Neural Computing and Applications (2020) 32:5745–5760
https://doi.org/10.1007/s00521-019-04139-3

ORIGINAL ARTICLE

Neural controller for the smoothness of continuous signals: an electrical grid example

Eduardo Matallanas[1] · Manuel Castillo-Cagigal[1] · Estefanía Caamaño-Martín[2] · Álvaro Gutiérrez[1]

Received: 3 August 2018 / Accepted: 9 March 2019 / Published online: 20 March 2019

Abstract
In this paper, the use of artificial neural networks (ANNs) is proposed to manage the local demand of different electric grid elements to smooth their aggregated consumption. The ANNs are based on the load automation of the local electric behavior, following a local strategy but affecting to the global system. In an electrical grid, there is no possibility to share information between the users because anonymity must be warranted. Therefore, a solution to the problem is elaborated with the minimum information possible without the need for communication between the users. A grid environment and behavior of different users is simulated.

Keywords Artificial neural network · Multi-agent system · distributed coordination · Demand-side management

1 Introduction

Working with ANNs in power systems has been long used [12, 20, 22, 29]. However, their application field is centered in tasks related to their pattern recognition and forecasting abilities, in order to handle problems that arise in the grid [24, 28, 31]. In this paper, the use of ANNs is proposed to manage the local demand of different grid elements to smooth their aggregated consumption [5]. However, contrary to standard approaches [30], it is done by focusing on the users instead of using the capabilities of cutting large grid consumptions [9, 10, 38, 39]. In this way, the security of supply is guaranteed for all users and the operation enhancement for the entire grid. This proposal is based on Demand-Side Management (DSM) techniques [1, 34].

DSM is defined as actions that influence the way that consumers use electricity in order to achieve savings or higher efficiency in energy use. DSM is a global term that includes a variety of activities, such as load management, energy efficiency, etc., in a long-term perspective, whereas the short-term perspectives are tackled by Demand Response (DR) programs, which are actions that result in short-term reductions in energy demand [2]. DR can also be defined as incentive payment programs to motivate users to respond to changes in price or availability of electricity by changing its consumption [17]. The different DSM techniques focus on the modification of the load shape. In general, these techniques seek the grid enhancement through an increase in the energy efficiency [17, 18].

Although some real experiments have already been developed [3, 4, 8, 25, 30], there exist some barriers, of different nature, that are difficult and slow their development. One of these barriers is the lack of knowledge and potential of DSM by users [34]. There is also a lack of knowledge about the costs and benefits of DSM because of a lack of methodologies to evaluate it. In [35], the authors analyze DSM experiences in Europe and conclude that limited knowledge has been developed about its overall energy saving capacities. Another barrier is the lack of infrastructure; there is no information coming from the grid

✉ Álvaro Gutiérrez
a.gutierrez@upm.es

Eduardo Matallanas
eduardo.matallanas@upm.es

Manuel Castillo-Cagigal
manuel.castillo@upm.es

Estefanía Caamaño-Martín
estefan@ies-def.upm.es

[1] E.T.S. Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain

[2] Instituto de Energía Solar, Universidad Politécnica de Madrid, 28040 Madrid, Spain

to the users, only the bill of the electricity consumed in a time period. This fact makes difficult the decision of the users in order to actuate over their consumption [23]. The last important barrier in the development of the DSM is the necessity of policies and incentives that motivate its use. The DSM concepts and techniques involve the different grid agents and the necessary development of them implies all the participants of the grid, from big energy producers to small consumers. Thus, it is difficult to develop programs and policies that encompass the entire grid. In addition, the conservative position of the majority of the agents of the electric power sector is slowing down the DSM implementation. Fortunately, this situation is changing as governments are taking part in enhancing the grid operation [6, 16, 36, 37]. Furthermore, new energy policies are emerging motivated by improving the grid efficiency and fighting climate change [13].

Therefore, the interest of developing these techniques consists of enhancing the grid operation and helping with the integration of new technologies. This is the strategy tackled in this paper.

The DSM algorithm proposed in this paper is based on the load automation of the local electric behavior of users. This strategy is applied locally, but its effects impact globally in the system. A neural approach is used to implement it, which consists of controlling the local electric behavior by taking into account the aggregated consumption of the electrical grid and local energy resources.

The use of ANNs allows, by applying their abilities of signal processing and forecasting [21], to vary the behavior of the grid aggregated consumption. Moreover, their adaptive and distributed properties make it perfect to implement a possible solution to this electric paradigm. Specifically, RNNs are used to tackle this problem, due to its dynamic properties and its short-term memory ability inherited in their structure [26, 27, 32]. Both are desirable properties to design a DSM controller that has to adapt to changes in the environment. However, working with RNNs can be difficult and computationally slow, especially for large structures. Therefore, a small RNN structure is sought.

The reminder of the paper is as follows. The environment and the different elements inside it are introduced in Sect. 2. A training algorithm is developed in Sect. 3. The different parameters involved in the training of the neural controller and how it is carried are explained in Sect. 4. Then, the results achieved during the training are presented in Sect. 5. In Sect. 6, a post-evaluation is carried out to test the neural controller achieved at the end of the training. Finally, conclusions are presented in Sect. 7.

2 Environment

In this paper, the environment is defined as a variable number of users, whose consumption is typical for a high electrified house (see Sect. 2.1). The electricity required by those users will be supplied by the grid. The grid is composed by two types of users: (i) non-controllable users, whose consumption cannot be controlled because it is not deferrable or they do not have any DSM system, and (ii) controllable users, who can control their demand in real time through the use of a controller [8, 11].

The aim of the neural DSM controller is to flatten the aggregated consumption of a grid composed by this two type of users. Mathematically, if $P(t)$ is the formulation of the aggregated consumption, then the objective of the neural DSM controller consists of $P(t) \rightarrow C$, where C is a constant. Moreover, $P(t)$ can be divided in the sum of the two users of the environment, such that:

$$P(t) = P^{\mathrm{nc}}(t) + P^{\mathrm{c}}(t) \tag{1}$$

where $P^{\mathrm{nc}}(t)$ is the non-controllable consumption and $P^{\mathrm{c}}(t)$ is the consumption available for the algorithm to be controlled.

Each part of the demand, $P^{\mathrm{nc}}(t)$ and $P^{\mathrm{c}}(t)$, is represented by the sum of the power consumed by each type of user (see Eqs. 2a and 2b).

$$P^{\mathrm{nc}}(t) = \sum_{i=1}^{m} p_i^{\mathrm{nc}}(t) \tag{2a}$$

$$P^{\mathrm{c}}(t) = \sum_{i=1}^{n} p_i^{\mathrm{c}}(t) \tag{2b}$$

where $p_i^{\mathrm{nc}}(t)$ is the demand of the ith non-controllable user and $p_i^{\mathrm{c}}(t)$ is the consumption of the ith controllable user.

The electrical behavior of the users and their aggregated consumption is described in Fig. 1. In the general case, neither of the users can control their demand so that each one of them can consume any power at any time as in Fig. 1a. In this case, there is no algorithm to control the demand of $P^{\mathrm{c}}(t)$ and the aggregated consumption possesses a high variability. Thus, it is necessary for the proposed algorithm to reshape the amount of $P^{\mathrm{c}}(t)$ in order to adapt to $P^{\mathrm{nc}}(t)$, achieving a flattened $P(t)$ consumption as in Fig. 1b. The main challenge consists of how to adapt $P^{\mathrm{c}}(t)$ to $P^{\mathrm{nc}}(t)$ to produce a flattened response. However, the current information coming from the grid and available by users is non-existent, making difficult the elaboration of a control strategy in order to achieve any goal.

In an electrical grid, in which users do not control their consumption, the aggregated consumption behaves periodically as shown in Fig. 2. The aggregated consumption of other countries may differ in the form or in the power

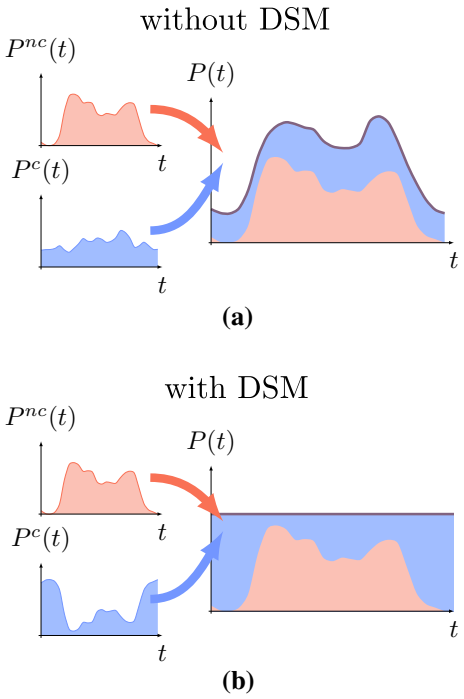

Fig. 1 Graphical representation of $P(t)$ signal as a sum of $P^{nc}(t)$ and $P^{c}(t)$ signals, where: **a** represents the current electric grid where $P^{c}(t)$ is not adapted to $P^{nc}(t)$ and **b** represents the proposed algorithm goal where $P^{c}(t)$ is adapted to $P^{nc}(t)$ in order to smooth $P(t)$

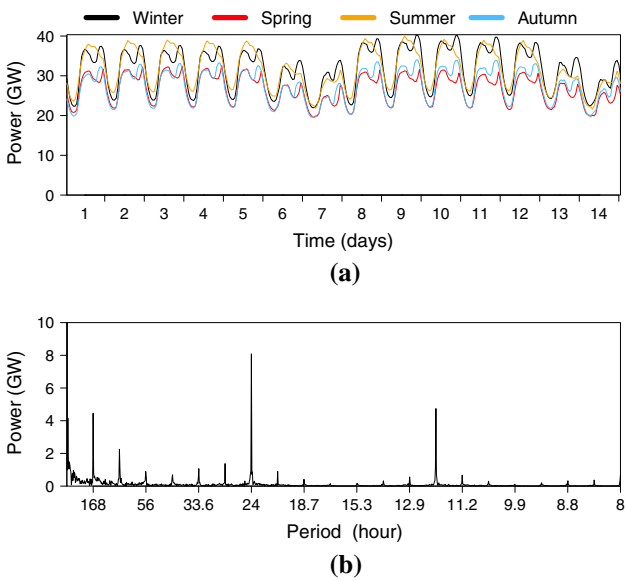

Fig. 2 Aggregated consumption of the Spanish grid during 2017: **a** temporal representation of 2 weeks for different seasons of the year and **b** annual spectrum

consumed, but the periodicity of valleys and peaks is very similar among them because consumption habits among users are very similar. Furthermore, it can be observed that the grid aggregated consumption of the same region is similar over the years, but with very small variations.

