Neural Networks 104 (2018) 15–25

Contents lists available at ScienceDirect

Neural Networks

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

Bio-inspired spiking neural network for nonlinear systems control

Javier Pérez *, Juan A. Cabrera, Juan J. Castillo, Juan M. Velasco

University of Malaga, C/ Ortiz Ramos s/n, 29071 Malaga, Spain

ARTICLE INFO

Article history:
Received 9 October 2017
Received in revised form 8 February 2018
Accepted 3 April 2018
Available online 12 April 2018

Keywords:
Spiking neural network
Supervised learning
Genetic algorithm
Nonlinear systems
Control

ABSTRACT

Spiking neural networks (SNN) are the third generation of artificial neural networks. SNN are the closest approximation to biological neural networks. SNNs make use of temporal spike trains to command inputs and outputs, allowing a faster and more complex computation. As demonstrated by biological organisms, they are a potentially good approach to designing controllers for highly nonlinear dynamic systems in which the performance of controllers developed by conventional techniques is not satisfactory or difficult to implement. SNN-based controllers exploit their ability for online learning and self-adaptation to evolve when transferred from simulations to the real world. SNN's inherent binary and temporary way of information codification facilitates their hardware implementation compared to analog neurons. Biological neural networks often require a lower number of neurons compared to other controllers based on artificial neural networks. In this work, these neuronal systems are imitated to perform the control of non-linear dynamic systems. For this purpose, a control structure based on spiking neural networks has been designed. Particular attention has been paid to optimizing the structure and size of the neural network. The proposed structure is able to control dynamic systems with a reduced number of neurons and connections. A supervised learning process using evolutionary algorithms has been carried out to perform controller training. The efficiency of the proposed network has been verified in two examples of dynamic systems control. Simulations show that the proposed control based on SNN exhibits superior performance compared to other approaches based on Neural Networks and SNNs.

1. Introduction

The use of systems based on Artificial Neural Networks (ANN) is increasingly spreading in the field of engineering. Their main applications are pattern recognition, data classification and prediction. A further use of ANN is the modeling and control of dynamic systems. Recently, research groups have begun to use ANN models inspired by the real behavior of biological neural networks. These types of networks have been called spiking neural networks (Gerstner & Kistler, 2002). This new approach offers a model closer to reality than previous generations of ANN.

The first generation of ANNs basically made use of McCulloch–Pitts threshold neurons, i.e. the neuron output only consisted of high or low levels. Second generation neurons use a continuous activation function to compute their output signals (a sigmoid function for example). The main difference between the 1st and 2nd generation of ANNs and SNN is the fact that the latter incorporate the concept of time into their operating model. In these networks, a spiking train between neurons, encodes and controls the system variables. Due to the time dependency of the variables, SNN is a suitable candidate for control of dynamic system (Meng, Wang, & Wang, 2017).

However, SNN has not been widely used in control schemes because of the complexity of the neuron model. The mathematical model of these neurons makes use of differential equations to obtain the state of the neuron at each time of the simulation. Therefore, a higher computational effort is required compared to ANN.

As a result, few works can be found in the literature in which SNN are applied to control system. For example, in Webb, Davies, and Lester (2011), the authors use spiking neurons to simulate a PID control. The authors claim that neurons approximate proportional, derivative and integral errors adequately. Despite that, they do not include a real application of control of a dynamic system using the PID neuron model. In Chadderdon, Neymotin, Kerr, and Lytton (2012) and Spüler, Nagel, and Rosenstiel (2015), the authors use a SNN to control an arm with 1 degree of freedom (DOF). To this purpose, the authors simulate the cortex, which is the region of our brain in charge of the control of voluntary movements, to control the arm. In Bouganis and Shanahan (2010), the authors present a SNN-based architecture to control a 4 DOF robotic arm. The structure is based on a large number of sensory neurons (1200 per signal) connected to the output or motor neurons (800 neurons per output). A similar application can be found in Carrillo, Ros,

* Corresponding author.
E-mail address: javierperez@uma.es (J. Pérez).

https://doi.org/10.1016/j.neunet.2018.04.002
0893-6080/

Boucheny, and Olivier (2008), where the authors aim to simulate the behavior of the cerebellum. As an application example, they use a SNN network to control a 2 DOF arm. Finally, another interesting work is a parallel architecture called SpiNNaker with more than one million neurons that can be used to control dynamic systems (Jin et al., 2010).

Contrary to classical control systems like PID or a fuzzy controller, in all these papers in which the control is carried out by means of a SNN, there is not a fixed control structure. In addition, they use complex neural networks structures with a high number of neurons and several layers. However, in nature there are examples of simple control with less than 10 neurons. In this case, a few input neurons are directly connected to other output neurons. For example, in Kandel (2001), the authors study the neural structure of a simple animal called Aplysia. They found that there are simple neural circuits to activate the gill. Furthermore, the connections between sensor and motor neurons are direct in these circuits. In Braitenberg (1984), a simple neural network is used to control vehicles. Vehicle behavior exhibits great dependence on the weights and type of connections between neurons. Another example of simple control neural networks can be found on the spinal cord, where a few neurons control the reflex acts of the human body (Balderas & Rojas, 2016), as happens with the patellar act. These neurons are completely isolated from the brain, performing a fast control action.

