OpenTUMFlex

Release 1.0

Babu Kumaran Nalini, Zhengjie You, Michel Zade

Jun 15, 2021

CONTENTS

Authors Babu Kumaran Nalini, Zhengjie You, Michel Zadé

Organization Chair of Energy Economy and Application Technology, Technical University of Munich

Version 1.0

Date 14.06.2021

DOI 10.5281/zenodo.4251512

CHAPTER

ONE

GENERAL INFORMATION

Authors Babu Kumaran Nalini, Zhengjie You, Michel Zadé

Organization Chair of Energy Economy and Application Technology, Technical University of Munich

Version 1.0

Date 01.06.2021

DOI 10.5281/zenodo.4251512

1.1 Description

An open-source python-based flexibility model to quantify and price the flexibility of household devices.

- uses mixed-integer linear programming (MILP) to obtain cost-optimal operational schedules for household devices.
- calculates the flexibility potential and flexibility prices based on price, weather, generation and load forecasts of household devices.
- supports the following devices: PV, battery storage systems (BSS), electric vehicles (EV), heat pumps (HP), combined heat and power (CHP) units.
- outputs flexibility offers for each household device in formats that can be used in flexibility markets (e.g. comax by Tennet or ALF by FfE e.V.)

1.2 Changes

14.06.2021 - documentation release

1.3 Dependencies

- Python IDE: Spyder or Pycharm
- Virtual environment: Please check environment_v1.0.yml
- Solver suggestions: glpk , cplex or any MILP solvers supported by Pyomo

CHAPTER

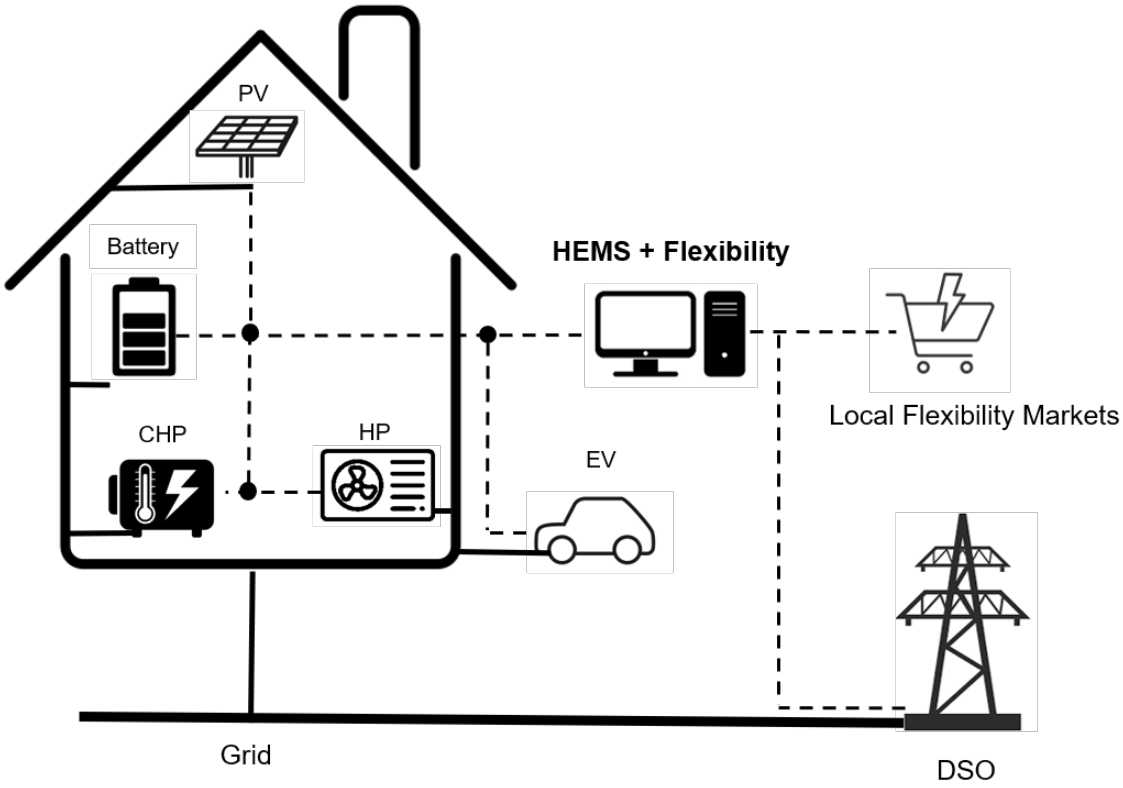
TWO

MODELLING PROSUMER FLEXIBILITY

This chapter details about the concept of flexibility and how prosumer can get involved with Local Flexibility Markets (LEMs) by quantifying and pricing their device flexibility using OpenTUMFlex. Following are the important terminologies used with understanding the concept of prosumer flexibility.

