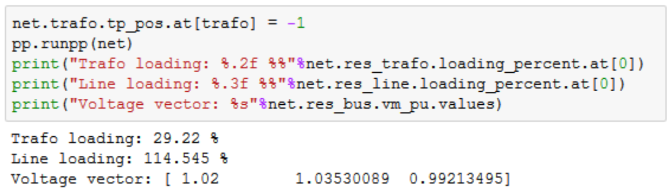
The example network depicted in Figure 1 consists of three buses, a line, a transformer, an external grid connection and a load. The source code that creates this network in pandapower is shown in Figure 2. First, an empty network is initialized and the three buses with their respective voltage levels are created. Then the branch elements that connect the buses are added. The line and transformer elements are created with the help of the standard type library. The electric parameters of the line type "NAYY 4x50 SE", like the resistance or reactance per kilometer, are loaded from pandapower's standard type library and written into the line table (see Figure 4). The transformer is created in the same way with the standard type "0.4 MVA 20/0.4 kV". Finally, the load element at Bus 3 and the external grid connection at Bus 1 are created. All elements are identified by an index that is unique within the element table. All create functions return the index of the created element, which can be used to identify the element.

B. *Running a Power Flow*

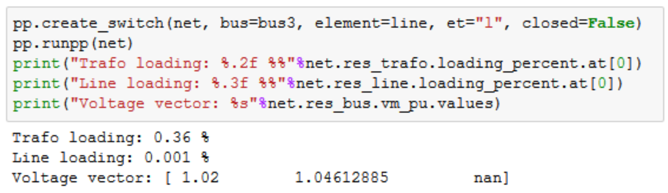
For the created network pandapower's runpp function calculates the power flow and stores the results into the result tables with the prefix "res_" for each element (e.g., res_bus, res_line ...). Figure 3 shows how the results can be extracted and the respective console output. It can be seen, that the transformer in the example network is loaded with about 29 % and the line is loaded with about 117 %. The voltage


```
pp.runpp(net)
print("Trafo loading: %.2f %%"%net.res_trafo.loading_percent.at[0])
print("Line loading: %.3f %%"%net.res_line.loading_percent.at[0])
print("Voltage vector: %s"%net.res_bus.vm_pu.values)
```
```
Trafo loading: 29.29 %
Line loading: 117.835 %
Voltage vector: [ 1.02        1.00884272  0.96443057]
```

(a) Extracting results of a power flow


```
net.trafo.tp_pos.at[trafo] = -1
pp.runpp(net)
print("Trafo loading: %.2f %%"%net.res_trafo.loading_percent.at[0])
print("Line loading: %.3f %%"%net.res_line.loading_percent.at[0])
print("Voltage vector: %s"%net.res_bus.vm_pu.values)
```
```
Trafo loading: 29.22 %
Line loading: 114.545 %
Voltage vector: [ 1.02        1.03530089  0.99213495]
```

(b) The tap position of the transformer has effect on the low voltage buses


```
pp.create_switch(net, bus=bus3, element=line, et="l", closed=False)
pp.runpp(net)
print("Trafo loading: %.2f %%"%net.res_trafo.loading_percent.at[0])
print("Line loading: %.3f %%"%net.res_line.loading_percent.at[0])
print("Voltage vector: %s"%net.res_bus.vm_pu.values)
```
```
Trafo loading: 0.36 %
Line loading: 0.001 %
Voltage vector: [ 1.02        1.04612885         nan]
```

(c) Opening a switch disconnects a bus

Fig. 3. Running a power flow and inspecting the results

[2]https://github.com/lthurner/pandapower/tree/develop/tutorials

	name	std_type	from_bus	to_bus	length_km	r_ohm_per_km	x_ohm_per_km	c_nf_per_km	max_i_ka	df	parallel	type	in_service
0	Line	NAYY 4x50 SE	1	2	0.1	0.642	0.083	210.0	0.142	1.0	1	cs	True

