# The Illuminator: An Open Source Energy System Integration Development Kit

Aihui Fu (A.Fu@tudelft.nl), Raghav Saini, Remko Koornneef,
Arjen van der Meer, Peter Palensky, Miloš Cvetković
*Electrical Engineering, Mathematics & Computer Science*
*Delft University of Technology*
The Netherlands

***Abstract*—This paper introduces a flexible and extendable easy-to-use energy system integration development kit: the Illuminator. The Illuminator illustrates challenges arising from the energy transition. Hence, it is suitable in education and for demonstration. It also acts as a sandbox for testing new research concepts, and particularly, distributed energy coordination algorithms in real and non-real time. The Illuminator technology is primarely a modular software platform developed to run on a Raspberry Pi (RasPi) cluster. It is open-source, available at GitHub and developed in Python. The Illuminator comprises models of common energy technologies, such as photovoltaic (PV) panels, wind turbines, batteries, and hydrogen systems. The uniqueness of the Illuminator is in its modularity and flexibility to reconfigure scenarios and cases on the fly, even by non-experts in a plug-and-play fashion. This paper introduces the Illuminator and shows its performance in a simple case study.**

***Index Terms*—Energy system integration, open source, education, energy transition, demonstration**

## I. Introduction

With the binding target of net zero greenhouse gas emissions to achieve a climate-neutral Europe by 2050 [1], the energy transition has become a central societal topic. Throughout the energy transition our energy system grows in complexity, becoming more intertwined and interdependent. Understanding such complex systems calls for a new generation of support tools for modeling, analysis and simulation. Such tools would be able to demystify multi-energy design and operational aspects, and improve system integration by enhancing communication amongst domain experts and stakeholders. In this paper, we introduce a tool called Illuminator, particularly developed for education and demonstration.

The energy system landscape is full of various tools developed over the years for various purposes that all can be deployed to teach energy transition challenges and solutions with various degree of success. The professional and academic engineering tools, such as Digsilent PowerFactory [2], Matpower [3], PyPSA [4] or PandaPower [5] are developed to teach highly skilled engineers (at MSc and PhD level) who will one day use the same tools in their work environment. These tools were not designed with education as their primary focus and therefore, they lack flexibility and modularity to create educational content outside of very specific engineering problems. Similar is the case with energy system optimization models, whose learning curves are quite steep (for example OSeMOSYS which focuses on the long-term integrated assessment and energy planning of multi-carrier energy systems [6]) or tools with limited access to their internal algorithms (for example Energy Hub Design Optimization tool [7]).

On the other end of the spectrum are the tools developed with the primary purpose of education. Reference [8] provides a learning tool in the field of the Internet of Energy through web server applications based on a low-cost single-board computer. Reference [9] and [10] provides educational setup methods for power electronics. Even though these tools can easily be used in a classroom they are not sufficiently modular and flexible to cover a variety of challenges and solutions that energy transition brings. The sheer complexity of the energy transition makes such tools narrowly-focused.

Finally, there are tools whose scope and flexibility are better suited for the diversity of questions at hand. A Decentralized Energy Management Simulation Toolkit (DEMKit) [11] provides a sandbox to test different optimization algorithms for energy management while allowing hardware-in-the-loop experiments. The laboratory test beds, such as [12] and [13] can be reconfigured in different connectivity architectures, emulating sometimes entire distribution grids. Such modular and flexible approaches are an inspiration for the Illuminator.

In this paper, we introduce a do-it-yourself kit for education and demonstration of the energy transition challenges and solutions. The kit, which we name *the Illuminator*, is flexible and modular, allowing for a variety of cases to be explored. The kit is available on GitHub under the LGPL license [14]. The Illuminator comes with models, case studies and scenarios to represent a variety of possible contexts relevant to the energy transition.

The architecture of the Illuminator is presented in Section II. The libraries of the Illuminator are introduced in Section III. The Illuminator usage is explained in Section IV. An example study showing the Illuminator performance is given in Section V. Finally, we present the conclusions in Section VI.