Focusing on the grid example of Fig. 2, it shows a periodicity both in time (see Fig. 2a) and in frequency (see Fig. 2b). Figure 2a shows the aggregated consumption of 2 weeks in different seasons of the year. The behavior of weekdays is that they are very similar to each other and to other weekdays from other weeks during the same weather season. The same behavior can be observed for weekends. There are also similarities between winter and summer days and the same for spring and autumn days. This is also corroborated with the annual grid consumption spectrum of Fig. 2b, in which the most significant components are shown. It can also be observed that the strongest component (apart from the continuous one of period 0, corresponding to the average consumption) is the one located at a period of 24 h or 1 day. The next stronger components are located at 12 h period (half day) and 168 h period (1 week), respectively. Thus, it is necessary to reduce these components to smooth the aggregated consumption. However, before describing the algorithm, Sect. 2.1 describes the facilities that compose the artificial grid.

2.1 Facility

An electrical grid is composed by a large variety of consumptions to improve their performance through maximizing the utilization of the installed capacity. Therefore, different consumption profiles are found inside them, such as industrial factories, commercial users or residential ones. The grid is managed by operators in charge of the different parts that comprise it. However, the load consumption is part from the local electric power system of the user. In order to simplify how these power systems are addressed, they will be referred as *facilities*. A facility is owned by a particular consumer whose management depends on him instead of the electric utility. A facility is characterized by different attributes, for example its size varies depending on the needs of the user, from a single family house to large factories or even a micro-grid composed by a neighboring community.

The facilities consider the possibility to actuate in their consumption, and they are divided in two groups: controllable and non-controllable users. The second one represents all those consumptions within the grid for which there is no information about their consumption except the one coming from the aggregated consumption of all of them. Thus, there is no possibility to know any individual information of their load profile. On the other hand, the controllable facilities are perfectly known and their individual consumption is completely specified knowing the instant power they are consuming. The facility is connected to the grid through an electric meter which serves as the exchange point between the grid and the user to measure the electricity consumed by the loads. Inside the facility,

there are three main parts that compose it: (i) generation, (ii) storage and (iii) consumption. The technology used to implement the local generator of the facilities in this paper is Photovoltaic (PV), being one of the most integrated technologies in users.

However, a PV generator can only produce electricity during day hours, producing a bell shape electricity curve related with the radiation of the Sun. Therefore, the facility also incorporates an energy storage system for a better use of the energy generated locally. The incorporation of a storage system gives the facility a way to store the surplus of PV generated for a posteriori use.

The consumption inside a controllable facility can be divided also into two groups (i.e., controllable and non-controllable) depending on the controllable capacity of the load. And for each controllable facility the algorithm will dispose an amount of energy that can be controlled. The rest of the energy will depend on the user preferences.

Figure 3 shows a topology representation of all the elements that integrate the facilities. The electrical behavior of the facility responds to Eq. 3:

$$P_{PV}(t) + P_B(t) + P_G(t) = P_L(t) \tag{3}$$

where $P_L(t)$ is the instantaneous power consumed by loads, $P_{PV}(t)$ is the instantaneous PV power generated, $P_B(t)$ is the instantaneous power exchanged with the local storage system, and $P_G(t)$ is the instantaneous power exchanged with the grid. The sign of each variable depends on the power flow; for example, $P_L(t)$ and $P_{PV}(t)$ are always positive, while $P_B(t)$ is positive when the local storage system supplies power to the loads and negative when it is storing energy, and $P_G(t)$ is positive when the grid supplies power to the loads and negative when the facility exports power to the grid.

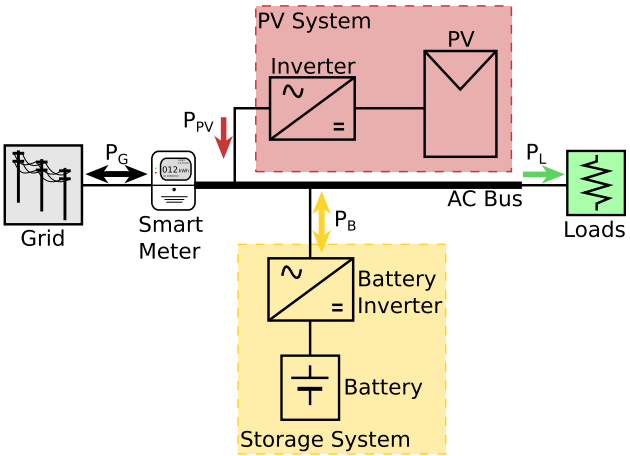

Fig. 3 Schematic representation of the controllable facility, where the arrows represent the possible direction of power flows

3 Derivative algorithm

In order to enhance the grid, the proposed algorithm has to be able to adapt the power of the controllable facilities to the non-controllable ones without any information apart from the one of the aggregated consumption and the power its facility is consuming ($p_i(t)$). Thus, there is no communication between the facilities of the grid. The objective of the algorithm consists of being able to meet the condition of Eq. 4.

$$P(t) = \sum_{i=1}^{m} p_i^{\mathrm{nc}}(t) + \sum_{i=1}^{n} p_i^{\mathrm{c}}(t) = K \tag{4}$$

Then, from Eq. 4, it can be deduced that the result of applying the algorithm should be

$$\frac{\mathrm{dP(t)}}{\mathrm{dt}} = 0 \tag{5}$$

Equation 5 also implies that $P(t)$ is constant, but the different parts, that integrate it, are not necessarily constant. Therefore, the algorithm would modify the response of each of the $p_i^{\mathrm{c}}(t)$ to adapt in real time to the $p_i^{\mathrm{nc}}(t)$. In addition, it is known that $P(t)$, with no active control of the facilities, presents strong periodicity components corresponding to the 12 h, 24 h and 11 week. [7] developed a solution which consisted of building a distributed band-stop filter through the synchronization of the different users. On the contrary, in this paper the objective is to neutralize $P^{\mathrm{nc}}(t)$ through the construction of an interference signal ($P^{\mathrm{c}}(t)$) that opposes to it, resulting in a flattened $P(t)$. This approach is inspired in Active Noise Control (ANC) which is a method for reducing an unwanted sound by the addition of a second one specifically designed to cancel it [14]. The noise source is counteracted by another signal with the same amplitude, that will either phase shift or invert the polarity of the original signal. Then, both signals are combined into a new one.

For the shake of simplicity, let's assume that there are only two facilities: one non-controllable and one controllable, which only consume power and there is no local generation and which present a continuous and periodic consumption profile such as the one of the members of a real grid. Then, $P(t)$ as the sum of the two facilities would be periodic, being similar to the response of the aggregated consumption of a grid. The chosen signal to represent the consumption of the facilities is of sinusoidal nature being easy to use and describe its properties. Thus, $P^{\mathrm{nc}}(t)$ and $P^{\mathrm{c}}(t)$ have a sinusoidal waveform, and they can be expressed as in Eq. 6.

$$P^{\mathrm{nc}}(t) = A^{\mathrm{nc}} \cdot \cos(\omega t + \phi^{\mathrm{nc}}) + \mu^{\mathrm{nc}} \tag{6a}$$

$$P^{\mathrm{c}}(t) = A^{\mathrm{c}} \cdot \cos(\omega t + \phi^{\mathrm{c}}) + \mu^{\mathrm{c}} \tag{6b}$$

where A is the amplitude, ω is the frequency, ϕ is the phase, and μ is the mean value, since consumption is always greater or equal to zero. There exist two periodic consumptions greater than zero with the same periodicity and each period corresponds to a day. By varying the parameters of the sinusoid, different waveforms can be obtained which represent the consumption of each facility. An example of the waveform of $P^{nc}(t)$ and $P^{c}(t)$ is shown in Fig. 4a. The result of the aggregated consumption can also be observed in Fig. 4b. The form of $P(t)$ is also sinusoidal since it is the sum of two sinusoids of the same period and its expression is calculated in Eq. 7.

$$\begin{aligned} P(t) &= P^{nc}(t) + P^{c}(t) \\ &= A^{nc} \cdot \cos(\omega t + \phi^{nc}) + \mu^{nc} + A^{c} \cdot \cos(\omega t + \phi^{c}) + \mu^{c} \\ &= A \cdot \cos(\omega t + \phi) + \mu \\ \text{where } A &= \sqrt{(A^{nc} \cdot \cos(\phi^{nc}) + A^{c} \cdot \cos(\phi^{c}))^2 + (A^{nc} \cdot \sin(\phi^{nc}) + A^{c} \cdot \sin(\phi^{c}))^2} \\ \phi &= \arctan\left(\frac{A^{nc} \cdot \sin(\phi^{nc}) + A^{c} \cdot \sin(\phi^{c})}{A^{nc} \cdot \cos(\phi^{nc}) + A^{c} \cdot \cos(\phi^{c})}\right) \\ \mu &= \mu^{nc} + \mu^{c} \end{aligned} \tag{7}$$