Fig. 4. Line Table for minimal example network

drops from 1.02 at the external grid bus to 0.9644 p.u. at the load bus. Of course the operational state of the network, such as the switching configuration or tap changer positions, influence the results of the power flow. Figure 3b shows how a change in the tap changer position can be modelled in `pandapower` and how this affects the power flow results. The parameters of the tap changer, such as the voltage step for each tap position, are part of the standard type data that was loaded from the standard type library. After the tap is changed from the default position 0 to -1, the voltage at the low voltage side of the transformer increases. The loading of the line and the transformer also changes slightly due to changes in the transformer losses. In 3c, an open switch is created at the end of the line, which separates the load bus from the grid connection. Since the load is now unsupplied, the line and transformer are in open loop operation and their loading drops to nearly zero. The voltage at the load bus is given as nan, since the bus is not connected to any voltage source.

C. Evaluating topology

The `pandapower` topology package allows to analyse the structure of the network. A typical graph search application is to find which buses are not supplied in the current state of the network. `pandapower` provides predefined search functions for this and other purposes. For the minimal example network with the opened switch, the unsupplied_buses functions correctly identifies that Bus 2 is cut from power supply (see Figure 5a). Other predefined search functions exist to find primary and secondary feeders or to calculate the distance between buses. Apart from using predefined search functions, `pandapower` also allows to translate the network into an NetworkX graph and run individual graph searches. An example can be seen in Figure 5b, where all buses connected to bus 2 are found. The include_trafos options is set to False, which means that transformers are not translated into edges in the graph. Therefore, only buses 1 and 2 are found as connected component of bus 2. Similar searches can for example be used to identify the area supplied by one transformer or by one feeder. Another common application in radial systems is to find cycles in networks, for which NetworkX provides efficient algorithms.

```python
import pandapower.topology as top
top.unsupplied_buses(net)
```

```
{2}
```

(a) Identifying unsupplied buses

```python
net.switch.closed.at[0] = True
mg = top.create_nxgraph(net, include_trafos=False)
list(top.connected_component(mg, 2))
```

```
[2, 1]
```

(b) Identifying galvanically connected components

Fig. 5. Topological evaluations with the `pandapower` topology module

D. Running a Short Circuit Calculation

To run a short circuit calculation, the short circuit power and r/x ratio of the external grid have to be specified as additional parameters compared to the power flow calculation. Figure 6 shows how the values are specified and short circuit currents at all buses are calculated. Here, the maximum short circuit currents for faults at each bus are calculated. The correction factors for the voltage source and transformer are automatically applied according to the IEC 60909 standard. The calculation returns current values for initial short circuit current `ikss_ka` and peak short circuit current `ip_ka`.

```python
import pandapower.shortcircuit as sc
net.ext_grid["s_sc_max_mva"] = 100
net.ext_grid["rx_max"] = 0.1
sc.calc_sc(net, case="max", ip=True, r_fault_ohm=2.)
print(net.res_bus_sc)
```

```
    ikss_ka     ip_ka
0  2.534707  4.317318
1  0.126631  0.182666
2  0.122698  0.176991
```

Fig. 6. Calculating short circuit currents according to IEC 60909

IV. Hosting Capacity

The term PV hosting capacity is defined as the maximum PV capacity which can be connected to a specific distribution grid, while still complying with relevant grid codes and grid planning principles. The automated analysis of the hosting capacity of a large number of grids as part of PV integration studies can answer questions like [12]:

- Are the distribution grids overall well dimensioned for future PV deployment?
- Which or how many distribution grids require additional measures for PV integration?
- When will additional measures for PV integration be required?

In this section, we present a simple algorithm to determine the hosting capacity of a network with `pandapower`.