## II. Architecture of the Illuminator

### A. Design requirements

To address broad education and demonstration needs, the most important design requirements of the Illuminator are:

The project is supported by Stichting 3E and the PowerWeb Institute. Contact: illuminator@tudelft.nl

1) represent common energy technologies and systems
2) table-top design
3) extendability
4) plug-and-play capability
5) easy on-the-fly reconfiguration of the examples
6) replicability

To achieve these design requirements, the Illuminator architecture is conceived based on Rasberry Pi (RasPi) technology and Mosaik co-simulation framework. Raspberry Pis address design requirements 2, 3 and 4. Mosaik addresses the design requirements 3, 4 and 5, while design requirement 6 is ensured by the open source nature of the project. In Section III we describe how design requirement 1 is addressed. To better address requirements 1, 3, 5 and 6, note that the Illuminator is primarily a simulation tool which relies on information exchange, in contrast to power hardware test-beds.

### B. Hardware considerations

The hardware inter-connectivity of the Illuminator is shown in Fig. 1. In our design, we have selected to interconnect RasPis with Ethernet cables via a switch. This approach was selected to ensure scaling to dozens of RasPis as it handles communication and power supply via the same cable, reducing the cabling required by the system while avoiding WiFi connectivity issues. For smaller setups, with only a few RasPis, WiFi and battery supported alternatives can be viable. Two different variants of the Illuminator module are shown in Fig. 2, one hosting 8 RasPis, another hosting 24 RasPis. These modules can be connected together to scale up the setup capacity.

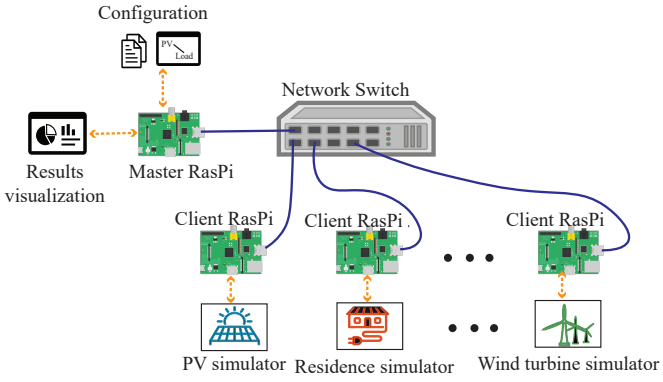


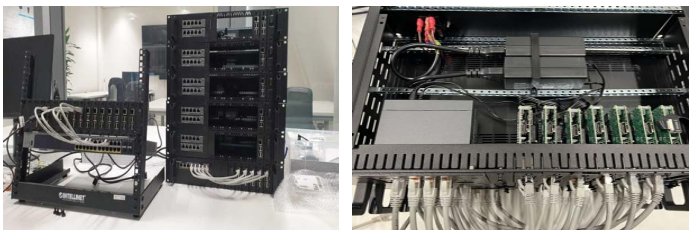

Fig. 1. The structure of the Illuminator

Each RasPi is supplied with its own SD card allowing simultaneous booting of the entire setup. Even though RasPis can be configured to use shared memory space, i.e. one SD card, we have abandoned this approach as it requires sequential access to the memory space, leading to unnecessary long boot times for setups with dozens of RasPis.

In this architecture, one RasPi is defined as the leader RasPi while the rest are the follower RasPis. The leader-follower architecture has been chosen for several reasons. First, the co-simulation framework Mosaik follows the same architecture in order to keep track of the global time propagation of the entire simulation channeling all the messages between RasPis through one single central point. Second, the auxiliary functionalities such as configuration, initialization, visualization, logging, etc. can be more easily streamlined from and by the leader.

Beside their table-top qualities, RasPis provide another benefit, namely, an abundance of peripherals that can be used to provide a more tangible feel to education and demonstration activities. They also have a vivid community with strong support.

Although RasPis have been taken as the target deployment platform, the subsequent simulation platform described in this paper is Python-based and can therefore be run on any operating system that supports Python, albeit without the use of exotic peripherals.



a) Iluminator hardware box b) Devices inside the box

Fig. 2. Two variants of the Illuminator module

### C. Simulation platform

The Illuminator software platform is largely based on Mosaik. Mosaik [15] is a simulation tool to link simulators and/or models together and to perform co-simulation. It synchronizes the simulation process against a global time clock and manages the data exchange between the simulators via a socket connection. These two features make it a suitable orchestrator for distributed simulations. In other words, Mosaik gives an opportunity to:

- run simulations across a computing cluster (such as several stringed RasPis),
- to independently develop models in any programming language
- to use models already encapsulated within various simulation tools.