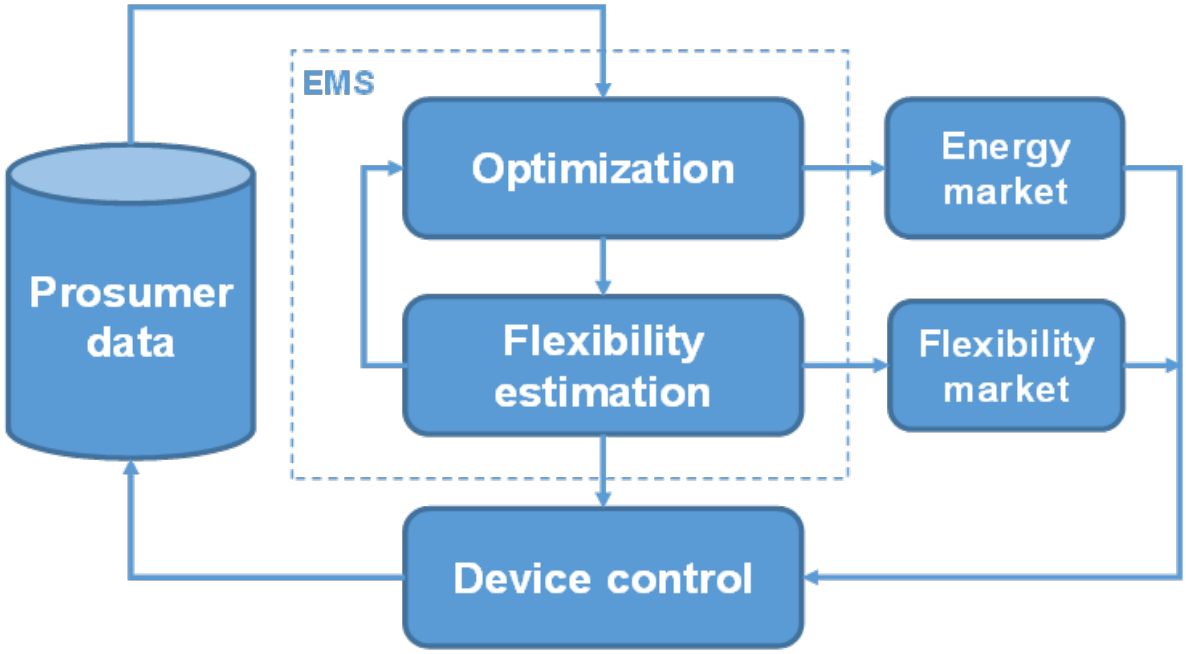
2.1 Background

The increasing share of renewable energy requires alternative methods to provide power system ancillary services to ensure a stable operation of the electricity grids. Recent research has inclined their interests towards the aggregation of small-scale system flexibility potentials to accommodate grid variations. The advancements towards local flexibility markets allow prosumers participation in solving grid congestion problems. In order to allow prosumers to interact with the LFMs and submit their bids, a flexibility model is required.



2.2 Design

This research proposes an open-source flexibility estimation model that quantifies all possible flexibilities from the available prosumer devices and prices them. The ideology is utilize an energy management system (EMS) which usually evaluate optimal operation plan such load scheduling and add additional feature to it to compute device flexibilies. This allows the EMS to interact with energy markets for its day to day operation and as well allows prosumer to offer flexibility services to the LFMs.



2.3 Flexibility

Flexibility is computed as a measure of quantifying the amount of deviation from the optimal operation of a device. For example, the following figure shows an abstract representation of a device flexibility. Considering the red line to be the optimal operation of a device then the blue region defines the possible deviation the device can undergo during any specific time without deviating from its device properties.

For example, lets suppose a PV system is generating 3kW of power between 12:00 and 12:15 in an optimal operating condition, then the maximum negative flexibility of PV system -3kW by curtailing the feed-in power.

2.4 Terminologies

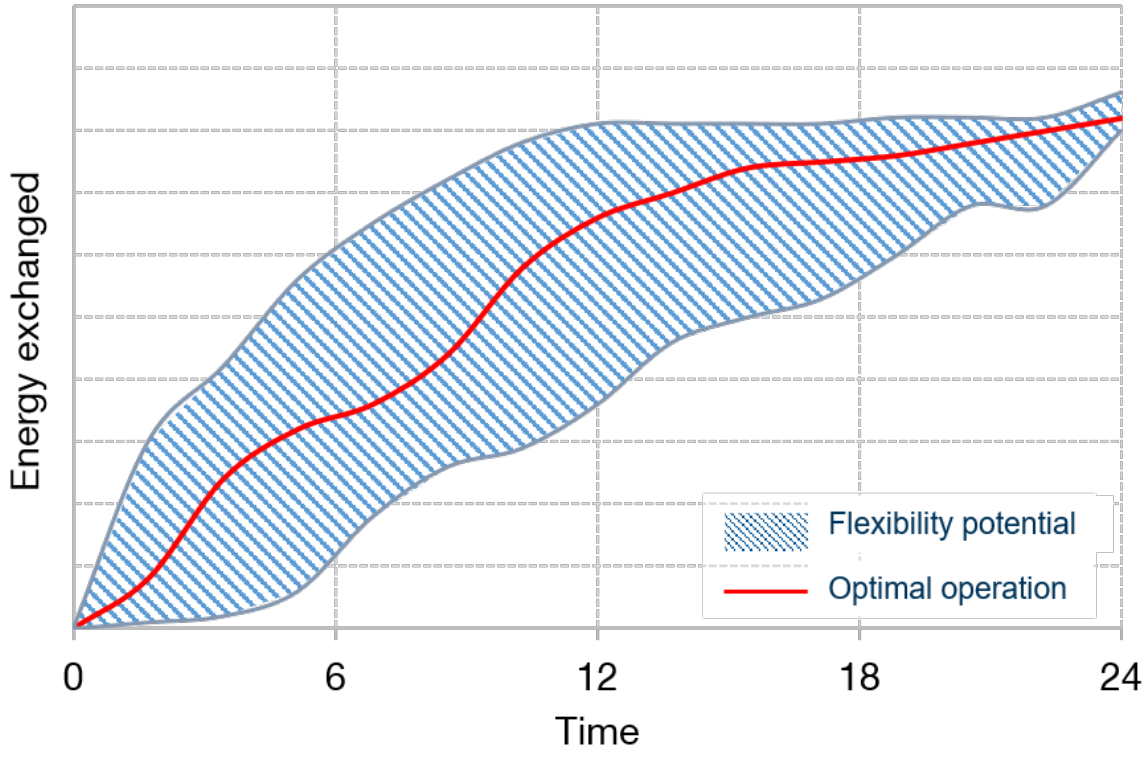
Prosumer

An electricity customer who can actively influence their generation or demand pattern

Flexibility

Flexibility comprises of the measures that influence the generation and/or consumption of the electricity in response to an external signal (price signal).

Positive flexibility



Flexibility measures that results in net addition of power to the grid.

Negative flexibility

Flexibility measures that results in net removal of power from the grid or curtailment of scheduled grid feed-in.

Local Energy Markets

Local energy markets are local exchanges that try to balance generation and demand close to real time.

2.5 Abbreviations

BSS Battery Storage System

CHP Combined Heat and Power system

DSO Distribution System Operator

EV Electric Vehicle

HEMS Home Energy Management Systems

HP Heatpump

HS Heat Storage

LEM Local Energy Market

LFM Local Flexibility Market

MILP Mixed Integer Linear Programming

PV Phovoltaic system

CHAPTER

THREE

GETTING STARTED WITH OPENTUMFLEX

OpenTUMFlex is an open-source flexibility estimation model that quantifies all possible flexibilities from the available prosumer devices and prices them.

To test and execute your OpenTUMFlex model, this chapter explains about the Python installation procedure and enabling virtual environments.

3.1 Installation

OpenTUMFlex is an open-source model developed using Python programming language and uses additional packages such as Pyomo, Matplotlib, etc., to enable its functioning. This section explains the step-by-step procedure to install Python IDE and run your first OpenTUMFlex example.

Python IDE To use Python programming language you require an integrated development environment (IDE). We recommend you to use one of the following IDEs to run OpenTUMFlex.

- Pycharm 2020 or later: Community or Professional edition
- Spyder 4 or later: Miniconda or Anaconda version

Solver To enable Mixed Integer Linear Programming (MILP) optimization, OpenTUMFlex can use GLPK, CPLEX, Gurobi or any other MILP solvers which are supported by Pyomo. By default, the OpenTUMFlex optimization runs using GLPK.

