Energy Conversion and Management 75 (2013) 643–650

Contents lists available at ScienceDirect

Energy Conversion and Management

journal homepage: www.elsevier.com/locate/enconman

# Multi-agent modeling for the simulation of a simple smart microgrid

Enrique Kremers [a,*], Jose Gonzalez de Durana [b], Oscar Barambones [b]

[a] European Institute for Energy Research, Emmy-Noether-Strasse 11, 76131 Karlsruhe, Germany
[b] EUI, Basque Country University, Nieves Cano 12, 01006 Vitoria-Gasteiz, Spain

ARTICLE INFO

*Article history:*
Received 29 April 2013
Accepted 16 July 2013

*Keywords:*
Power grid
Smart grid
Renewable energy systems
Energy saving
Smart meter
Microgrid modelling
Complex systems
Agent-based modelling
Demand side management
Smart grid simulation
Load flow
Load shedding

ABSTRACT

The smart grid is a highly complex system that is being formed from the traditional power grid, adding new and sophisticated communication and control devices. This will enable integrating new elements for distributed power generation and also achieving an increasingly automated operation so for actions of the utilities as for customers. In order to model such systems, a bottom-up method is followed, using only a few basic elements which are structured into two layers: a physical layer for the electrical power transmission and one logical layer for element communication. A simple case study is presented to analyze the possibilities of simulation. It shows a microgrid model with dynamic load management and an integrated approach that can process both electrical and communication flows.



## 1. Introduction

In the last years, energy systems are moving away from a centralized and hierarchical structure, under strict control of the electricity supply companies, toward a new system where distributed actors influence the energy supply. Production is no longer limited to large energy providers, as small decentralized producers in the form of distributed generation (DG) enter the network and are able to inject energy at much lower voltage levels than before.

This paradigm shift involves new challenges for the modeling and simulation of energy systems for which decentralized models are needed. Among the presented in a review in [1], there are, for example, several commercial tools used in power engineering, such as Eurostag or PSS/E. Further, a number of non-proprietary tools exist, such as several toolboxes based on Matlab/Simulink, for example Matpower or PSAT.

Since some years, the term *smart grid* has become widespread in the energy sector. The introduction of smart grids involves a change from manual operations toward an intelligent, ICT based and controlled network. These changes will especially affect the distribution grid [2], and in this way, microgrids.

A number of models have been developed to analyze and understand the behavior of microgrids. Some of them focused on decentralized control strategies, usually using Matlab/Simulink and similar classic tools, are of special interest [3–5]. Particularly relevant is the work on *demand side management*, using *virtual powers plants* [6] and on *multi-agent platforms* [7]. And also worth mentioned as significant the recently extended idea, discussed in some conferences [8] and accepted very well in some countries, of using microgrids as building blocks of the future smart grid [9].

Here, we recall that traditional methods used for analysis of electrical networks are based on static *power flow* calculations [10]. But these methods are not suitable for computing the system response to special events (such as power changes in generators fed by renewable energies, sudden connection and disconnection of loads and sources, or even when the network structure changes after a disaster occurs), because in these cases, the values of the variables have to be updated in a short time, as they are used for the network control or simulation, and therefore, a new "steady" state computation is required each time one of such special event occurs, which is not efficient at all.

Trying to give a solution to this problem the authors designed a decentralized power flow algorithm for this kind of models (see

* Corresponding author.
*E-mail addresses:* kremers@eifer.org (E. Kremers), josemaria.gonzalezde durana@ehu.es (J. Gonzalez de Durana), oscar.barambones@ehu.es (O. Barambones).

0196-8904/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.enconman.2013.07.050

below). This method provides a more flexible and dynamic way than traditional methods and is able to cope with sudden changes and disasters. The method is described in [11] and was successfully validated against the Matlab PSAT Toolbox [12].

### 1.1. Modeling the smart grid

In [11], the complexity for modeling smart grids was identified. A first smart grid model was developed, which represents the system on the physical layer, by integrating a distributed load flow algorithm. The model was tested by running different simulations, letting interact a wind generation unit, a photovoltaic panel, a battery, two loads and a diesel generator. By modeling the individual elements as agents, a modular and flexible approach was used, where the agents can be programmed to have different behaviors such as the charging and discharging times for the battery system, or the integration of variable wind speeds by adding a wind speed simulator module which directly interacts with the turbines. On the logical layer, a first approach was made.