In other words, Mosaik allows to blure the line between a model and a simulator, integrating both into a unified simulation environment. Mosaik is written in Python, and hence, we choose Python as the development language for creating the core libraries of the Illuminator.

The process of using and setting up the Illuminator is shown in Fig. 3.

Configuring simulation runs, first requires specifying which models will run and in which configuration (see next section for details). Second, it must be specified which model will be deployed on which RasPi. Note that more than one model can run on a single RasPi since they are connected via socket

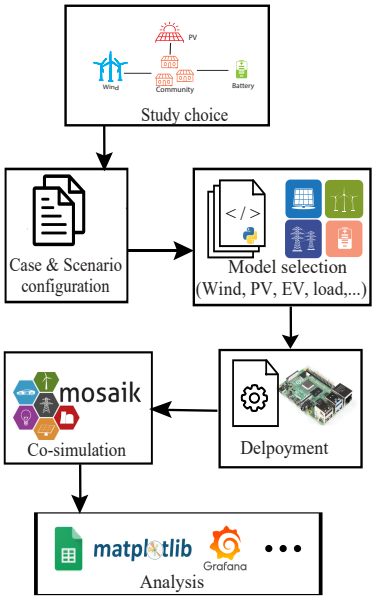



Fig. 3. The simulation process of the Illuminator

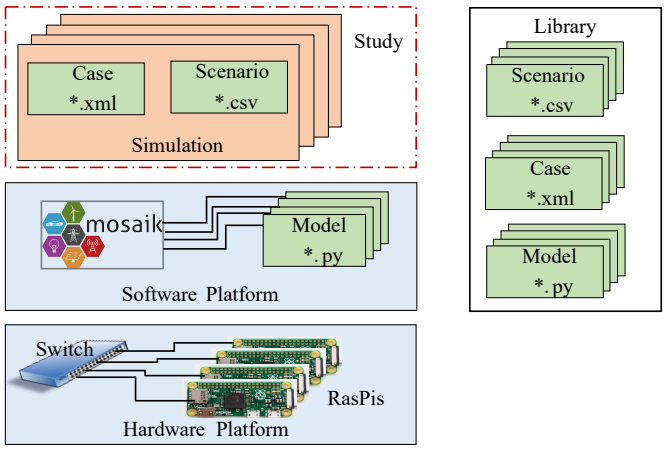



Fig. 4. The Illuminator concept

connections. The deployment of the models across RasPi is achieved through the shell script (sh-file) via the leader RasPi, leaving room for creation of an automated software deployment architecture in the future.

Since the entire message exchange must pass through the leader RasPi due to the simulation synchronization needs, it is easy to setup monitoring and logging functionalities from this node. User can select the model states to be monitored. The states and the simulation results are stored as *.csv files, while the Illuminator kit allows plotting data per simulation step or, alternatively, at the end of simulation. The second approach allows faster simulation times. WANDB [14] is used as a dashboard for results visualization during the simulation and to compare results of different simulation runs.

## III. Libraries of the Illuminator

In order to develop education and demonstration around the Illuminator kit, we distribute several libraries for creation of energy transition case studies. The Illuminator comes with:

- a library of models,
- a library of cases, and
- a library of scenarios.

The library of *models* contains the common power system models including PV systems, wind turbines, electric vehicles, households, etc. These models are combined into *cases*. We provide several typical cases relevant for power and energy systems, spanning from a household energy system to the national energy system. Finally, the library of *scenarios* contains the relevant profiles of energy generation and consumption that can be used to investigate the case under different scenarios. The provided models are common and rather simplified academic models, but the open source nature of the Illuminator gives possibilities to extend and enhance them. The models, cases and scenarios are populated with typical parameters and profiles giving a representative outlook of the energy system.