- ***Installing GLPK***
    1. Go to GNU Linear Programming Kit (GLPK) website.
    2. Copy the pip command and run it using Anaconda or command prompt.
    3. In case you have any trouble in your installation, the following video can help you.
- ***Installing Gurobi (for students and researchers)***
    1. Go to Gurobi website and create an account with your university email.
    2. When the account has been activated, log in and download the newest Gurobi solver.
    3. Go to Academia->Academic Program and Licenses
    4. Follow the installation instructions under "Individual Academic Licenses" to activate your copy of Gurobi

Clone repository You may use anyone of the following methods to clone the OpenTUMFlex repository to your local drive.

- Directly download to a local directory of your choice.
- Use version control tools such as GitHub Desktop, Sourcetree, GitKraken, etc.

- Use pure Git link `git clone https://github.com/tum-ewk/OpenTUMFlex.git`

3.2 Virtual environment

To install all the required packages, the easiest way is to use a virtual environment.

Shift to the local directory where OpenTUMFlex is to installed `cd "Local directory path"`

Pycharm - Python 3.9.

1. Create the virtual environment: `open "Anaconda Prompt" -> conda env create -f environment_v1.0.yml`
2. Activate the virtual environment:
   - In the command prompt type conda activate OpenTUMFlex
   - Go to Project interpreter `File->Settings->Project->Python Interpreter`
   - Now add an environment `Add->Conda Environment->Existing environment->Select folder->OK`

Spyder - Python 3.7

1. Create the virtual environment: `open "Anaconda Prompt" -> conda env create -f environment_v1.0_py37.yml`
2. Activate the virtual environment:
   - In the command prompt type conda activate OpenTUMFlex_py37
   - A new Spyder IDE application will be installed and can be found in the start menu.

Note: Spyder while writing this documentation doesn't support Python 3.9 which is originally used in our enviroment file, we have added an additional enviroment file to support Python 3.7 for Spyder IDE.

3.3 Test your installation

Run the example.py file to test if the OpenTUMFlex model is correctly installed. If the installation was succesful, you will see the following results:

