# A comprehensive framework for Smart Grid simulation

University of Applied Sciences and Arts of Southern Switzerland
SUPSI

Rosato, F., Nespoli, L., Strepparava, D., Medici, V.

SUPSI - Institute for Sustainability applied to the built environment (ISAAC)

## Introduction

A complete framework for the simulation of smart grids needs to provide several features: power flow resolution, graphing, management of inputs and outputs, an interface for plugging in decision algorithms for managed flexibilities, integration of physical and control models, et cetera. The OPTISIM framework was created as a tool capable of satisfying each of these needs, easing the core research task of writing realistic models and novel centralized and distributed control algorithms for demand side management and steering grid-connected apparel.

## Power Flow simulation

The electric simulation engine underlying OPTISIM is OpenDSS [1] a freely available, industry-grade distribution system simulator by EPRI. It is a fast, widely used and powerful simulator, but needs third-party bindings for usage with languages such as Python in order to be interfaced with modern scientific computing packages. Many Python projects exist offering thin wrappers around the core OpenDSS library [2]. We created the Krangpower package [3,4] with the aim to provide several enhancements.Krangpower is built over the OpenDSSdirect.py thin wrapper and offers syntactic sugar such as operator-based insertion of elements in the circuit and retrieval of objects through simple indexing. Furthermore:

- Items that OpenDSS returns as simple lists of floating point numbers, representing real and imaginary parts of flattened arrays of physical quantities, are returned as a numpy array [5] with the correct shape and format.
- Structured items such as the admittance matrix are returned as Pandas DataFrame [6,7] for easier manipulation and export
- Items come, where appropriate, as Quantities (from the pint [8] package) including information on the measurement unit. This enables easy conversions and secures against miscalculations.
- The OpenDSS text interface is checked for errors (normally just returned as strings without raising a Python error).

Another key additional feature is the ability of returning a graph (Networkx package, [9]) representing the underlying grid and featuring the simulation results as node/edge attributes, enabling advanced graphing and analysis.

## Interaction with models

Many of the most interesting smart grid studies involve co-simulation with physical models of batteries, heating, houses and other appliances connected to the grid. We will call them, generically, "agents". OPTISIM offers the possibility to insert software models of these kind of objects directly into the simulation. These models are configured in a dedicated structured json file, whose parameters are fed to the constructor of these objects. During the main simulation cycle, OPTISIM computes the electrical consumption for every model, obtaining active and reactive powers that are then used to configure Krangpower for the following step. Often, these models need time-series data streams as input for computing the final power (such as consumption profiles for uncontrolled loads, irradiance profiles for the models of photovoltaic plants). In vanilla Krangpower, it is possible to use simple delimited files, leveraging the underlying capabilities of OpenDSS. In the OPTISIM framework, a more flexible solution was chosen, involving a time-series database, InfluxDB [10]. The single agents, such as house models, batteries, etc., are aggregated under a single "meter", representing a commercial point of delivery that owns the underlying appliances. Each meter with its agent models runs asynchronously, in a separate process, to achieve high concurrency.

## Interaction with algorithms

The central feature of OPTISIM is the ability to run separately algorithms for governing the behavior of the agents. This is to be distinguished from the built-in, parametrized models of control circuitry that are coded in the agents; we refer to computational, decisional processes that govern the agents in order to obtain an individual or collective goal, such as demand side management [11]. The scope of these algorithms is vast [12,13] and they constitute the central item of research that OPTISIM aims to aid. Examples of algorithms that can be experimented are individual self-consumption optimizers that use a forecast of local production and demand to optimally manage flexibility, or more complicated coordination schemes that involve communication between the agents and the iterative solution of an optimization problem for achieving a common goal under certain sets of rules.

## Overview and message broker

As we have seen, the OPTISIM framework involves several processes running in parallel (the main script with the power flow simulation, the database, the externally interfaced algorithms, the agent models). In Figure 1, an overview of the whole modular architecture is depicted, together with the data flows between the parts.