Most of the provided models are static models, comprised of algebraic equations. They are snapshot models which are stringed together into cases. The storage models differ. They contain memory in terms of the State of Charge (SoC).

The Illuminator runs in real and non-real time. The time step of the Illuminator is defined by the user in the whole multiples of one second. Therefore, one second is the finest resolution of the Illuminator kit.

At present, no specific ontology has been proposed for the exchange of Illuminator messages. Typically, information that is transmitted between models includes the input and output power of the model interface, although other internal states may also be shared. This can be useful when, for example, one wishes to devise a control logic of the charging point to react on the battery SoC. The definition of an ontology for Illuminator messages remains an area for future research

## IV. Using the Illuminator

Before we explain how the Illuminator can be used, we first introduce the notion of simulation and study. A combination of a case and a scenario defines one *simulation* run. According to this simulation run, the models are deployed and simulated. A *study* comprises one or more simulation runs. Any phenomena under investigation would be first captured in the notion of study which would then be split into simulation runs specifying desired cases and scenarios.

Although the cases are possible to define by directly modifying *.xml case files, the user also has an option to interact more vividly with the Illuminator by drawing a configuration diagram on a touch screen or a smart board, such as shown in Fig 5. Based on the drawing, the Illuminator will automatically assemble the case file. Assignment of the models to specific RasPis in this maner will be enabled in the near future.

- *Energy transition studies* - As already explained in the main motivation of our work, the Illuminator is intended to capture the relevant challenges and solutions to the

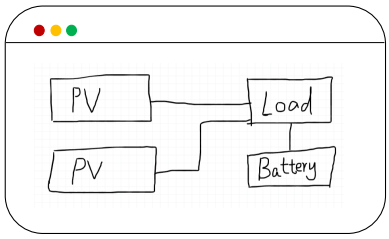



Fig. 5. The Configuration diagram

energy transition, and therefore, the first domain of use of the Illuminator is as a table-top replica of the energy system for education and demonstration purposes. However, its design features allow two other interesting applications.

- *Sandbox for validation and verification* - The Illuminator can be used as a validation and verification platform for newly developed coordination and control algorithms. Python language, together with the distributed architecture and an abundance of models, makes Illuminator perfectly suitable as a sandbox for testing of algorithms, in real and non-real time.
- *Digital twin* - Flexible nature and its ability to be easily reconfigured and deployed, make the Illuminator suitable to spawn a multitude of system variations in operational policies and/or system configuration, in this way informing future design and operational decisions.

## V. An Example Study

One trend to reduce energy bills and greenhouse gas emissions is by forming self-sufficient energy communities. Having sufficient renewable energy generation and storage could potentially make the community self-sufficient. To manage the energy flows, an energy management system (EMS) is needed.

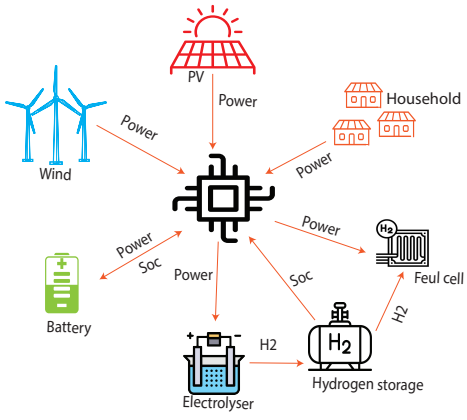



Fig. 6. The schematic of the case

In this example study, we analyze one such system to assess the feasibility of a self-sufficient community. This study also shows the deployment of the Illuminator kit and verifies its performance. The models used are those of a small scale wind turbine, a rooftop PV system, a typical household consumption, and storage assets. Within storage assets, we model a fuel cell, a hydrogen storage tank and an electrolyser as a long-term storage system, and a lithium-ion battery as the short-term storage system. The schematic of this case is shown in Fig. 6. We observe graphically that a controller with the EMS is centrally positioned in this case as an energy routing device. The scenario uses typical June profiles for the consumed power of the households and the generated power from PV and the wind turbine. These profiles are shown in Fig. 7. The simulation time step is set to 15 minutes and one month of system behavior is simulated.