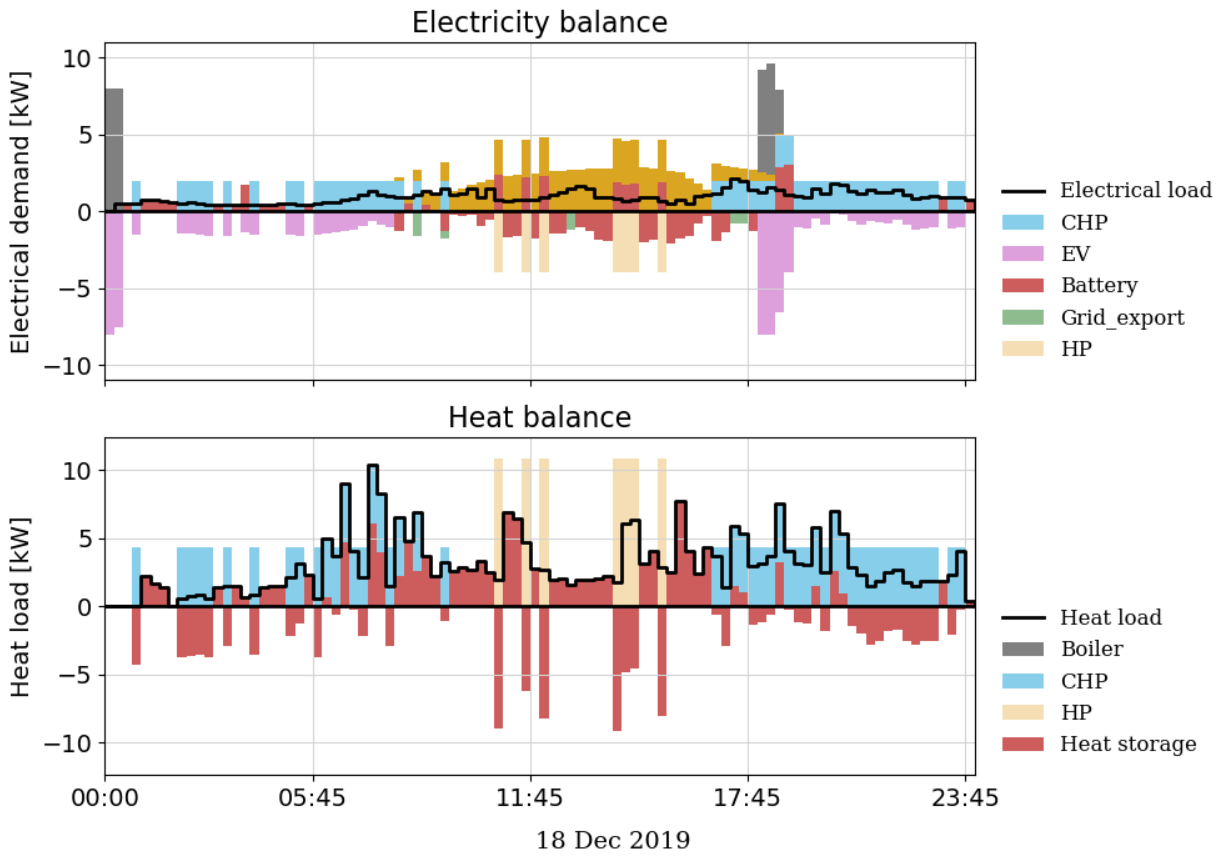


Figure 1: Electrical and heat load balance

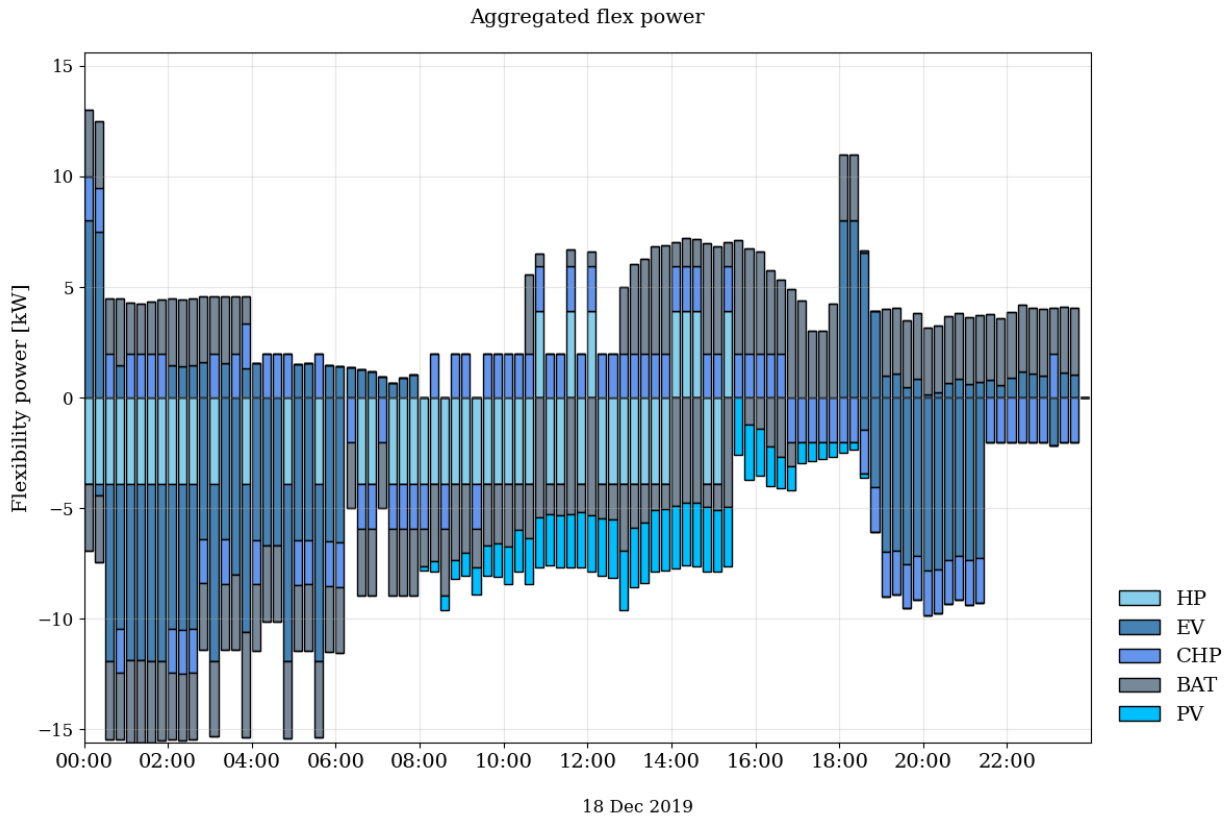


Figure 2: Aggregated flexibility for all the devices

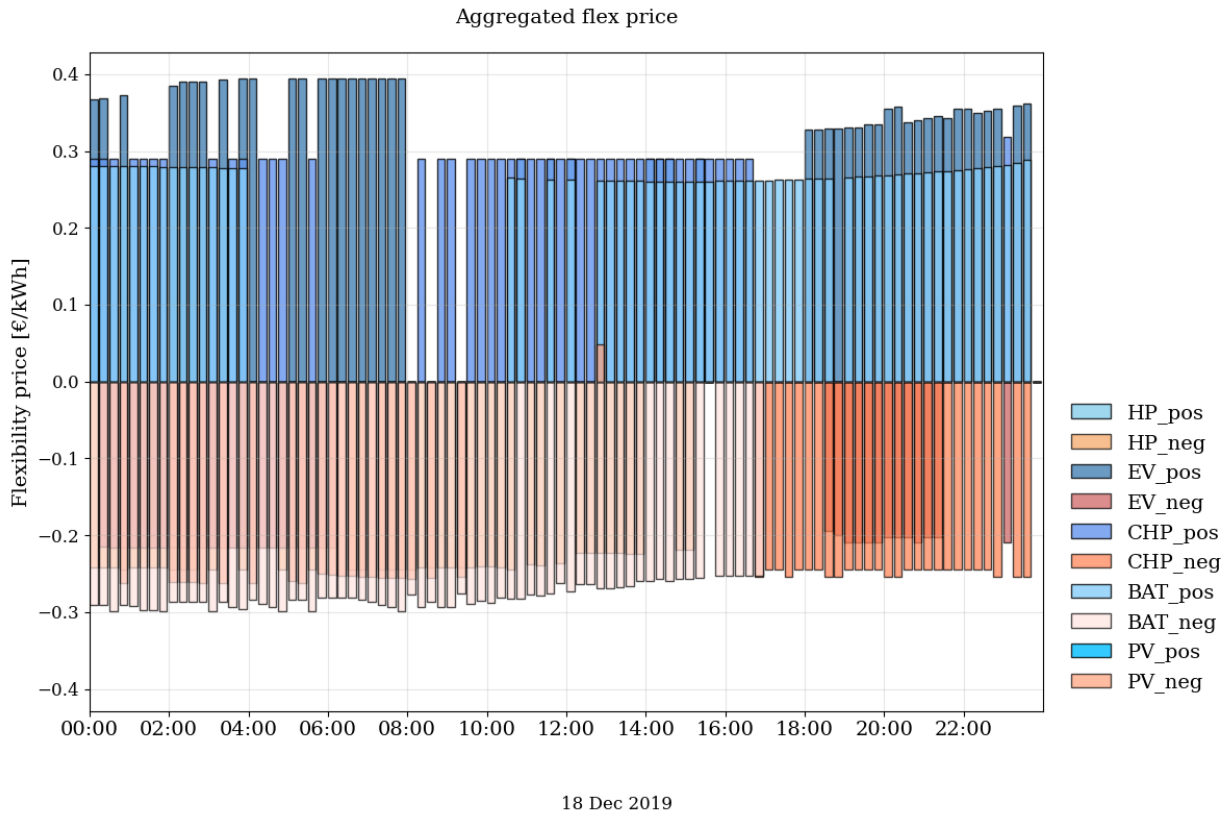


Figure 3: Aggregated flexibility price

CHAPTER

FOUR

CONFIGURING SCENARIOS

A scenario-based approach is incorporated in the OpenTUMFlex design. The term scenarios in OpenTUMflex means cases or examples. A scenario refers to the device configurations installed in a household. For example, a scenario can refer to a household with PV, BSS and EV while a different household with just PV and HP is considered as another scenario.

You can create scenarios of your own or modify/use the existing scenarios. The file inside `OpenTUMflex->Scenario->scenario.py` serves as the main file to decide which scenarios are available.

4.1 Default scenarios

Once you have successfully installed and test run your OpenTUMFle, then `scenario_apartment` is executed by default. You can choose to change your scenario by replacing the desired keyword in line 12 in `example.py`. To give an idea, we have already created 8 sample scenarios inside scenario.py.

Following scenarios are created based on possible combination of devices installed at any prosumers premises.