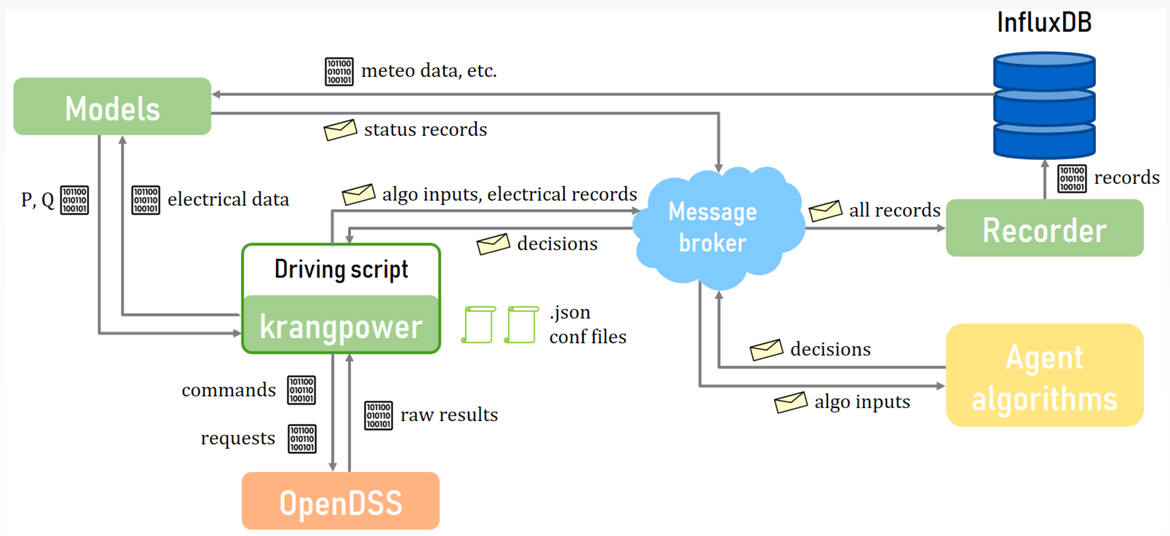


*Figure 1: simulator architecture.*

## Execution time and test runs

The OPTISIM package is written in Python language. The choice was natural, since it is a framework connecting several existing software packages. The wide availability of bindings for all the tools involved and the scripting nature of the project made the choice natural. Nevertheless, the tool is quite optimized. In Table 1 the performance observed with two examples of tests is reported. The two test grids are the following:

- IEEE EU LV (modified): it is a stripped-down version of the IEEE European Low Voltage test network [14]. The agents include thermal building models with heat pumps and boiler, rooftop PV, pure consumption profiles.
- LIC: it is a network modeled from real data from the Lugaggia Innovation Community [15] pilot project in Lugano (CH). The data was supplied by the local DSO, AEM.

| | nr. nodes | nr. meters | nr. agents | wall-clock time per step |
|---|---|---|---|---|
| IEEE EU LV (modified) | 207 | 107 | 447 | 715 ms |
| LIC | 24 | 21 | 120 | 250 ms |
| platform: Intel®Core™i9-7960X @2.80 GHz; 128GB DDR4 @ 3200 MT | | | | |

*Table 1: simulation performance*

In both cases, no algorithm is governing the devices connected to the grid, but the agent physical models are included. Running control algorithms, naturally, can increase the time for taking a simulation step according to their complexity. Examples of results are shown in Figures 2a/b.


*Figure 2a/b: simulation output for a building model and a heat pump model.*

## Conclusion

In this article, the OPTISIM framework was presented as a complete tool for simulating electrical grids interfaced with physical models and management algorithms. The general architecture was investigated and each of the modular components was described. The Python code for the project is under review and will be soon made available to the community as open source.

## Acknowledgements

The authors would like to thank the Swiss Federal Office of Energy (SFOE), Fondo cantonale per le energie rinnovabili (FER) and Ente Regionale per lo Sviluppo del Luganese (ERSL) for their support.

## References

1. Dugan, R.C., McDermott, T.E.: An open source platform for collaborating on smart grid research. In: IEEE Power and Energy Society General Meeting (2011)
2. DSS Extensions: multi-platform OpenDSS extensions. https://dss-extensions.org/Accessed 2020-06-13
3. Rosato, F., Medici, V., Rudel, R.: Krangpower: a smart grid simulation package (2018).http://repository.supsi.ch/10320
4. Krangpower — krangpower 0.2.3 documentation. https://krangpower.readthedocs.io/en/master/
5. Van Der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy array: A structure for efficient numerical computation. Computing in Science and Engineering13(2), 22–30 (2011). 1102.1523
6. pandas development team, T.: Pandas-dev/pandas: Pandas
7. Wes McKinney: Data Structures for Statistical Computing in Python. In: Stefan van der Walt, Jarrod Millman(eds.) Proceedings of the 9th Python in Science Conference, pp. 56–61 (2010)
8. Pint: a Python units library — pint 0.6 documentation. https://pint.readthedocs.io/en/0.6/ Accessed 2020-06-13
9. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using networkx(2008)
10. Naqvi, S.N.Z., Yfantidou, S., Zim anyi, E.: Time series databases and influxdb. Studienarbeit, Universit́e Libre de Bruxelles (2017)
11. Gelazanskas, L., Gamage, K.A.A.: Demand side management in smart grid: A review and proposals for future direction. Elsevier (2014)
12. Siano, P.: Demand response and smart grids - A survey (2014)
13. Vardakas, J.S., Zorba, N., Verikoukis, C.V.: A Survey on Demand Response Programs in Smart Grids: Pricing Methods and Optimization Algorithms. IEEE Communications Surveys and Tutorials17(1), 152–178 (2015)
14. Schneider, K.P., Mather, B.A., Pal, B.C., Ten, C.W., Shirek, G.J., Zhu, H., Fuller, J.C., Pereira, J.L.R., Ochoa,L.F., De Araujo, L.R., Dugan, R.C., Matthias, S., Paudyal, S., McDermott, T.E., Kersting, W.: Analytic Considerations and Design Basis for the IEEE Distribution Test Feeders. IEEE Transactions on Power Systems33(3), 3181–3188 (2018)
15. LIC – LIC Project. https://lic.energy/ Accessed 2020-06-1