A. Hosting Capacity Distribution

The spatial distribution and the rated power of the PV systems have a significant impact on the hosting capacity, but are unknown beforehand. It is therefore hard to reflect the hosting capacity of a network by a single number. Instead, it is rather characterized by a statistical distribution of values for different scenarios. The distribution can be evaluated with a Monte-Carlo approach that statistically varies interconnection points and rated power of the PV systems.

A resulting distribution of maximum installable PV systems for a specific low voltage grid is depicted in the middle of Figure 7. It was generated by simulating 50 different possible future PV installation scenarios. For every scenario (varying connection points and system sizes) the maximum number of PV system that can be additionally installed without leading to a violation of constrains (voltage band, limit for maximal line loading) is analyzed. As can be seen, the hosting capacity varies between 8 and 15 additionally installable PV plants. A boxplot visualizes the resulting distribution and helps to understand the behaviour of the grid when additional PV systems will be installed. The whiskers show the maximum and minimum of PV plants that can be installed before the first problem occurs. The box shows the range that incldudes 50% of all values (quartile) and the vertical line marks the median of the distribution. The example in Figure 7 shows two installation scenarios in both ends of the spectrum: in the network plot shown above, 15 PV plants are connected to the system without any constraints being violated. In the network plot shown below, a voltage violations occurs after only connecting 9 PV plants. It can be seen that the PV plants in the network above are evenly distributed on different feeders and at different distances from the transformer station, which leads to a balanced line loading and voltage drop. In the network below, the installed PV plants are concentrated at the end of the longest feeder in the network, which leads to a higher voltage drop over this feeder and subsequently to a voltage violation. The boxplot thereby not only visualizes how many PV plants can be installed, but also shows how much the hosting capacity depends on the different connection points. This is important in order to evaluate uncertainty in future scenarios.

B. *Evaluating Hosting Capacity with* `pandapower`

As the example above has shown, hosting capacity can only be analyzed by evaluating a large number of networks and scenarios. `pandapower` is well suited to tackle this task because it is optimized for automated evaluations. Furthermore, the detailed element models of `pandapower` allow a convenient evaluation of constraint violations, such as line or transformer loading and voltage band violations.

In the following we will show a simple example of how to calculate a hosting capacity distribution with `pandapower`. The code for this example is also available as an interactive jupyter notebook on github[3].

The main function which is used to calculate a hosting capacity is shown in Figure 8. After a network is loaded, PV plants are installed iteratively until a constraint violation occurs. The installed power and the type of constraint violation is then saved and the algorithm is restarted until a fixed number of iterations is reached.

This basic algorithm depends on three functions: one function that defines and evaluates constraint violations, one function that returns the size of the PV plant and one that chooses the connection bus for the PV plant.

1) Constraint Violations: An example for a function that evaluates constraint violation is shown in Figure 9. The function runs a power flow and then checks for line loading and transformer loading, both of which must not exceed 50 %, and for voltage rise, which must not exceed 1.04 pu. The function returns a boolean flag to signal if any constraint is violated as well as a string that indicates the type of constraint violation.

2) PV Plant Size: An example for a function that returns a plant size is shown in Figure 10. It returns a random value from a normal distribution with a mean of 500 kW and a standard deviation of 50 kW. Such a distribution can be seen in Figure 11. Depending on the existing information, it is of