The resulting $P(t)$ is expressed in terms of $P^{nc}(t)$ and $P^{c}(t)$. and it also corresponds to a sinusoidal waveform depending on the parameters of the $P^{nc}(t)$ and $P^{c}(t)$. The three signals $P(t)$, $P^{nc}(t)$ and $P^{c}(t)$ are formed by a constant part (μ) and an oscillating part (sinusoid). Then, the algorithm seeks to cancel the alternating part of $P(t)$. To cancel it, the parameters of the controllable consumption are modified to

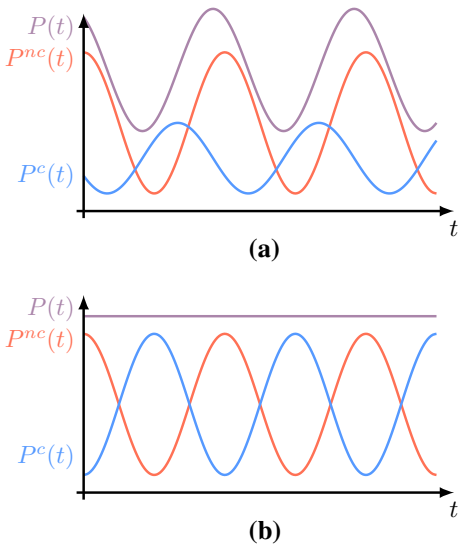

Fig. 4 Environment simplification using sinusoidal signals as the demand profile for the different grid elements: **a** $P^{c}(t)$ is not adapted to $P^{nc}(t)$, so the form of $P(t)$ is not constant because the demand is arbitrary and **b** $P^{c}(t)$ is adapted to $P^{nc}(t)$, being $P(t)$ constant. $P^{c}(t)$ is drawn in blue, $P^{nc}(t)$ is drawn in red and $P(t)$ is drawn in purple (color figure online)

adapt to the non-controllable part and the conditions are gathered in Eq. 8.

$$P(t) = \cancelto{0}{A \cdot \cos(\omega t + \phi)} + \mu = \mu^{nc} + \mu^{c}$$

$$then,\ A_c \cdot \cos(\omega t + \phi_c) = -A_{nc} \cdot \cos(\omega t + \phi_{nc}); \tag{8}$$

$$\begin{cases} A_c = -A_{nc} \\ \phi_c = \phi_{nc} \end{cases}$$

Moreover, to remove the oscillating parts of Eq. 8, the algorithm reconfigures the controllable signal to create a destructive interference with those conditions. Therefore, the aggregated consumption is smoothed. Figure 4b shows the result of modifying $P^{c}(t)$ in order to cancel the alternating part of the $P^{nc}(t)$ getting a constant $P(t)$. In this case, it can be observed that one signal is opposite to the other and they are in antiphase or opposite phase with the same amplitude. When one signal grows, the other one decreases to compensate the growth. This effect causes $P(t)$ being at equilibrium and constant over time.

In this case, the non-controllable demand has an analytical expression and it behaves the same over time. Thus, the construction of the controllable demand signal can be easily adjusted to interfere with the rest of the demand and be opposite to it. However, in general, there is no information about the analytical expression of $P(t)$ and how it evolves over time. So, the algorithm must ensure that the condition of Eq. 4 is satisfied regardless the waveform of $P^{nc}(t)$. The algorithm will seek for a $P^{c}(t)$ that satisfies,

$$P^{c}(t) = K - P^{nc}(t) \tag{9}$$

Nevertheless, it is difficult to calculate the exact value of $P^{c}(t)$ needed to counteract $P^{nc}(t)$. The reason is that the information of $P(t)$ is not available at the moment the algorithm has to calculate and takes a decision of what power the controllable facility has to consume. This measure is only available after all the components of the grid have consumed power and the grid response evolves depending on how much power was consumed. So, the algorithm has to apply a different strategy that is not based on the instant value of $P(t)$ but based on historical values enclosing information about it. A first approach should consist of using the tendency of $P(t)$. Thus, the condition is based on the trend of $P(t)$ and how it changes through time. If Eq. 5 is expressed in terms of the components of the grid, the following relationship is achieved

$$\frac{dP(t)}{dt} = 0 \implies \frac{dP^{c}(t)}{dt} = -\frac{dP^{nc}(t)}{dt} \tag{10}$$

$P^{c}(t)$ in order to neutralize the variability of $P^{nc}(t)$ should grow when $P^{nc}(t)$ decays and vice versa. So the tendency of $P^{c}(t)$ should oppose to $P^{nc}(t)$ and it has to grow or decay

at the same rate as $P^{\mathrm{nc}}(t)$. However, there is no information about the form of the non-controllable demand. The only information available is $P(t)$ and the local power consumed by the facility $p_i^{\mathrm{c}}(t)$. From the point of view of one controllable user, the rest of the demand is non-controllable. Therefore, for the jth controllable facility, the aggregated consumption, from its individual point of view, is as follows

$$p_j^{\mathrm{c}}(t) = P(t) - P^{\mathrm{nc}}(t) - \sum_{i=1}^{n-j} p_i^{\mathrm{c}}(t) = P(t) - \hat{P}^{\mathrm{nc}}(t) \tag{11}$$

where $\hat{P}^{\mathrm{nc}}(t)$ groups together the rest of the grid consumption except the one coming from the local facility. Then, the local consumption of one facility is negligible compared to the sum of the rest of consumptions. Normally, a grid is composed by a huge number of facilities, so it is normal that $\hat{P}^{\mathrm{nc}}(t) \approx P(t)$. Thus, a controllable facility seeks to modify its consumption opposing to the rest of the components of the grid. If Eq. 10 is developed for one individual, it can be obtained that

$$\frac{\mathrm{d}p_j^{\mathrm{c}}(t)}{\mathrm{d}t} = -\frac{\mathrm{d}\hat{P}^{\mathrm{nc}}(t)}{\mathrm{d}t} \approx -\frac{\mathrm{d}P(t)}{\mathrm{d}t} \tag{12}$$

As aforementioned, each facility will use a neural controller that should acquired the dynamic behavior of the grid to counteract the effects of the rest of the users based on the derivate of the aggregated consumption. ANNs should oppose to the derivate of $P(t)$ with the only information of the grid signal, $P(t)$, and the local behavior of its facility, $p_i^{\mathrm{c}}(t)$.

4 Neural controller

Figure 5 shows a simplified environment with two facilities: Facility A, being the non-controllable part of the demand, and Facility B, which is the controllable one. Both

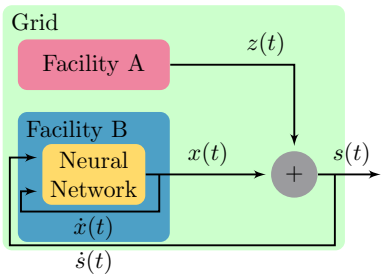

Fig. 5 Electrical grid consisting of two facilities: facility A does not have a controller and Facility B has a neural controller that control its demand. $z(t)$ represents the non-controllable demand, $x(t)$ represents the controllable demand, and $s(t)$ is the environment signal or the aggregated consumption

facilities only consume power, and there is no generation and no energy storage system. The different signals of the environment are renamed in order to facilitate its nomenclature. $P^{\mathrm{nc}}(t)$ becomes $z(t)$, $P^{\mathrm{c}}(t)$ is $x(t)$ and $P(t)$ is renamed as $s(t)$. Then, the environment equation is as follows,

$$s(t) = z(t) + x(t) \tag{13}$$

The controller has to extract the information from the history of the signals and build a destructive interference to obtain a flattened aggregated consumption. The inputs available are the derivatives of the facility local behavior, $\dot{x}(t)$, and the environment aggregated consumption, $\dot{s}(t)$, both of them with respect to time. In this case, the output of the neural controller is of the form,

$$x(t) = f(\dot{s}(t), \dot{x}(t)) = s(t) - z(t) \tag{14}$$

4.1 Neural structure

- *Structure* The neural structure selected is composed by one input layer, one or two hidden layers, that will be explained later, and an output layer.
  - *Input* This layer is composed by only two neurons because there are only two inputs, $\dot{s}(t)$ and $\dot{x}(t)$. Both neurons receive both inputs multiplied by a gain.
  - *Hidden* The number of hidden layers is selected between one or two layers. The number of neurons in each layer is varied from 2 to 4.
  - *Output* The output of the network is directly $x(t)$ which should be in antiphase to cancel the fluctuations of the environment signal.
- *Neuron function* The function that each neuron computes is described in Eq. 15.

$$\begin{aligned}\dot{y}_i(t) &= f_i(I_i(t), y_1(t), \ldots, y_n(t)) \\ &= \frac{1}{\tau_i} \cdot \left(-y_i(t) + \sum_{j=1}^{N_{pre}} w_{ij} \cdot \sigma_i\big(y_j(t) + \theta_j\big) + \sum_{m=1}^{n_{in}} g_m \cdot I_m(t) \right)\end{aligned} \tag{15}$$

  where y_i is the activation of the ith neuron, $\dot{y}_i$ is the rate of change of the ith activation neuron, τ_i is the time constant, w_{ij} is the connection weight of the jth to the ith neuron, $\sigma_i(\cdot)$ is the activation function, θ_i is the bias of the neuron, and I_m is the external input of the neuron (if any), which is multiplied by a gain, g_m. The network is composed by homogeneous neurons, so all of them compute the same function. The output of each neuron of the network equals $\sigma_i(y_i(t) + \theta_i)$.
- *Flow of information* The RNN(CTRNN) has been designed in such a way that each neuron of the input layer has a feedback loop from its output to its input. In

the hidden layer, the feedback loop is more complex and the output of the neurons is connected from the output to the input of the rest of the neurons and itself. The structure of the hidden layer resembles the one of the Elman networks [15] in which the state of the network is stored in a group of neurons connected to the hidden layer. Finally, in the output layer, the neurons are connected as in the input layer.

Based on these tips, a first approximation of the neural controller structure is elaborated. Section 4.2 explains the training performed to obtain the fittest structure.