This approach is extended in the current work by adding real-time communications to the simulation, which represent one of the main features of the logical layer. Because of their scarce resources, microgrids need a flexible demand side [13], so introducing communication is essential as it allows performing monitoring, control [14] and demand side management, among others.

As some specific aspects of the author's work in this area were presented in some conferences [11,15,16,11,17], in a rather informal manner, here we intend to offer a more understandable and reproducible description of them, by using some elements from the ODD protocol [18].

## 2. Overview

An agent-based approach was chosen for modeling a simple microgrid, trying to represent a minimalistic smart grid (or *smart-microgrid*). The implementation was done in the multi-paradigm modeling environment AnyLogic.

### 2.1. Purpose

The main aim is to demonstrate the feasibility and convenience of the agent-based methods in that small instance, and further be able to use it as a building block [8] in other more complex *power networks*, and even to grow it to other more capable multiple layered structures, able to deal with *multicarrier energy networks*.

The model is intended to be very flexible and, although its actual size was chosen to be very small, in order to simplify explanations and demonstrations, it may be easily adapted to any other given microgrid structure.

This shall allow for practically any virtual experiment for control and management schemes, some of which, still under development, are required for a sustainable operation of microgrids within the energy system [14].

A very first and simplified model of a load management is proposed, using communicating smart meters. The model focuses on the system view, which rather than representing the load management mechanism in detail, aims to represent the microgrid as a whole in order to observe the effects at that level. The model shows the importance of a logical layer in the smart grid.

### 2.2. Structure – state variables and scales

The model structure is inspired on *power flow analysis* [10]. In this context, an electrical network is represented by a weighted graph $G = (V,E)$ where $V$ is a set of $n$ vertices and $E \in \{V \times V\}$ is the set of edges in the network.

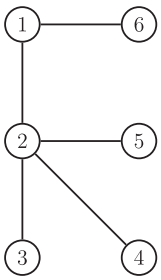


**Fig. 1.** Graph of the model electrical network.

Fig. 1 shows a graph with six vertices, representing the simple electric network of the model. Vertices are also called *buses* by electrical engineers and represent electrical power generators and loads, and edges, also called *lines*, represent electrical power transmission lines. Physically, an electrical network represents a circuit where the electricity is flowing from each node to some other.

The approach the authors proposed in [11] is to split the network node in two parts, in order to obtain a two layer structure, composed of a logical layer and a physical layer. An agent is assigned to each node – so there are also $n$ agents – who acting at the two layers are able to perform their respective tasks: power flow calculations at the physical layer, communications tasks at the logical layer, and also the implied inter-layer actions.

The very simple example showed in Fig. 2, with six agents, has been chosen for instantiate the model. Note the number of vertices is the same, $n = 6$, for both layers, but the edges are different: while at the logical layer, edges represent communication channels, so at the physical layer, they represent electric lines.

These figures also show the vertex numbers and their type in the physical layer: *Slack*, *Control* and *Load*, also named, respectively, *Reference*, *PV* and *PQ* buses, referring to the known real data pair at the two last.

### 2.3. Process overview and scheduling

At the lowest level, like a daemon, the AnyLogic simulation engine is conducting the most basic rhythms for the launched model, using a well defined, scheduled pattern of time and event steps. During each time step, the model clock is advanced, the discrete state of the model remains unchanged, active equations, if any, are being solved numerically, the variables are changed correspondingly, and also, awaited change events are tested for occurrence. During each event step, no model time elapses, the actions of states, transitions, events, ports, etc. corresponding to this event are executed, the discrete state of the model may change, some scheduled events may be deleted, and the new events may be scheduled in the AnyLogic Engine event queue.

At any simulation time, the user can view the event queue of the AnyLogic simulation engine to view what is happening there and even to make some changes to the event processing (see *The Big Book of AnyLogic* [19]). This amazing process is one of the main virtues of the program since it frees the user from the complicated task of treating events, greatly facilitating the implementation of discrete event models.

At the highest level, the Main Class allows the user for instantiate all the other classes corresponding to sub-models, agents creation and replication, main variables definition, model initialization, and also to create the user interface and some complementary variables. This class is associated with the so-called Main Window, through which the user define the main parameters and variables and instantiate the classes. A specially important instance here is the Environment one (whose class can be found in the AnyLogic libraries), where the agents "live", because it is usually used to define the basic rules for agent behavior.