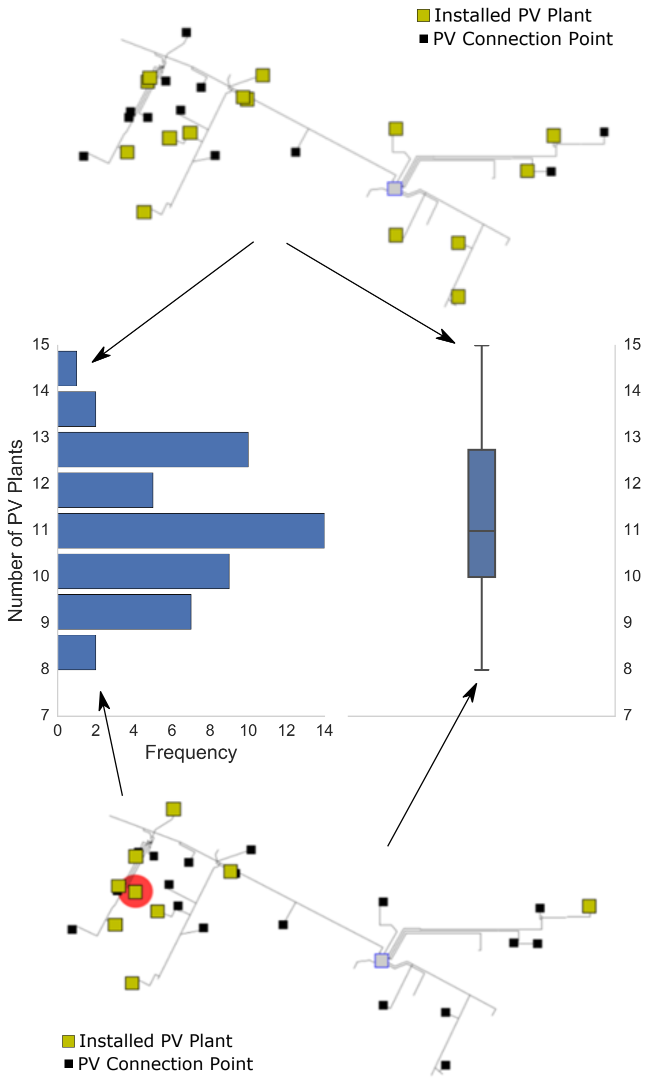
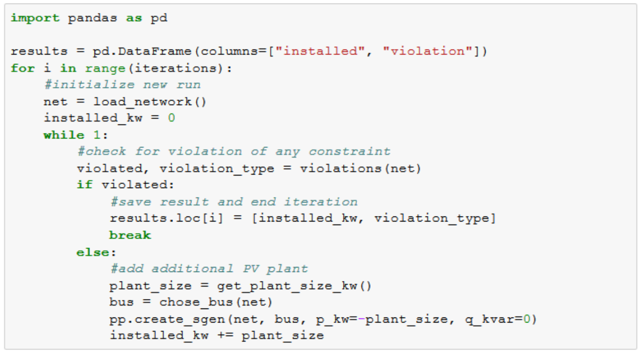


Fig. 7. Example for hosting capacity distribution

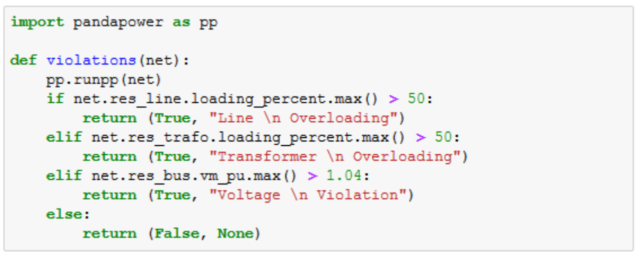


```
import pandas as pd

results = pd.DataFrame(columns=["installed", "violation"])
for i in range(iterations):
    #initialize new run
    net = load_network()
    installed_kw = 0
    while 1:
        #check for violation of any constraint
        violated, violation_type = violations(net)
        if violated:
            #save result and end iteration
            results.loc[i] = [installed_kw, violation_type]
            break
        else:
            #add additional PV plant
            plant_size = get_plant_size_kw()
            bus = chose_bus(net)
            pp.create_sgen(net, bus, p_kw=-plant_size, q_kvar=0)
            installed_kw += plant_size
```