4.2 Neural parameters

Different simulations have been prepared in which the performance of various CTRNN architectures is compared.

A genetic algorithm (GA) [19] has been used to adjust the free parameters of the neural structure. The definition of a fitness function will be based in Eq. 12, evaluating the result of applying the neural controller. The main objective of the GA consists of evolving the neural structure of the CTRNN in order to minimize the fitness function. All the possible neural parameters are divided into two groups:

- *Structures* 12 structures, with different number of neurons distributed in one or two layers, are going to be evaluated. Figure 6a represents the architectures with one hidden layer, whereas Fig. 6b shows the architectures with two hidden layers. In each hidden layer, the number of neurons varies from 2 (solid units) to 4 neurons (dashed units), making a total of 3 different architectures with only one hidden layer and 9 architectures with all the possible combinations of neurons in both layers.
- *Free parameters to adjust* From Eq. 15, the free parameters of the network are: τ_i, w_{ij}, θ_i and g_m. g_m and w_{ij} can be grouped together because they represent the connections of different parts of the network. The value of τ_i is going to be fixed to 1 because it is not necessary to variate the reaction speed of the neurons at the moment. So the parameters of the network used during the evolution are w_{ij} and θ_i. 6 different intervals are used in order to know in which range of parameters the network performs better and achieves the best results. These 6 intervals are divided in two types: $[0, a]$ and $[-a, a]$ with $a \in \{1, 5, 10\}$. Thus, there are 3 intervals with only positive values and 3 with positive and negative values. The reason is that it is necessary to test whether the problem can be solved with only excitatory connections (positive values) or inhibitory weights (negative values) must be added. Both the synaptic connections, w_{ij} and g_m, and the bias, θ_i, take their values from the same interval. In each simulation scenario, only one parameter interval is used generating 72 experiments for each structure and interval.

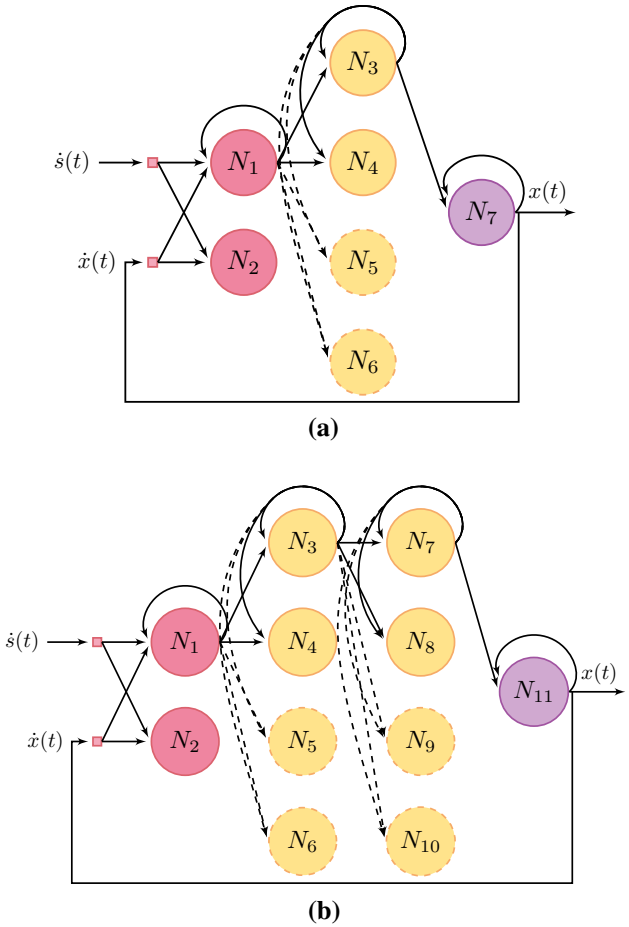

Fig. 6 CTRNN structure of the hidden layer: **a** one hidden layer and **b** two hidden layers. The synaptic weights of each neuron of the layer are represented in the first neuron of it. The minimum size of the hidden layer is two neurons (drawn with solid lines), whereas the maximum number of neurons is four (drawn with dashed lines)

4.3 Genetic algorithm configuration

A genetic algorithm is used to evolve the CTRNN structure. This GA is composed by the three basic operations, selection, crossover and mutation, over a population of individuals whose performance will be evaluated based on a fitness function. The different parameters of the GA and its values are as follows:

- *Chromosome* The length of the chromosome depends directly from the architecture of the CTRNN. Thus, the chromosome is composed by the different w_{ij} and θ_i, as shown in Eq. 16.

$$l_{\text{ch}} = \overbrace{N_{\text{in}} \cdot (n_{\text{in}} + 1)}^{\text{input}} + \overbrace{\sum_{l=1}^{L} N_{h,l} \cdot \left(N_{pl,l} + N_{h,l}\right)}^{\text{hidden}} + \overbrace{N_o \cdot \left(N_{pl} + 1\right)}^{\text{output}} + \overbrace{\underset{i}{N_T}}^{\theta} \tag{16}$$

where l_{ch} is the length of the chromosome, N_{in} is the number of neurons in the input layer, n_{in} is the number of inputs associated with the g_m terms, $N_{h,l}$ is the number of neurons of the lth hidden layer, $N_{pl,l}$ is the number of neurons of the previous layer, N_o is the number of neurons in the output layer, and N_T is the total number of neurons. The minimum length of the chromosome corresponds to the neural architecture with only 1 hidden layer and 2 neurons in it, and it has a value of 22 genes, whereas the maximum length corresponds to the structure with 2 hidden layers and 4 neurons in each of them, that is 78 genes.

- *Population size* The population size affects the speed to find the solution. The higher the size of the population is, the greater the number of solutions is tried. During the tuning process, different population sizes are tested to find the optimum number of individuals per generation. The size of the populations varies between 20 and 100 individuals with intervals of 10 individuals of difference between them. Thus, 9 different population sizes are tested.
- *Generations* The number of generations should be high enough so that the algorithm can reach the solution within it and was fixed to 1000 generations.
- *Selection* The one selected is the roulette-wheel method where a number of individuals are selected randomly based on their probability and its number is based on the population size. However, based only on the randomness of the process, diversity and the best fitted solution of the problem can be lost. To solve this problem, a combination of the roulette-wheel method with the elitism is selected. Thus, a number of individuals of the next generation is reserved to copy directly which consists of the best individuals of the previous generation. In this case, the number of elites is fixed to 6.
- *Crossover* The crossover operator is applied over the individuals obtained from the previous selection. The idea of the operator is to choose two individuals, select randomly a point in their chromosomes and then swap the parts of the chromosomes. Therefore, two individuals are created for the new generation with part of their predecessors. The crossover operator only selects 1 point to cross the chromosome. This point is randomly chosen among the different genes of the chromosome.
- *Mutation* The value of the gene is altered following a Normal Gaussian distribution with a variance of $\sigma = 0.2$, whose value will be added to the gene in order to modify its value. Different mutation rates are used. These rates are between 0.01 and 0.1 in steps of 0.01. So, there are 10 different mutation rates to select the best choice for our algorithm.
- *Fitness function* FF(.) is in charge of evaluating the performance of the different chromosomes. The condition of the $\text{FF}_i(\dot{s}(t))$ is relaxed, and a gradual transition is made between the two states. Equation 17 shows a gradual fitness function formulation:

$$\text{FF}_i(\dot{s}(t)) = \begin{cases} 1 & \text{if } |\dot{s}(t)| < \varepsilon \\ -\dfrac{|\dot{s}(t)| - \delta}{\delta - \varepsilon} & \text{if } \varepsilon \le |\dot{s}(t)| \le \delta \\ 0 & \text{if } |\dot{s}(t)| > \delta \end{cases} \tag{17}$$

where ε is an inferior limit and δ is a superior limit. $\varepsilon << \delta$. If ε is too low, the solution is nearer to $\dot{s}(t) = 0$ and the GA will take a long time to converge. If $\varepsilon \approx \delta$, the fitness function has a discontinuity of two states, being difficult to evolve. If δ is too high, then almost all the individuals will be able to score and the GA will not evolve. Based on these concepts, these limits are fixed to $\varepsilon = 10^{-5}$ and $\delta = 10^{-2}$. During the simulation results, this assumption will be checked. In addition, it is necessary to discretize $\dot{s}(t)$ used in the fitness function. Thus, a first approximation of $\dot{s}(t)$ is based on the definition of the derivative.

$$\dot{s}(t) = \lim_{x \to 0} \frac{s(t + \Delta t) - s(t)}{\Delta t} \approx \frac{s[k + \Delta k] - s[k]}{\Delta k} \underset{\Delta k = 1}{=} s[k+1] - s[k] = \Delta s[k] \tag{18}$$

However, from the difference of one sample, it might be possible that the CTRNN could not extract the tendency behavior of the signal. Thus, a second definition is proposed consisting of the mean of an interval of samples of the environment signal. Equation 19 reflects this behavior.

$$\Delta s[k] = \frac{1}{N_s} \sum_{l=k-N_s}^{k-1} s[l+1] - s[l] \tag{19}$$

where N_s is the number of samples over with the mean is elaborated and with $k \ge N_s$. A value of $N_s = 10$ samples is used to verify that this approximation is correct and the better results than the instantaneous difference are obtained. So, the same $\text{FF}_i(\Delta s[k])$ function has been defined with 2 approximations of the derivative of the environment signal. Equation 17 shows the value of fitness for only one instant of time,

so for the whole time length the value of individual fitness is in Eq. 20.

$$\mathrm{FF}_i(s) = \frac{1}{K_\mathrm{s}} \sum_{k=0}^{K_\mathrm{s}-1} \mathrm{FF}_i(\Delta s[k]) \tag{20}$$

where K_s is the total number of samples of $s[k]$. The fitness values of the different time instants have been integrated, and then the mean of all of them is taken to obtain the value of an individual. Consequently, the average fitness value of the population is computed in order to know how well the performance of the generation was during that realization (see Eq. 21).

$$\begin{aligned} \mathrm{FF}(s) &= \frac{1}{N_\mathrm{pop}} \sum_{i=1}^{N_\mathrm{pop}} \mathrm{FF}_i(s) \\ &= \frac{1}{N_\mathrm{pop} \cdot K_\mathrm{s}} \sum_{i=1}^{N_\mathrm{pop}} \sum_{k=0}^{K_\mathrm{s}-1} \mathrm{FF}_i(\Delta s[k]) \end{aligned} \tag{21}$$

where N_pop is the number of individuals of the generation. Finally, each generation is evaluated over and over again until the last generation or a plateau value is reached. To sum up, 2 fitness functions are evaluated based on how the derivative of the $s(t)$ is done (see Eqs. 18 and 19).