Scenario	Devices
`scenario_hp`	HP and boiler
`scenario_pv`	PV and boiler
`scenario_bat`	BSS and boiler
`scenario_ev`	EV and boiler
`scenario_simple_house`	PV, BSS, heat storage and boiler
`scenario_residential_house`	PV, BSS, heat storage and boiler
`scenario_mini_apartment`	PV, BSS, heat storage, boiler, HP and EV
`scenario_apartment`	PV, BSS, heat storage, boiler, CHP, HP and EV

Scenario from file

OpenTUMFlex allows to you to directly create a scenario from your input file based on the devices included. To allow this use `scenario_fromfile` as keyword.

4.2 Creating scenarios

OpenTUMFlex allows two ways to create a new scenario.

Using .xlsx/.csv input file

- Inside the input folder, we have added an example .xlsx file to understand the input format.
- Modify the excel cells based on your scenario.
- Add zeros to parameters in case the device is not available in your scenario.
- Point the input directory variable `input_file=Path` to the input .xlsx/.csv
- Use `opentumflex.scenario_fromfile` as the argument while calling `run_scenario` function.
- To speed up the total time, we have the option to create/use CSV. See x.x.
- Remember this method uses both the input file sheets namely properties and time_series.
- This method is useful if you wish to run OpenTUMflex for a single case study.

Using functions

- Create a new function inside `OpenTUMflex->Scenario->scenario.py`.
- As an example, let's say the function name is `scenario_happyhouse`.
- Use `opentumflex.scenario_happyhouse` as the argument while calling the run_scenario function.
- Remember this method also uses the input file but only the time_series sheet.
- Make changes to the .xlsx/.csv input file to amend the input time series data.
- This method is useful during analysis, e.g. modify device parameters iteratively.

4.3 File formats

Input file formats

- OpenTUMflex accepts two different file formats (.xlsx/.csv) as input.
- Example input file is available inside the input folder.
- CSV files are generally preferred as they reduce the overall optimization time.
- XLSX file can be also used for better readability and practice.
- The model has an inbuilt option to convert any .xlsx file to .csv files.

> **Warning:** Do not change the structure of the .csv/.xlsx files. OpenTUMFlex can read the inputs only using the defined sheet settings. Sample .csv/.xlsx formats are available in the input folder.

Converting .xlsx files to .csv

To convert your .xlsx file to .csv format, run `example.py` once with .xlsx file as your input and make sure to assign `convert_input_tocsv= True`. After the execution a CSV file is automatically generated and saved into the input folder. For the subsequent runs you may use the CSV format and disable `convert_input_tocsv = False`.

Output file format

- OpenTUMFlex can generate a variety of plots such as optimal operation, SOC, flexibility, etc.

- These images can be stored as JPG files in the output folder.
- X.X discusses the arguments to save the output images.
- Flexibility offers of each device in suitable formats can be generated as .csv files
- Formatted Output files can be used in FlexMarket trading - e.g. ALF/ COMAX platforms.
- X.X discusses the arguments to save the output files in defined format.

4.4 I/O structure

Input file structure

The input file consists of two parts which are namely the device properties and the forecasted time series values. In XLSX format, it is implemented as two sheets namely `properties` and `time_series`. In the CSV format, the time series data is concatenated with the device properties and are seperated by `;`.

The following parameters are used for defining device properties.

Device	Parameters
PV	Peak power
BSS	Charging/discharging Power (Min & Max), Initial SOC, Capacity, Efficiency
EV	Charging/discharging Power (Min & Max), Initial & End SOC, Capacity, Efficiency
HP	Min & Max Power utilized
CHP	Min & Max Power utilized
HS	Charging/discharging Power (Min & Max), Initial SOC, Capacity, Efficiency, Self discharge

The following parameters are used for defining the time series forecast data.

Forecast data	Units
Temperature	°C
PV power	kW
EV availability data	Binary
Heat load	kW
Electrical load	kW
Electricity import price	€/kWh
Electricity export price	€/kWh
Gas price	€/kWh

Output file structure

The output of the OpenTUMFlex model is the flexibility table that can be used for trading. The flexibility table consists of the scheduled operation along with the possible flexibility service that can be offered in terms of power, energy and price.

The flexibility energy refers to the amount of negative or positive flexibility power that can last over a specific period (timesteps). The flexibility price is the cost that prosumer bids for providing a specific flexibility service. An excerpt from a sample table is provided below and uses COMAX flexibilty platform based output format.

Time	Sch_P	Neg_P	Pos_P	Neg_E	Pos_E	Neg_Pr	Pos_Pr
2019-12-18 01:30	0.6615	-3.661	2.338	-0.915	1.169	-0.298	0.279
2019-12-18 01:45	0.5808	-3.580	2.419	-0.895	0.604	-0.298	0.279
2019-12-18 02:00	0.0000	-3.000	3.000	-3.000	0.750	-0.287	0.279
2019-12-18 02:15	0.0000	-3.000	3.000	-3.000	0.750	-0.287	0.279

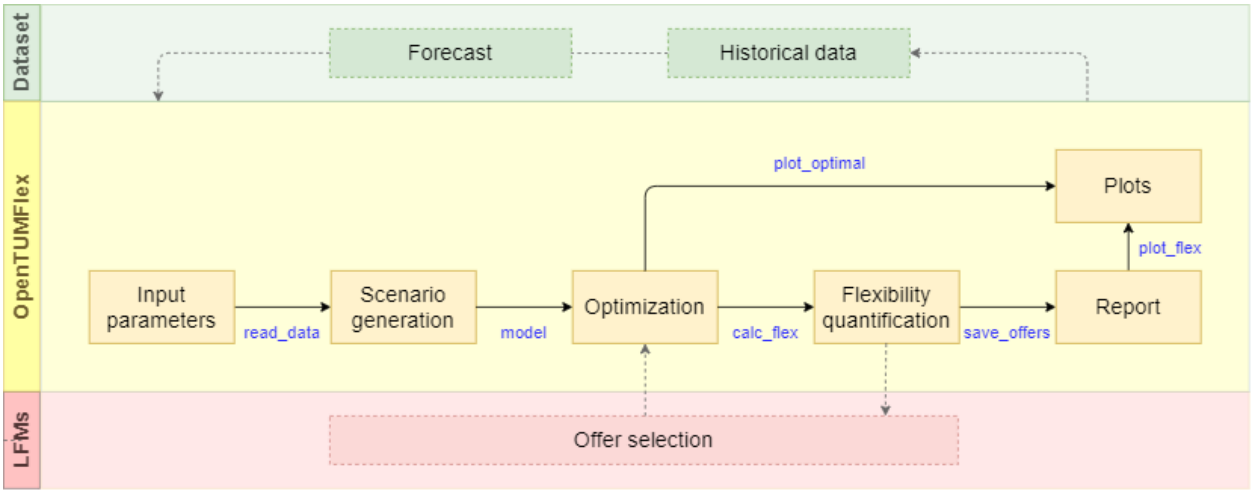
CHAPTER

FIVE

MODULE DESCRIPTION

In this chapter we will disucss the OpenTUMFlex model structure and explanations about how each python files are designed and their respective arguments to control or access different parameters.