Fig. 8. Example for algorithm to calculate hosting capacity

[3]https://github.com/lthurner/pandapower/tree/develop/tutorials

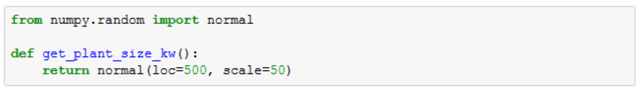


```python
import pandapower as pp

def violations(net):
    pp.runpp(net)
    if net.res_line.loading_percent.max() > 50:
        return (True, "Line \n Overloading")
    elif net.res_trafo.loading_percent.max() > 50:
        return (True, "Transformer \n Overloading")
    elif net.res_bus.vm_pu.max() > 1.04:
        return (True, "Voltage \n Violation")
    else:
        return (False, None)
```

Fig. 9. Example for a function that evaluates constraint violations

course also possible to use other probability distributions, such as a Weibull distribution, or to draw values from existing plant sizes.



```python
from numpy.random import normal

def get_plant_size_kw():
    return normal(loc=500, scale=50)
```

Fig. 10. Example for a function that returns randomized plant sizes

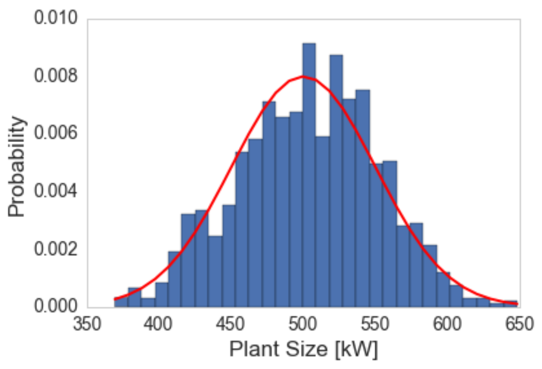
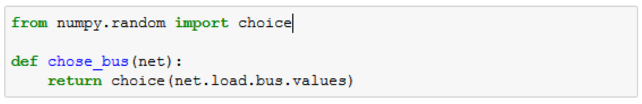


Fig. 11. Example for a plant size distribution

3) Selection of Connection Point: An example for a function to select connection buses is shown in Figure 12. This implementation assumes all buses with loads are possible connection points for future PV installation. It therefore returns one random bus of all buses with a connected load. In this implementation, all buses are chosen with the same probability. Depending on what information is available, it is also possible to define probabilities for each bus, for example depending on the size of a MV/LV transformer station or on geographical parameters. Furthermore, one bus can be selected multiple times in this implementation. It is also possible to either restrict the maximum installed power at one bus or to only allow one installation per bus. The latter is especially relevant in LV networks, where one connection point represents one household that is not expected to have multiple PV plants.



```python
from numpy.random import choice

def chose_bus(net):
    return choice(net.load.bus.values)
```

Fig. 12. Example for a function that returns randomized connection points

The algorithm shown in Figure 8 is carried out for the generic network MV Oberrhein which can be found in the networks package of `pandapower`. 50 different PV scenarios were calculated and the definitions of violations, plant size and bus selection were used as given in Figures 9 - 12. The results show, that the hosting capacity of this network ranges from 6.5 MW to 14.5 MW overall (see Figure 13). It is between 9.5 and 11.5 MW in 50% of the runs and the median is at about 11 MW of additional PV capacity. The results also allow a conclusion about the potential problems that arise due to connection of additional PV plants. Here, the reason why no further PV plants can be installed is an overloading of a transformer in 78 % of cases, an overloading of a line in 18 % of the cases and a violation of the voltage band in 4 % of the cases.

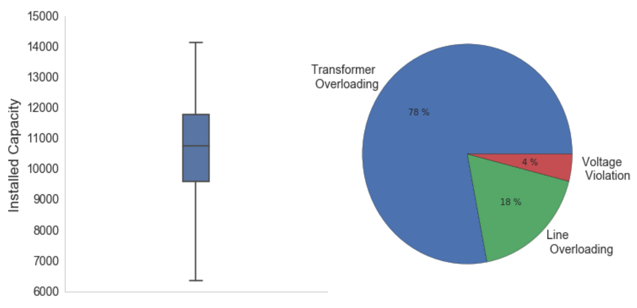


Fig. 13. Results for an example run of hosting capacity evaluation: distribution of installed power and percentage of constraint violations that occurred