– *Seeds* For each experiment, 30 different seeds have been used to randomize the initialization of the algorithm. Thus, each experiment is repeated 30 times in different parts of the search space. With this number of seeds, it is enough to discover the dependency with the initial conditions and if the best solutions are reached for these configurations of the GA.

In summary, the different parameters of the simulations are reduced to Tables 1 and 2. Thus, there are 12 different CTRNN structures which use 6 different intervals for their free parameters and they are trained with a GA that has 9 different sizes of populations evolved in 1000 generations with 3 basic genetic operators: selection with 6 elites, 1 crossover and 10 different mutations rates, evaluated for 2 fitness functions. Each experiment is repeated 30 times to randomize the initial conditions of the algorithm, so the number of simulations to be done is 388,800 in total.

Table 1 Configuration of the ANN for the different simulations

ANN parameters	Min	Max
W and g_m	17	67
θ_i	5	11
τ_i	–	1
# Hidden layers	1	2
# Hidden neurons	2	4
Total # of neurons	5	11
Total # structures	12	
Total # intervals	6	

Table 2 Configuration of the GA for the different simulations

Parameters	Min	Max
Chromosome length	22	78
Population size	20	100
# Generations	–	1000
# Elites	–	6
# Crossovers	–	1
Mutation rate	0.01	0.1
# $FF(.)$	–	2
# Seeds	–	30

5 Training

All the simulations under the set of parameters of Tables 1 and 2 are conducted by *NeuralSim* simulator. *NeuralSim* is an open source simulator under GPLv3.0 license developed in C++. It was build as a general purpose simulator to prove different neural structures with different methods to train them, learning algorithms and GAs. All the simulations were run in a computing cluster of 60 unit processors of *AMD Opteron(tm) Processor 4180* with 40 GB of RAM memory, under a distribution *Rocks 6.1.1 (Sand Boa)* of 64-bit architecture.

A continuous periodic signal of class C^1, a sinusoidal function, is used to evaluate the neural structure. The sinusoid is used with a day period and 72 samples per period to simulate a peak and a valley. Moreover, the input signal is formed of 3 periods of 72 samples in order to isolate the initial state of the CTRNN and reach a stable state to evaluate the performance of the neural controller.

Figure 7 shows the overall performance per structure for all possible combination of parameters. The nomenclature followed for referring to each structure is of the form $\#ML\#N$, where $\#M \in \{1, 2\}$ is the number of hidden layers of the structure, and $\#N \in \{2, 3, 4\}$ is the number of neurons of the hidden layer, $\#N \in \{2, 3, 4\}$. At a first glance, the performance of each structure is very similar. All boxplots are composed by 32,400 points corresponding to the individual with the best fitness value reached after 1000 generations for all the different configurations of the CTRNN and GA parameters.

For the 12 structures, the value of the higher whisker is located closely to 1.0 which is the maximum value that the FF(.) can take. This assures that in all the structures, there is a combination of parameters that ensures that $\dot{s}[k] < 10^{-5}$,

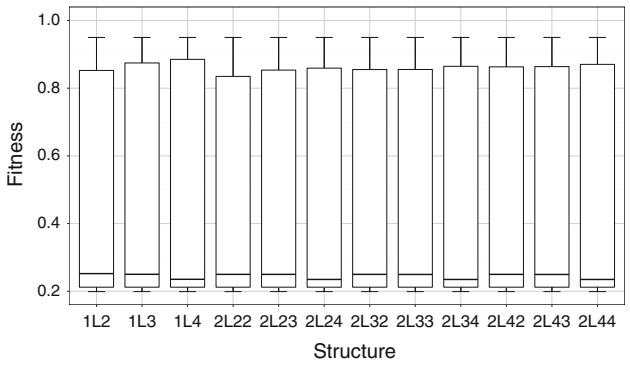

Fig. 7 Overall fitness performance per structure for all the possible configurations of the simulations. The nomenclature of the structure is $\#ML\#N$ with $\#M$being the number of hidden layers and $\#N$being the number of hidden neurons. Each box comprises observations ranging from the first to the third quartile. The median is indicated by a horizontal bar, dividing the box into the upper and lower part. The whiskers extend to the farthest data points that are within 1.5 times the interquartile range. Outliers are shown as dots

where 10^{-5} is the higher limit of the fitness function. On the other hand, the minimum fitness value for all the 12 structures is also near the vicinity of a 0.2 fitness value. Thus, in those cases, the environment signal variations are in between 10^{-2} and 10^{-5} (the limits of the fitness function) since the average fitness value is different from 0. Both whiskers are situated to 1.5, the value of the interquartile range (IQR), or $1.5 \cdot |Q3 - Q1|$. Again, for all the structures the $Q1$ is located above 0.2 but close to it and the 25% of the simulations reached this maximum fitness. Close to the $Q1$ is the median which represents the 50% of the data. The median for all structures is close to a fitness value of 0.25; however, structures with 4 neurons in the last hidden layer have a little lower median value.

Finally, the $Q3$ for all the simulations is above a 0.8 fitness value so that the 25% of the simulations are above this value. Some differences between the values of $Q3$ depending on the structure can be observed. The best value of the $Q3$ corresponds to the structure $1L4$ (1 hidden layer with 4 neurons), but in general the performance of the architectures with only one hidden layer is only 1% above the two layers architectures. This figure seems relatively too low to state which structure is the best. However, it can be concluded that the complexity introduced by the two hidden layers does not achieve better results than the ones with a hidden layer. The tuning is reduced since fewer parameters are used and the processing capabilities are more fitted to requirements of the problem.

The data corresponding to each structure are split in the different ranges of the tuple $\{\boldsymbol{W}, \boldsymbol{\Theta}\}$. Figure 8 shows all the results for the different ranges of values that the parameters of the CTRNN can take. In order to reduce the volume of data, the seeds of the different configurations

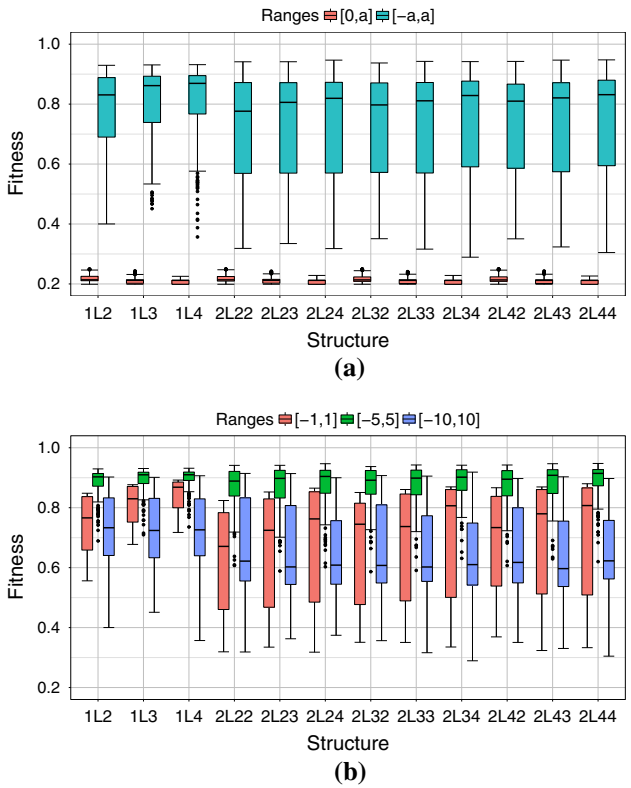

Fig. 8 Performance of the 12 structures divided by the range of values that $\{\boldsymbol{W}, \boldsymbol{\Theta}\}$ can take: **a** a range of values are grouped in two intervals $[0, a]$ and $[-a, a]$. **b** three range of values of the form $[-a, a]$ with $a \in 1, 5, 10$

have been grouped to analyze the mean performance of each evolution since all the structures behave similarly. A first division has been made based on the form of the value ranges. The two ranges selected are: (i) $[0, a]$, only positive values, and (ii) $[-a, a]$, symmetric range of values.

For all the structures, when $\{\boldsymbol{W}, \boldsymbol{\Theta}\} \in [0, a]$ there is no difference in the fitness value for each one. In all these cases, the structure is far from reaching the maximum value of the fitness, so hardly these structures are evolved. The neurons are saturated due to the monotonically growing of the activation function together with the positive synapse, making impossible to evolve the structure to reach its maximum. In addition, the deviation of the data is too small for all the structures and it is less than $IQR < 0.02$ (see Fig. 8a). Thus, it can be assumed that for this type of ranges, the neural structure does not evolve properly and this type of configuration can be discarded.

On the other hand, for structures in which $\{\boldsymbol{W}, \boldsymbol{\Theta}\} \in [-a, a]$, the maximum fitness is almost reached. The maximum whisker for all of them is between 0.9 and 1.0 fitness value. It can be observed that the median is slightly higher for structures with only one layer rather than the ones with two layers. Moreover, the data deviation presented in one-layer structures is less than half the

deviation of the two-hidden-layer structures. Structures with two hidden layers have very similar performances, and they are not so different to each other. On the other hand, the increase in neurons in one hidden layer structures reduces the variance of the result.