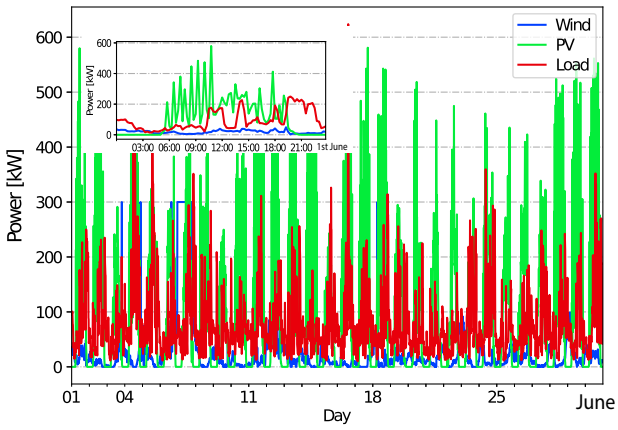



Fig. 7. The Load and generated power from PV and wind in June

The EMS algorithm is deployed on the leader RasPi, while the rest of the models are separately deployed in different follower RasPis. The data flows between the models are shown in Fig. 6. Based on the generated wind and solar power, the consumed power of the household, and the state of charge (Soc) of the storage assets, the EMS decides on the (dis)charging rates for the storage assets. In this case, we use simple EMS logic. If the battery has enough capacity to ensure power balance, battery is used first. If battery does not have enough capacity, then either the electrolyser or the fuel cell are activated.

With the explained control logic, the (dis)charging power of the battery is shown in Fig. 8. The power to the electrolyser and from the fuel cell is shown in Fig. 9. The SoC changes are shown in Fig. 10. From this figure, we see that the battery is mainly used to balance the fluctuation of solar and wind in the daily cycle. The hydrogen storage is mainly used to balance seasonal variations. The battery is sized well to accommodate almost entire daily variability.

The simulation on a RasPi cluster takes 1035 seconds. The same simulation, if deployed on a desktop PC with Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz takes 493 seconds.

This is a simple case with a simple control logic that illustrates how the Illuminator works. The developers can add other models to the library, build their own control logic, link

the models in different configurations and change the time step of the simulation. Each variation could lead to innovative studies and insights.

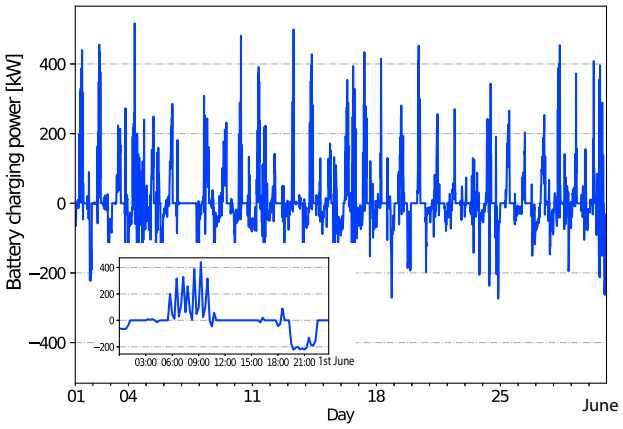



Fig. 8. Battery charging (positive values) and discharging (negative values) power in June

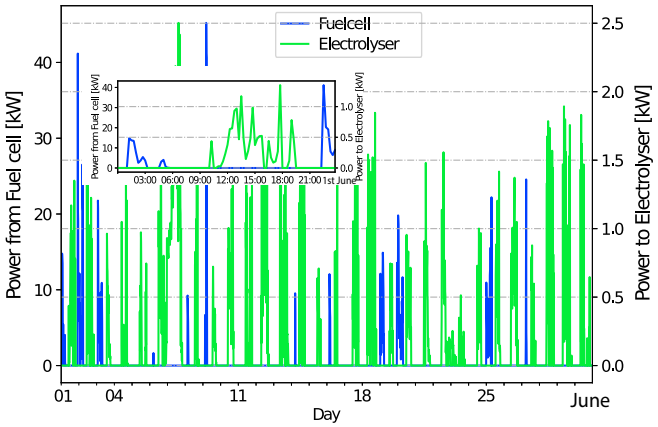



Fig. 9. Electrical power injection from the fuel cell and electrical power taken by the electrolyser in June