V. Case Studies

A. Comparing Hosting Capacity to Expected PV Expansion

Within the scope of a PV integration study carried out for the Swiss distribution system operator Romande Energie, the hosting capacity of 111 LV networks were evaluated and compared to different PV forecast scenarios provided by the DSO. Figure 14 shows a comparison of the PV hosting capacity (green boxplots) and the expected PV installations 2035 (red boxplots) in the most progressive scenario for the LV level. The expected PV capacity 2035 exceeds the determined PV hosting capacity only in a very few LV grids. Therefore, the LV grids are mostly well dimensioned for the expected additional PV installations and additional measures for PV integration are just expected in a few LV grids.

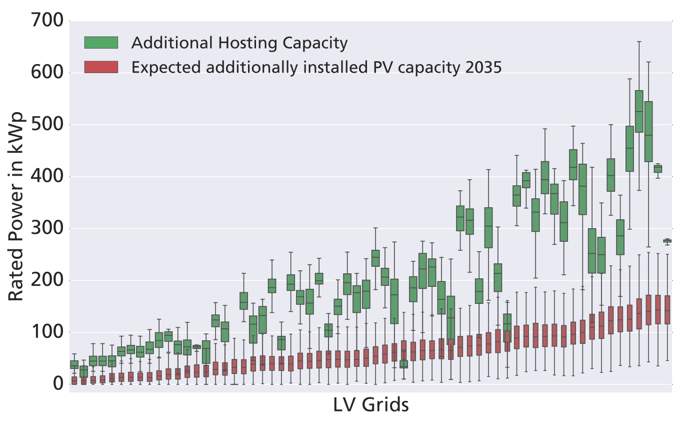
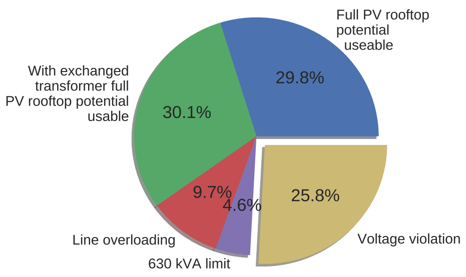


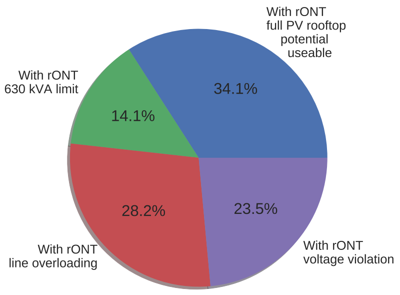
Fig. 14. Example for hosting capacity distribution

B. Technical and Economical Assesment of MV/LV OLTC transformer

In LV grids the application of an On Load Tap Changer (OLTC) for the MV/LV transformer (abbreviated here as rONT, from the German regelbarer Ortsnetztransformator) is a promising technical solution to increase the PV hosting capacity. In a grid integration study in cooperation with the German DSO Bayernwerk the technical and economic potential of the rONT was analysed for 329 LV grids. A simplified analysis of the PV rooftop potential forms the basis of this PV integration study. First of all, the base hosting capacity of all networks was determined. Fig. 15a shows the distribution of the specific reasons that limit the hosting capacity of the networks. As can be seen, in one third of the networks the full roof top potential can be used (that is, the maximum hosting capacity is reached). In another third the roof top potential can be used if the transformer is exchanged to a transformer of larger size (maximum 630 kVA). In ∼10% of the networks overloaded lines restrict the hosting capacity and in ∼5% of the networks a transformer larger than 630 kVA would be necessary in order to use the full roof potential. The hosting capacity of the remaining 25% of the grids is bound by voltage limit. For these 85 networks the application of a rONT could help to overcome this limit. Fig. 15b shows the limiting factor of grid's hosting capacity if a rONT is installed. As can be seen, in more than 75% of the grids the rONT mitigates all voltage problems. However, the rONT only solves all problems in about 34% of these networks, which amounts to about 9% of all networks.