Figure 8b shows the performance for each value of $a \in \{1, 5, 10\}$ for the range of values of the form $[-a, a]$. For all the structures, the best performance is achieved for the range of values in which $a = 5$. The worst performance is for the one in which $a = 10$, which presents lower median values than the other two intervals. The reason is because of the length of the interval, since there are more parameters to choose and it makes difficult the GA to find a better solution in the same number of generations. In spite of having more dispersion than the rest of values ranges, $a = 1$ has a higher performance than $a = 10$ since the median for all the structures is higher. However, it is still lower than the performance of $a = 5$ for all the structures.

Therefore, the best performance is the one in which $\{\boldsymbol{W}, \boldsymbol{\Theta}\} \in [-5, 5]$. In order to help with the decision of which structure presents a better performance in this interval, the numerical information has been gathered in Table 3.

Among single hidden layer structures, there are not significant differences among them, according to Table 3, but in the overall performance of the three value ranges, *1L4* is clearly the best of the three. Finally, the best fitted structure for the problem is ***1L4*** with $\{\boldsymbol{W}, \boldsymbol{\Theta}\} \in [-5, 5]$.

6 Post-evaluation

A process of post-evaluation is performed to test the evolution. This process will consist of studying the output of the evolved neural controller for the characteristics of the environment described in Fig. 5. In order to carry out the post-evaluation, 7 figures of merit are declared. To analyze the waveform of $s[k]$, 4 central moments have been used which are computed in terms of the signal mean value, the next 2 coefficients evaluate its variability, and the last one measures when the algorithm reaches a stable operating mode:

- *Mean* ($\mu_0 = \mu$) The mean is the central value that in average the signal takes through a time period. This parameter is calculated as in Eq. 22.

$$\mu_0 = \mu = E[X] = \frac{1}{K}\sum_{k=1}^{K} s[k] \tag{22}$$

- *Variance* ($\mu_2 = \sigma^2$) The variance is a measure of the dispersion of the data around a value, in this case μ. For the experiment, it will indicate how far the points of $s[k]$ are spread out from its average. σ^2 is the second central moment and its mathematical expression is as follows:

$$\mu_2 = \sigma^2 = E\left[(X - E[X])^2\right] = \frac{1}{K}\sum_{k=1}^{K}(s[k] - \mu)^2 \tag{23}$$

- *Skewness* (μ_3) This parameter measures the asymmetry of the data. Thus, it indicates how the signal is distributed around its average in a time period. It can be interpreted for positive values as the signal is above μ most of the time. Whereas if it is below μ, it indicates that most of the time the signal is below the data. In case of 0, the data are around the mean. This is the third central moment, and it is calculated as:

$$\mu_3 = E\left[(X - E[X])^3\right] = \frac{\frac{1}{K}\sum_{k=1}^{K}(s[k] - \mu)^3}{\sigma^3} \tag{24}$$

Table 3 Statistical data of the simulation performance for all the structures where $a = 5$

Structure	Min	Q1	Median	Q3	Max	IQR	Mean	SD
1L2	81.92	87.16	90.28	91.41	92.95	4.25	88.51	4.46
1L3	82.63	88.05	90.98	91.85	93.08	3.80	89.29	4.17
1L4	85.34	89.08	90.99	91.89	93.16	2.81	89.84	3.36
2L22	71.51	83.88	88.88	92.12	94.12	8.24	86.93	6.82
2L23	70.16	83.23	89.78	92.49	94.14	9.26	87.04	7.00
2L24	74.29	84.80	90.43	92.48	94.68	7.69	87.54	6.98
2L32	72.67	84.47	89.17	92.42	93.73	7.95	87.33	6.40
2L33	72.02	84.31	89.87	92.55	94.26	8.24	87.47	6.83
2L34	76.75	85.72	90.18	92.67	94.18	6.95	88.35	5.65
2L42	72.32	83.96	89.51	92.36	94.25	8.40	87.22	6.57
2L43	75.56	84.67	90.81	92.71	94.68	8.04	88.09	6.46
2L44	79.53	87.28	91.45	92.73	94.76	5.45	89.07	5.62

The fitness value is expressed in %

– *Kurtosis* (μ_4) This is another measure of the form of the signal as μ_4. In this case, the kurtosis is studying the dispersion of a signal related to the average through the points closest to it compared with those points of the distant ends. Therefore, a high value of μ_4 means that there are lots of points around the mean, but at the same time lots of them are also in the ends of the signal. On the contrary, a low value means that all the points are concentrated in the average of the signal. The mathematical expression is:

$$\mu_4 = E\left[(X - E[X])^4\right] = \frac{\frac{1}{K}\sum_{k=1}^{K}(s[k]-\mu)^4}{\sigma^4} \tag{25}$$

– *Coefficient of variation* (c_v) This parameter establishes a relationship between the average of the signal and its variance. When the data are around the mean, this coefficient is next to 0 and otherwise the data will present mean deviations. This coefficient is described mathematically in Eq. 26.

$$c_v = \frac{\sigma}{\mu} \tag{26}$$

– *Crest factor* (C_f) The last parameter related to the form of the signal is the crest factor. This factor measures the waveform through the ratio of maximum values of the signals compared to its effective value. Thus, it indicates how high the peaks are present in the form of the signal. A C_f of 1 indicates that the signal does not have any present peak and it is constant, whereas higher values indicate how accused those peaks are. This factor compares the maximum of a signal with respect to rms of the signal for a time period (see Eq. 27).

$$C_f = \left.\frac{|s|_{\text{peak}}}{s_{rms}}\right|_K \tag{27}$$

– *Time of convergence* (t_c) The last parameter used is the convergence time of the neural controller to flatten the environment signal. It is a measure of the stability of the algorithm indicating when the signal becomes stable and abandons the transitory state. Hence, it is considered that the environment has reached a stable state when the σ^2 is less than an established minimum during a period of time. This is expressed in Eq. 28.

$$t_c = \left|\sigma^2\right| < \sigma^2_{\min}\Big|K \tag{28}$$

The different measurements have been introduced to evaluate the response of the evolved neural controller. The best chromosome with higher fitness value has been chosen for the neural architecture of 1 hidden layer and 4 neurons inside it with its free parameters of the form $\{\boldsymbol{W}, \boldsymbol{\Theta}\} \in [-5, 5]$. To begin the post-evaluation, it is necessary to select several waveforms of $z[k]$ that the neural controller has to smooth. 6 different signals have been selected, each with different characteristics. These signals are chosen in order to evaluate the adaptability and the ability of generalization of the network. Their mathematical expressions are gathered in Eq. 29f.

$$z_1[k] = 0 \tag{29a}$$

$$z_2[k] = \sin(f \cdot k) \tag{29b}$$

$$z_3[k] = \sin(2 \cdot f \cdot k) \tag{29c}$$

$$z_4[k] = \sin\left(\frac{f \cdot k}{2}\right) \tag{29d}$$

$$z_5[k] = \begin{cases} a \cdot k & \text{if } 0 \le k < \dfrac{K}{2}, \\ 1 - a \cdot k & \text{if } \dfrac{K}{2} \le k \le K. \end{cases} \tag{29e}$$

$$z_6[k] = \begin{cases} 0 & \text{if } 0 \le k < \dfrac{K}{2}, \\ 1 & \text{if } \dfrac{K}{2} \le k \le K. \end{cases} \tag{29f}$$

The first signal $z_1[k]$ of Eq. 29a is a constant signal of zero value. This signal is used to test the response of the network in absence of any other element, only its own output. Then, $z_2[k]$ is a periodic sinusoid with the same frequency of the one used during the evolution (see Eq. 29b). $z_2[k]$ was chosen to test that the fitness value obtained after the evolution was correct, and it is able to compute the opposed derivative of the input signal. $z_3[k]$ and $z_4[k]$, whose expressions are gathered in Eqs. 29c and 29d, are both sinusoid waveforms, but with different frequencies. $z_3[k]$ has a frequency double of the sinusoid of $z_2[k]$ and $z_4[k]$ has half of the frequency. These two signals test how the network responds to changes in the speed of the waveforms. Finally, two other signals were defined that present some peculiarities in their form and in the derivative of both of them. Both are periodic but its derivative suffers some sudden changes. $z_5[k]$ is a triangular signal which is periodic and continuous, but the sign of the derivative changes suddenly at the peak (see Eq. 29e). And $z_6[k]$ is a square signal with a duty cycle of 50% (half of the period is at 0 and the rest is at 1), which is also periodic but not continuous (see Eq. 29f). In both cases, these signals are used to analyze what happen when the derivative change ($z_5[k]$) or it is undefined ($z_6[k]$) to evaluate the response of the network to signals of different nature to the ones proposed for the training.

The post-evaluation was done by evaluating the behavior of $s[k]$ over a period of the waveform selected when the algorithm has reached its stable state. Moreover, normality in the data output has been checked prior to post-evaluation in order to provide reliable interpretation.

Notice that geometric aspects of robust testing must be taken into account [33]. All the results to the signals described above are presented in Table 4. In addition, Fig. 9 shows the different waveforms for each of the signals post-evaluated. In general, the algorithm is able to compute the derivative of the environment signal, to oppose to it and to flatten it. Moreover, it also converges in few steps to a stable state in which it will remain until the end of the simulation.

However, the results are not the same for all the signals due to its characteristics. In the case of the absence of input, $z_1[k]$, it can be observed in Fig. 9a that $x[k]$ is constant and also $s[k]$ since there is no other signal. Thus, in the absence of $z[k]$, the network obtains a constant output. Table 4 shows that μ is located at a value of 1 and the rest of the dispersion parameters are *zero*. The C_f has the best value that it can take which is 1 because it does not present any peaks. The t_c of the network is quickly, in 14 time steps it reaches the stable state in which it will remain until the end.