## VI. Conclusion

This paper presents a new kit for the education and demonstration of the energy transition challenges and solutions. We explain the details of the Illuminator architecture and its use. As an open-source, replicable, and scalable kit, the Illuminator has multiple applications:

- Educate students and broader communities in energy system operation and energy transition.
- Demonstrate the basic concepts of the system integration of renewable energy and its implications.
- Test innovative algorithm by researchers and developers

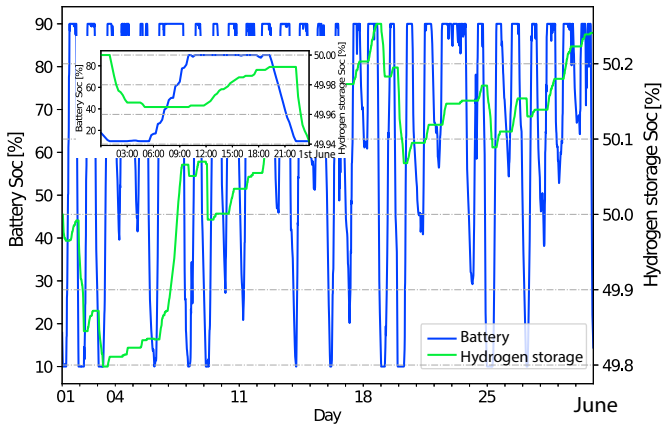



Fig. 10. The SoC of the battery and the hydrogen storage in June

## References


[1] Union, Europen, "O2050 long-term strategy," 2020.

[2] PowerFactory, DIgSILENT, "Digsilent powerfactory 2021 user manual, digsilent gmbh," 2021.

[3] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[4] T. Brown, J. Hörsch, and D. Schlachtberger, "Pypsa: Python for power system analysis," *arXiv preprint arXiv:1707.09913*, 2017.

[5] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, 2018.

[6] T. Niet, A. Shivakumar, F. Gardumi, W. Usher, E. Williams, and M. Howells, "Developing a community of practice around an open source energy modelling tool," *Energy Strategy Reviews*, vol. 35, p. 100650, 2021.

[7] M. Wirtz, P. Remmen, and D. Müller, "Ehdo: A free and open-source webtool for designing and optimizing multi-energy systems based on milp," *Computer Applications in Engineering Education*, vol. 29, no. 5, pp. 983–993, 2021.

[8] C. C. Capasso, D. A. Assante, and O. V. Veneri, "Internet of energy training through remote laboratory demonstrator," *Technologies*, vol. 7, no. 3, 2019.

[9] M. Pajpach, O. Haffner, E. Kučera, and P. Drahoš, "Low-cost education kit for teaching basic skills for industry 4.0 using deep-learning in quality control tasks," *Electronics*, vol. 11, no. 2, p. 230, 2022.

[10] P. van Duijsen, J. Woudstra, and P. van Willigenburg, "Educational setup for power electronics and iot," in *2018 19th international conference on research and education in mechatronics (REM)*. IEEE, 2018, pp. 147–152.

[11] G. Hoogsteen, J. L. Hurink, and G. J. M. Smit, "Demkit: a decentralized energy management simulation and demonstration toolkit," in *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, 2019, pp. 1–5.

[12] V. Salehi, A. Mohamed, A. Mazloomzadeh, and O. A. Mohammed, "Laboratory-based smart power system, part i: Design and system development," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1394–1404, 2012.

[13] J. M. Maza-Ortega, M. Barragán-Villarejo, F. de Paula García-López, J. Jiménez, J. M. Mauricio, L. Alvarado-Barrios, and A. Gómez-Expósito, "A multi-platform lab for teaching and research in active distribution networks," *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4861–4870, 2017.

[14] Illuminator-TUDelft-IEPG, "Illminator development kit," 2022. [Online]. Available: https://github.com/Illuminator-TUDelft-IEPG/Illminator

[15] S. Schütte, S. Scherfke, and M. Tröschel, "Mosaik: A framework for modular simulation of active components in smart grids," in *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*. IEEE, 2011, pp. 55–60.