(a) Analysis of the cause that restricts the hosting capacity of all 329 LV grids



(b) For the 85 grids that are restricted by voltage: effect of rONT application

Fig. 15. Results of the analysis of the hosting capacity of 329 Low Voltage grids of the German DSO Bayernwerk

VI. Conclusion

This paper introduced the new open source tool `pandapower` and demonstrated how it is well suited to tackle tasks evaluating future scenarios in distribution systems with a high degree of automation. Minimal examples that show the typical work flow with `pandapower` have been presented and are available as interactive jupyter notebooks on github [4]. A basic algorithm to evaluate the hosting capacity of a distribution system with a Monte-Carlo approach has been presented as an exemplary application of `pandapower`. Furthermore, studies on real networks have been presented. The studies show a large diversity of results for different network regions, which makes it hard to draw general conclusions about the impact of integrating renewable energy sources into the network. Tools such as pandapower are therefore necessary to allow automated investigations with a high degree of automation for a large number of grids [13], [14].

Acknowledgment

This research was in part funded by the German Federal Ministry for Economic Affairs and Energy and the Projekttrger Jülich GmbH (PTJ) within the framework of the project *SmartGridModels* (FKZ 0325616).

References

[1] O. H. Jadhav and S. M. Jadav, "Aspects of renewable energy potential in india and future scope," in *2017 International Conference on Nascent Technologies in Engineering (ICNTE)*, Jan 2017, pp. 1–6.
[2] "Zeitreihen zur entwicklung der erneuerbaren energien in deutschland," Federal Ministry for Economic Affairs and Energy, Germany, 2017 (in German).
[3] W. McKinney, "pandas: a foundational python library for data analysis and statistics," *Python for High Performance and Scientific Computing*, pp. 1–9, 2011.
[4] "IEEE recommended practice for industrial and commercial power systems analysis (brown book)," *IEEE Std 399-1997*, pp. 1–488, Aug 1998.
[5] J. J. Grainger and W. D. Stevenson, *Power system analysis*. McGraw-Hill, 1994.
[6] H. Wang, C. E. Murillo-Sanchez, R. D. Zimmerman, and R. J. Thomas, "On computational issues of market-based optimal power flow," *IEEE Transactions on Power Systems*, vol. 22, no. 3, pp. 1185–1193, 2007.
[7] "IEC 60909-0:2016: Short-circuit currents in three-phase a.c. systems - part 0: Calculation of currents," International Standard, 2016.
[8] A. Abur and A. Expósito, *Power System State Estimation: Theory and Implementation*, ser. Power Engineering (Willis). CRC Press, 2004.
[9] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.
[10] R. Zimmerman, C. Murillo-Sanchez, and R. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 12–19, Feb 2011.
[11] K. Strunz, N. Hatziargyriou, and C. Andrieu, "Benchmark systems for network integration of renewable and distributed energy resources," *Cigre Task Force C*, vol. 6, pp. 04–02, 2009.
[12] A. Scheidler, L. Thurner, M. Kraiczy, and M. Braun, "Automated grid planning for distribution grids with increasing pv penetration," in *6th International Workshop on Integration of Solar Power into Power Systems*, Vienna, Austria, November 2016.
[13] L. Thurner, A. Scheidler, A. Probst, and M. Braun, "Heuristic optimization for network restoration and expansion in compliance with the single contingency policy," *IET Generation, Transmission & Distribution*, July 2017. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/iet-gtd.2017.0729
[14] A. Scheidler, L. Thurner, and M. Braun, "Automated distribution system planning for large-scale network integration studies," *IET Renewable Power Generation (submitted)*, 2017.

[4]https://github.com/lthurner/pandapower