5.1 Workflow diagram



5.2 File description

example.py Used as an example main file as well as user interface to control different settings and run a specific scenario.

run_scenario.py Secondary to the main file and executes all the functional modules within opentumflex folder including initialization, optimization, flexibility calculation, result generation and plotting.

scenarios.py Create and/or use diverse scenarios for executing a specific set of devices using OpenTUMFlex.

set_time.py Initialize and configure the time settings and device configuration for the simulation.

init_ems.py Initialize and save ems dictionary object. It also includes the functions for reading forecasting data and device parameters. It contains all the information including devices parameters, optimal operational plan and flexibility for a better overview and user interaction.

devices.py Initialize device parameters with default or customized values or save input data to a local file.

model.py Create and solve Pyomo based MILP models.

report.py Save the MILP optimization results in given path.

flex_pv.py Flexibiltiy quantification and pricing for photovoltic devices.

flex_bat.py Flexibiltiy quantification and pricing for battery storage systems.

flex_ev.py Flexibiltiy quantification and pricing for electric vehicles.

flex_hp.py Flexibiltiy quantification and pricing for heatpumps.

flex_chp.py Flexibiltiy quantification and pricing for combined heat and power devices.

plot_optimal_results.py Plot MILP optimization results as electricity and heat balance.

plot_flex.py Visualize the flexibility quantification and pricing results of OpenTUMFlex for each device individually.

plot_stacked_flex.py Visualize the flexibility quantification result of all the devices in one stacked plot or bar plot. Also plots aggreagated flexibility bid price plots.

plot_flex_reoptimized.py (Yet to be implemented completely) Visualize reoptimized flexibiltiy results once an offer gets selected.

generate_market_offers.py Save flexibility offers in .xlsx/.csv format

5.3 Important arguments

Plotting results Change the arguments to enable/disable each plot:

- `show_opt_res`: plot the optimization results (energy balance and device SoCs)
- `show_flex_res`: plot the flexibility result table of all available devices individually
- `show_aggregated_flex`: plot the cumulative flexibility power and price of all the available devices

Saving results Save the values of the optimization results and devices flexibility results.

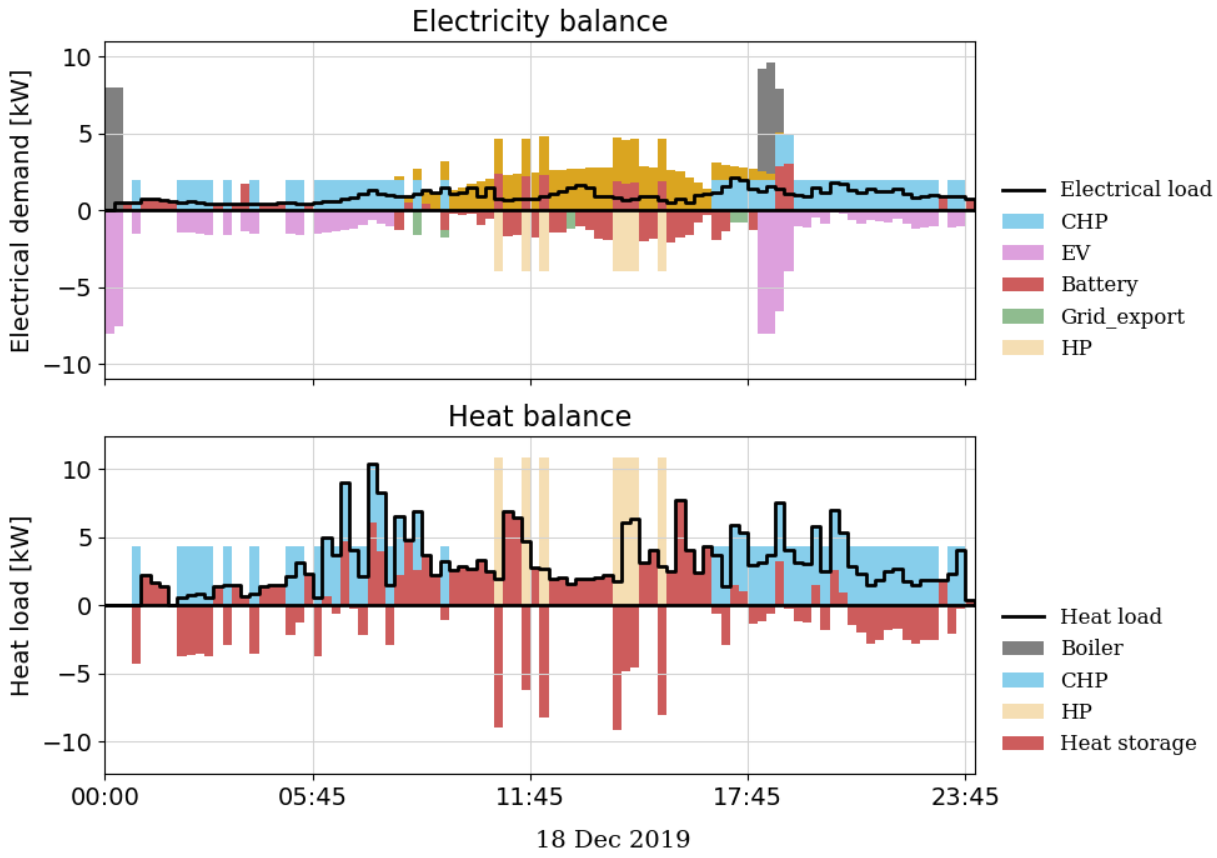
- `save_opt_res`: save the optimization results in a spreedsheet
- `save_opt_res`: save flexibility result of all the devices
- `market`: choose between Comax/ALF LFM formats for the result file