The next signal, $z_2[k]$ is the same signal used for the training, so it is expected a high performance. Table 4 shows that $s[k]$ is centered in 1 again and there is not any dispersion around μ. Its σ^2 is negligible, the skew of the signal is near zero and they are very concentrated in the mean since the kurtosis is also negligible. The c_v also indicates that the dispersion of the data is small and C_f confirms the absence of peaks for $s[k]$ since is near to the optimum. t_c is again very low in 25 time steps the algorithm has already converged. The waveforms of Fig. 9b show that $x[k]$ is in antiphase to $z_2[k]$ to produce a flattened $s[k]$.

$z_3[k]$ and $z_4[k]$ are again two sinusoids to prove the behavior of the network when the frequency of the sinusoid varies. In the case of $z_3[k]$, it can be observed in Fig. 9c that the controller does not behave as great as in Fig. 9b, because it presents a ripple in $s[k]$. In Table 4, the moments σ^2, μ_3 and μ_4 are higher than ones for $z_2[k]$ so the data dispersion is bigger. And also the form of the waveform presents some peaks that are also present in the value of C_f. In addition, t_c is approximately double the one of $z_2[k]$. On the contrary, when the frequency is lower, the results are very similar to the one obtained for $z_2[k]$. Low values for the dispersion coefficients, σ^2, μ_3, μ_4 and c_v. And also a very close value of C_f to 1, so that it has not any pronounced peak. t_c is also as low as $z_2[k]$. Thus, the neural controller behaves similar for frequencies less than the one selected for the evolution. For frequencies higher than the one selected in the evolution, they reduce the variability of the signal but they present some peaks in the resultant waveform.

$z_5[k]$ has little dispersion around the mean as the value of the σ^2 and c_v are low (see Table 4). In addition, the data are concentrated in the mean and with no skew above or below due to moments μ_3 and mu_4 are also low. The waveform is also really flatten as it can be seen in Fig. 9e and checked with a C_f close to 1. And the convergence time is low, only 35 time steps. On the other hand, the performance of $z_6[k]$ is the worst compared to the rest of signals. There is more dispersion around the mean because the σ^2, μ_4 and c_v are higher compared to the rest. The data are not skewed because it presents a relative low value. About the form in Fig. 9f, the presence of peaks can be observed at the point in which the signal changes of state. Therefore, C_f has a high value far from 1 because of the presence of those peaks. However, the response of the network is very fast and in only 24 steps the network has converged.

Table 5 shows a comparison summary of the algorithm application. In the case of $z_1[k]$, there is no improvement due to the absence of signal at the input of the network. There is a reduction in C_f in all cases. In the best of the cases (for signals $z_2[k]$ and $z_4[k]$), a reduction of 60% was achieved, whereas in the worst case (for signal $z_6[k]$), a reduction of 15% was achieved. Thus, the application of the algorithm always reduces the peaks, increasing the flattening of the $s[k]$.

A last trial was carried out in order to test how the neural controller performs with a signal similar to the grid aggregated consumption. Thus, an artificial grid signal is made based on the principal components of its spectrum. It is used the sum of three sinusoids with the weekly, daily and half daily frequency components. The mathematical expression is gathered in Eq. 30.

Table 4 Summary of the post-evaluation results for the 6 different signals

Signal	μ	σ^2	μ_3	μ_4	c_v	C_f	t_c
$z_1[k]$	1.000	0.00	0.00	0.00	0.00	1.0000	14
$z_2[k]$	0.999	$2.98 \cdot 10^{-7}$	$4.77 \cdot 10^{-7}$	$-1.13 \cdot 10^{-6}$	$5.46 \cdot 10^{-4}$	1.0008	25
$z_3[k]$	1.003	$1.59 \cdot 10^{-3}$	$-1.67 \cdot 10^{-5}$	$9.65 \cdot 10^{-6}$	$3.98 \cdot 10^{-2}$	1.0880	56
$z_4[k]$	0.998	$4.65 \cdot 10^{-5}$	$-2.38 \cdot 10^{-7}$	$2.98 \cdot 10^{-7}$	$6.83 \cdot 10^{-3}$	1.0101	28
$z_5[k]$	1.003	$3.67 \cdot 10^{-5}$	$-9.54 \cdot 10^{-7}$	$1.91 \cdot 10^{-6}$	$6.04 \cdot 10^{-3}$	1.0149	35
$z_6[k]$	0.994	$6.68 \cdot 10^{-2}$	$8.97 \cdot 10^{-4}$	$6.37 \cdot 10^{-2}$	$2.60 \cdot 10^{-1}$	1.9468	24

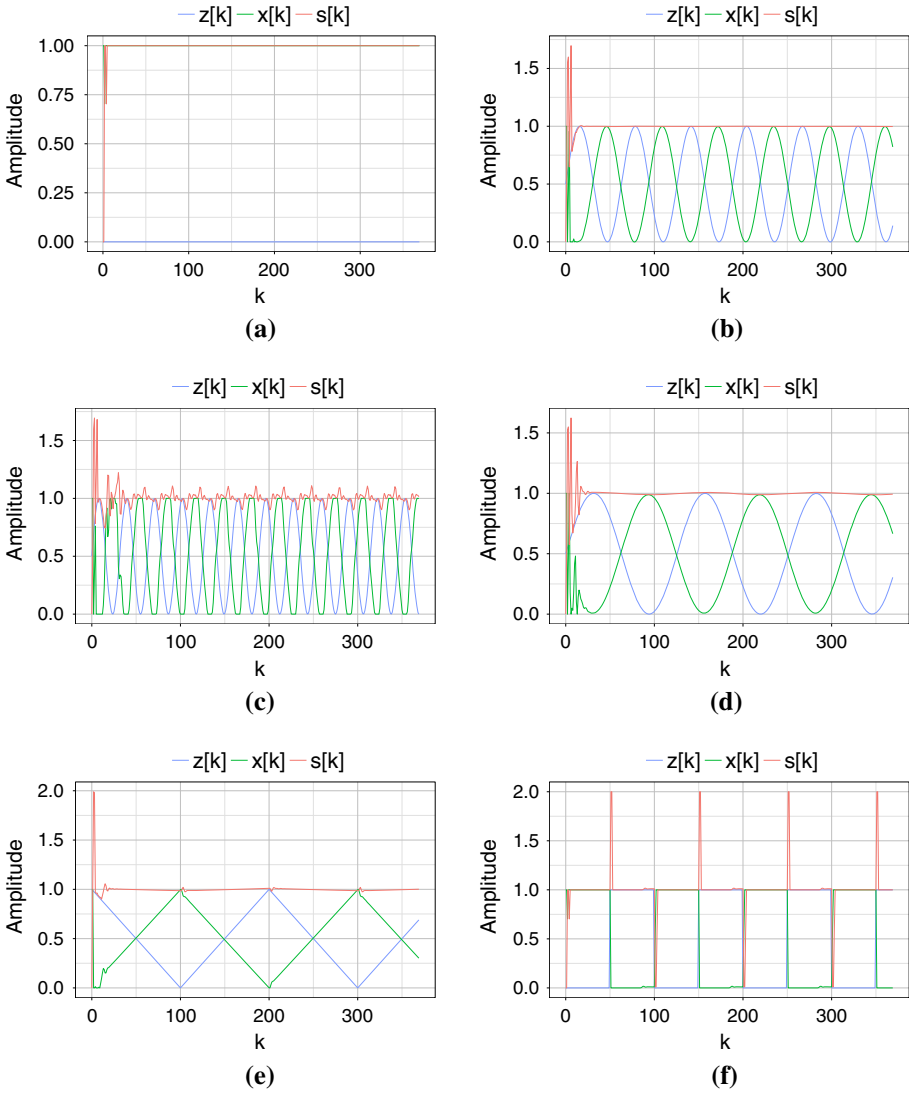

Fig. 9 Waveforms for the post-evaluated signals selected: **a** $z_1[k]$, absence of $z[k]$, **b** $z_2[k]$, sinusoidal $z[k]$ with the same frequency used for evolution, **c** $z_3[k]$, sinusoidal $z[k]$ with double frequency than the one in **b**, **d** $z_4[k]$, sinusoidal $z[k]$ with half frequency than the one in **b**, **e** $z_5[k]$, triangular $z[k]$ and (f) $z_6[k]$, square $z[k]$. In red is the environment signal, $s[k]$, in blue is the non-controllable signal, $z[k]$, and in green is the output of the CTRNN, $x[k]$ (color figure online)

Table 5 Comparison of C_f with and without the evolved neural controller

Signal	z_{C_f}	s_{C_f}	$\frac{\Delta C_f}{s_{C_f}}(\%)$
$z_1[k]$	–	1	0
$z_2[k]$	1.63065	1.00083	62.9302
$z_3[k]$	1.63137	1.08802	49.9397
$z_4[k]$	1.63166	1.01013	61.5298
$z_5[k]$	1.32944	1.01495	30.9859
$z_6[k]$	2.27303	1.94682	16.7563

$$z_7[k] = A_1 \cdot \cos(14 \cdot f \cdot k + \varphi_1) + A_2 \cdot \cos(7 \cdot f \cdot k + \varphi_2) + A_3 \cdot \cos(f \cdot k + \varphi_3) + \mu \tag{30}$$

Hence, each sinusoid possesses a frequency in each component and the amplitude of $z_7[k]$ is between 1 and 0.5 since the consumption of a real grid is greater than 0. The application of the controller to this signal can be observed in Fig. 10. It can be observed that the result is as expected with the previous signals, the network is able to compute the derivative, change its sign and be in antiphase to $z_7[k]$ producing an almost constant $s[k]$. μ of $s[k]$ is in 1.004 and σ^2 of the signal is $1.84 \cdot 10^{-5}$, so $s[k]$ does not present much variation and is close to the μ. $s[k]$ is concentrated in the value of μ since the skew is low, $mu_3 = -8.34 \cdot 10^{-7}$ and $\mu_4 = 3.10 \cdot 10^{-6}$. The C_f of $z_7[k]$ is also reduced, $C_f(s[k]) = 1.0078$ and $C_f(z[k]) = 1.2703$, so the reduction of applying the neural network is a 26.05%. This factor is also closer to the ideal value of 1. Finally, the convergence time of the algorithm is 31 steps, again the CTRNN reaches quickly a stable state.