CHAPTER

SIX

ANALYSING RESULTS

6.1 Energy balance

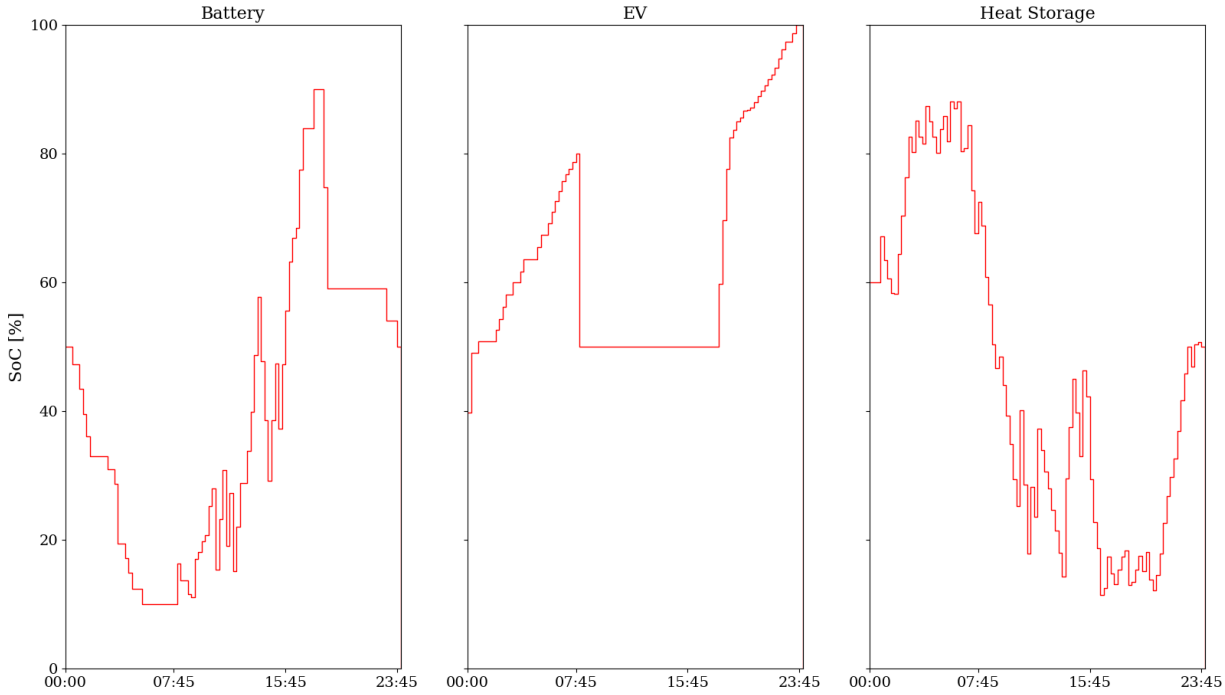


- Upper subplot shows the electricity balance and lower subplot for heat balance
- Bars in positive direction indicate the supply of commodities
- Bars in negative direction mean consumption
- Colors of bars indicate different household devices.
- Thick black line represents the electrical/heat load

6.2 State of charge

State of Charge of the following devices are shown here as combined subplots.

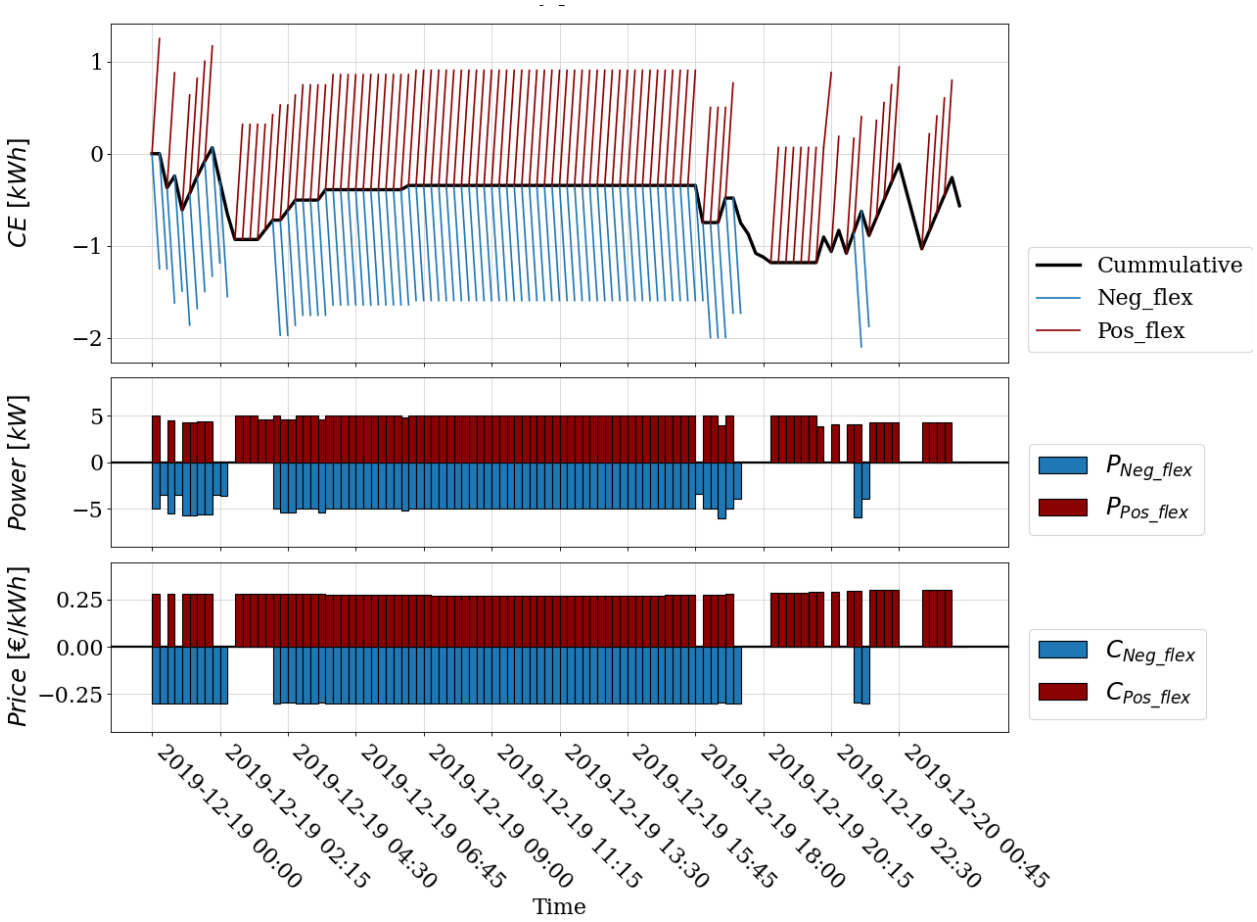
- Battery storage system (BSS)
- Electric vehicle (EV)
- Heat storage system (HSS)



6.3 Device flexibility

To understand device flexibility one specific device is discussed here. The following section discusses about the flexibility of a battery storage system.

Battery Flexibility



- Upper subplot shows the cumulative energy (electricity) consumption/generation[kWh]
- Lower subplot for flexibility power and prices
- Black line represents the cumulative energy consumption/generation [kWh] in the optimal schedule.
- Red lines mean positive flexibility, while blue lines indicate negative flexibility.
- Color of flexibility power and price corresponds with the upper subplot

CHAPTER

SEVEN

TOOLBOX AND PUBLICATIONS

In this chapter we take a look into the toolboxes that use the opentumflex package to perform various studies and the relevant publications.

7.1 Analysis toobox

EV Case Study (GitHub) A case study for the quantification of flexibility of electric vehicles. The case study can be used to analyze the impact of different controller/user strategies, pricing tariffs, and charging power levels on the flexibility of electric vehicles.

7.2 Publications

2020

Zade, M., You, Z., Kumaran Nalini, B., Tzscheutschler, P., & Wagner, U. (2020). Quantifying the Flexibility of Electric Vehicles in Germany and California—A Case Study. Energies, 13(21), 5617, doi: 10.3390/en13215617

(Poster) B. Kumaran Nalini, Z. You, M. Zade, P. Tzscheutschler and U. Wagner, Open-source Flexibility Estimation Python Model for Prosumer Interaction with LFMs. MSE Colloquium 2020, Technical University of Munich, July 2020.

2019

Z. You, B. K. Nalini, M. Zade, P. Tzscheutschler and U. Wagner, Flexibility quantification and pricing of household heat pump and combined heat and power unit, 2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), Bucharest, Romania, 2019, pp. 1-5, doi: 10.1109/ISGTEurope.2019.8905594

B. K. Nalini, M. Eldakadosi, Z. You, M. Zade, P. Tzscheutschler and U. Wagner, Towards Prosumer Flexibility Markets: A Photovoltaic and Battery Storage Model, 2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), Bucharest, Romania, 2019, pp. 1-5, doi: 10.1109/ISGTEurope.2019.8905622

2018

M. Zade, Y. Incedag, W. El-Baz, P. Tzscheutschler and U. Wagner, Prosumer Integration in Flexibility Markets: A Bid Development and Pricing Model, 2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, 2018, pp. 1-9, doi: 10.1109/EI2.2018.8582022

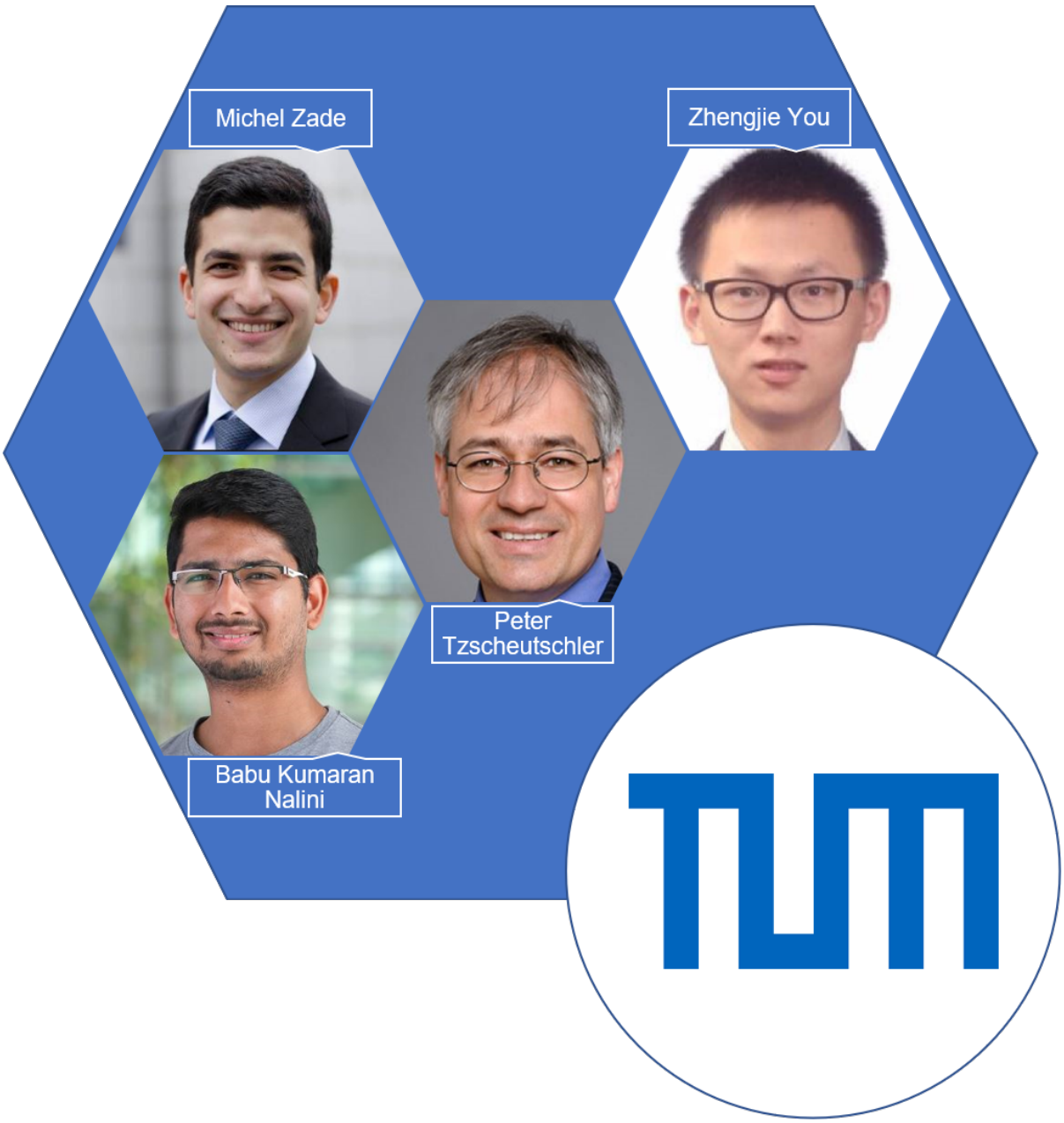
CHAPTER

EIGHT

GETTING INVOLVED

Write to us for a quick chat!

Our team: Babu Kumaran Nalini, Zhengjie You, Michel Zadé, Peter Tzscheutschler



8.1 Project funding

OpenTUMFlex was funded by the C/sells project. C/sells is one of five show cases of the research program Smart Energy Showcases – Digital Agenda for Energy Transition (SINTEG) sponsored by the German Federal Ministry of Economics and Technology.