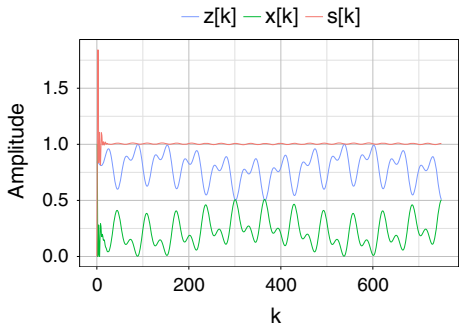

Fig. 10 Result of the post-evaluation for a $z[k]$ with a waveform similar to the grid. In red is the environment signal, $s[k]$, in blue is the non-controllable signal, $z[k]$, and in green is the output of the CTRNN, $x[k]$ (color figure online)

To sum up, the application of the evolved neural network always improves the behavior of the environment signal. Reducing the variability of the $z[k]$ and obtaining an antiphase signal from the CTRNN. In the case of signals with the same nature as the one evolved, the results are better than in other cases. For signals in which the derivative does not exist, the algorithm does not perform so well, but still it could also reduce the variability of the signal, converging quickly to a steady state.

7 Conclusions

In this paper, a neural network for the smoothness of continuous signals, with the electrical grid as an example, has been presented. A grid is composed by a series of facilities, which consume electricity and could have some source of local generation. Hence, they have been divided from the controllability point of view of its consumption in controllable and non-controllable facilities.

388,800 simulations were conducted to obtain the best neural architecture consisting of 1 input layer with 2 neurons, 1 hidden layer with 4 neurons and 1 output layer with 1 neuron corresponding to the output of the neural controller taking values for the free parameters inside the interval $[-5, 5]$.

A series of post-evaluation experiments have been done in which the first 4th central moments and the c_v were analyzed. Then, the C_f was evaluated to know the form of the resultant environment signal. These coefficients were tested with seven different signals which possess different characteristics. In all the cases, good results were achieved. In the case in which the derivate is not defined, it was able to reduce its variability but not as well as with continuous and class C^1 signals. With abrupt changes (triangular), the controller is able to produce a flattened $s[k]$; however with discontinuities (square), it is almost impossible and peaks are produced at the discontinuity point.

The evolved neural controller is the central block of the DSM controller. However, its direct application in the grid will create some problems to the network because users will try to consume the same power at the same time. In this case, greater variabilities of the aggregated consumption could be reached, contrary to the objective of the controller. Therefore, a coordination algorithm must be implemented in a future to produce collectively the antiphase consumption necessary to flatten the aggregated consumption.

Acknowledgements This work was partially supported by the "DEMS: Sistema Distribuido de Gestión de Energía en Redes Eléctricas Inteligentes", funded by the Programa Estatal de Investigación Desarrollo e Innovación orientada a los retos de la sociedad of the Spanish Ministerio de Economía y Competitividad (TEC2015-66126-R).

References

1. Akasiadis C, Chalkiadakis G (2017) Mechanism design for demand-side management. IEEE Intell Syst 32(1):24–31
2. Albadi M, El-Saadany E (2007) Demand response in electricity markets: an overview. In: 2007 power engineering society general meeting, IEEE, Piscataway, NJ, pp 1–5
3. Behrangrad M (2015) A review of demand side management business models in the electricity market. Renew Sustain Energy Rev 47:270–283
4. Bharathi C, Rekha D, Vijayakumar V (2017) Genetic algorithm based demand side management for smart grid. Wirel Personal Commun 93(2):481–502
5. Calpa M, Castillo-Cagigal M, Matallanas E, Caamao-Martín E (2016) Álvaro Gutiérrez: effects of large-scale PV self-consumption on the aggregated consumption. In: Proceedings of the 6th international conference on sustainable energy information technology (SEIT-2016), pp 816–823. Elsevier, Madrid, Spain
6. Canadian Electricity Association (2010) The smart grid: a pragmatic approach. Canadian Electronic Library, Ottawa
7. Castillo-Cagigal M (2014) A swarm intelligence approach based on coupled oscillators: an application in demand side management with photovoltaic distributed generation (2014)
8. Castillo-Cagigal M, Caamaño-Martín E, Matallanas E, Masa-Bote D, Gutiérrez A, Monasterio-Huelin F, Jiménez-Leube J (2011) Pv self-consumption optimization with storage and active dsm for the residential sector. Solar Energy 85(9):2338–2348
9. Castillo-Cagigal M, Gutiérrez A, Monasterio-Huelin F, Caamaño-Martín E, Masa-Bote D, Jiménez-Leube J (2011) A semi-distributed electric demand-side management system with PV generation for self-consumption enhancement. Energy Convers Manag 52(7):2659–2666
10. Castillo-Cagigal M, Matallanas E, Gutiérrez A, Monasterio-Huelin F, Caamaño-Martín E, Masa-Bote D, Jiménez-Leube J (2011) Heterogeneous collaborative sensor network for electrical management of an automated house with pv energy. Sensors 11(12):11544–11559
11. Castillo-Cagigal M, Matallanas E, Monasterio-Huelin F, Martín EC, Gutiérrez A (2016) Multifrequency-coupled oscillators for distributed multiagent coordination. IEEE Trans Ind Inform 12(3):941–951

12. Dillon TS (1991) Artificial neural network applications to power systems and their relationship to symbolic methods. Int J Electr Power Energy Syst 13(2):66–72
13. Edenhofer O, Pichs-Madruga R, Sokona Y, Seyboth K, Matschoss P, Kadner S, Zwickel T, Eickemeier P, Hansen G, Schlömer S, von Stechow C (2011) Special report on renewable energy sources and climate change mitigation. Tech. Rep. 1472, IPCC
14. Elliott SJ, Nelson PA (1993) Active noise control. IEEE Signal Process Mag 10(4):12–35
15. Elman JL (1990) Finding structure in time. Cognit Sci 14(2):179–211
16. Commission E (2006) European smartgrids technology platform: vision and strategy for Europe's electricity networks of the future. EUR-OP, Luxembourg
17. Gelazanskas L, Gamage KA (2014) Demand side management in smart grid: a review and proposals for future direction. Sustain Cities Soc 11:22–30
18. Gellings C (1985) The concept of demand-side management for electric utilities. Proc IEEE 73(10):1468–1470
19. Golberg DE (2006) Genetic algorithms in search. Optimization and machine learning. Addison Wesley, Indiana
20. Haque MT, Kashtiban AM (2007) Application of neural networks in power systems: a review. Int J Energy Power Eng 1(6):897–901
21. Haykin S (2009) Neural networks and learning machines. Prentice Hall/Pearson, New York
22. Kalra PK, Srivastava A, Chaturvedi DK (1992) Possible applications of neural nets to power system operation and control. Electr Power Syst Res 25(2):83–90
23. Kim JH, Shcherbakova A (2011) Common failures of demand response. Energy 36(2):873–880
24. Kurbatsky V, Sidorov D, Tomin N, Spiryaev V (2014) Optimal training of artificial neural networks to forecast power system state variables. Int J Energy Optim Eng 3(1):65–82
25. Li C, Yu X, Yu W, Chen G, Wang J (2017) Efficient computation for sparse load shifting in demand side management. IEEE Trans Smart Grid 8(1):250–261
26. Mandic DP, Chambers JA (2001) Recurrent neural networks for prediction: learning algorithms, architectures, and stability. Wiley, Chichester
27. Matallanas E, Castillo-Cagigal M, Gutiérrez A, Monasterio-Huelin F, Caamaño-Martín E, Masa D, Jiménez-Leube J (2012) Neural network controller for active demand-side management with PV energy in the residential sector. Appl Energy 91(1):90–97
28. Meireles MRG, Almeida PEM, Simoes MG (2003) A comprehensive review for industrial applicability of artificial neural networks. IEEE Trans Ind Electron 50(3):585–601
29. Mishra SK, Mishra S, Mishra G (2013) Realization of artificial neural network in power system and micro-grids: a review. Int J Eng Comput Sci 2(2):408–415
30. Palensky P, Dietrich D (2011) Demand side management: demand response, intelligent energy systems, and smart loads. IEEE Trans Ind Inf 7(3):381–388
31. Patel F, Prajapati HN (2015) A review on artificial neural network for power system fault detection. Indian J Res 4(1):52–54
32. Pearlmutter BA (1990) Dynamic recurrent neural networks. Tech. Rep. CMU-CS-90-196, School of Computer Science Carnegie Mellon University
33. Richter WD, Stelec L, Ahmadinezhad H, Stehlk M (2017) Geometric aspects of robust testing for normality and sphericity. Stoch Anal Appl 35(3):511–532
34. Strbac G (2008) Demand side management: benefits and challenges. Energy Policy 36(12):4419–4426
35. Torriti J, Hassan MG, Leach M (2010) Demand response experience in Europe: policies, programmes and implementation. Energy 35(4):1575–1583
36. UK Department of Energy & Climate Change (2009) Smarter grids: the opportunity. UK Department of Energy & Climate Change, London
37. U.S. Department of Energy (2009) Smart grid system report. U.S. Department of Energy, Washington, DC
38. Ye M, Hu G (2017) Game design and analysis for price-based demand response: an aggregate game approach. IEEE Trans Cybern 47(3):720–730
39. Yu CN, Mirowski P, Ho TK (2017) A sparse coding approach to household electricity demand forecasting in smart grids. IEEE Trans Smart Grid 8(2):738–748