Evolution tends to reduce the size of neural networks that govern movements in living beings by optimizing their control function. Natural neuronal systems have been improving for millions of years which has enhanced their operation and reduced their energy demand. Nature shows that, in some cases, it is not necessary to increase the number of neurons to perform a complex control. SNNs take advantage of simulating natural systems to perform a faster and improved control compared to previous ANN approaches. Therefore, these systems are capable of carrying out the required control tasks with minimal resources and energy consumption.

Furthermore, the learning process is another important issue to be studied in this type of neural networks. Learning is a very important feature in biological neural networks since it allows a fast response to events that appear unexpectedly. Literature focuses on two types of learning processes: supervised and unsupervised. In unsupervised learning, biology-based learning rules are established, such as Spiking Timing Dependent Plasticity (STDP) (Froemke & Dan, 2002) or Bienenstock Cooper Munro (BCM) (Bienenstock, Cooper & Munro, 1982). These learning rules are based on the reinforcement or weakening of synaptic connections. On the contrary, in supervised learning, a set of training data consisting of pairs of input objects and the desired output values are required. The neural network must learn to predict the correct output for any valid input. There are several methods to perform this type of learning, such as Error Back Propagation (Bohtea, Koka, & La Poutr, 2002), Supervised Hebbain Learning (SHL) (Ruf & Schmitt, 1997), Remote Supervision (Ponulak & Kasiński, 2010) or the use of Evolutionary Algorithms (Belatreche, Maguire, McGinnity, & Wu, 2003). In this work, supervised learning based on genetic algorithms is employed. Despite a higher computational cost, the main advantage of using genetic algorithms in this application is that they can provide a nearly global optimal set of weights and neuron types in a reasonable time with simple programming.

Finally, the stability and robustness of the proposed control system, based on neural models of simple living beings and taught through supervised learning, must be verified. Two systems of different degrees of complexity, according to the number of differential equations that model them, have been chosen to test their performance. The first application is a model of a DC motor that includes the nonlinearities of the system. In the second example, the arm model described in Winters and Stark (1985) is controlled. In this case, arm movement is managed by different muscles coordinated by an activation signal (Chadderdon et al., 2012; Hulea & Caruntu, 2014).

The novelty of this work lies in the use of a reduced control structure based on spiking neural networks replicating biological control systems to control industrial applications. A further advantage of using a minimal number of neurons is that it enables knowing the function of each one of its components. This advantage makes it possible to adjust the parameters of the network (Arena, Fortuna, Frasca, & Patané, 2009) allowing to search for a concrete kind of action. The implementation in a real time control system is assured thanks to the simple execution code achieved and by the absence of filters and delays.

The remainder of the paper is organized as follows: Section 2 is devoted to the description of the neuron model. Next, the development of the control system based on SNN is included in Section 3. Section 4 presents the supervised learning process carried out using evolutionary algorithms. The performance of the whole system is verified through simulations included in Section 5. Finally, conclusions are drawn in Section 6.

2. Spiking neural network model

The main goal of this work is to implement a neural network with a reduced number of neurons in which the overall network performance can be known a priori. To do so, a description of the neuron model as well as connections between the neurons of the network, called synapses, is included next.

2.1. Neuron model

The choice of the neuron model is a subject of debate (Izhikevich, 2004). There are several models with different levels of complexity that reproduce biological neurons with lower or higher levels of accuracy. In this paper, the neuron presented by Izhikevich (2003) is utilized due to its good balance between computational cost and accuracy. Besides, its use is the most widespread within research groups in this area.

The Izhikevich neuron model is defined by two variables, u and v. Variable v is the potential of the membrane. It can be obtained from the following equation.

$$\frac{dv}{dt} = 0.94v^2 + 5v + 140 - u + I(t) \tag{1}$$

where I is the input current from the previous neuron synapses. Variable u is the recovery potential, which determines the response period. It can be obtained from Eq. (2).

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

where a and b are parameters that define the neuron type. c defines the reset potential of the membrane after a spike. d is parameter to be added to the recovery variable after the spike takes place.

$$\textit{If } v \geq 30 \textit{ then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d. \end{cases} \tag{3}$$

Parameters a, b, c and d fully define the spiking neuron model. This way, it is only necessary to describe the connections between them to have a complete neural network.

2.2. Synapse model

Synapses play a highly significant role in the neural network. They control the spike flow between neurons as well as the effect that they produce in the post-synaptic neuron. This way, an exciting action in the synapse can cause a fast trigger in the neuron, or,

on the contrary, an inhibitory action can reduce the firing rate of the neuron.

The behavior of the synapse is modeled based on the time between fires. This model generates an output current $I(t)$ that will decrease exponentially as time increases from the last spike, according to the following equations.

$$I(t) = w_{ij}\varepsilon(t - t_{spike}) \tag{4}$$

$$\varepsilon(t - t_{spike}) \begin{cases} \dfrac{t - t_{spike}}{\tau} e^{1-\frac{t-t_{spike}}{\tau}}, & if\ t \geq t_{spike} \\ 0, & if\ t < t_{spike} \end{cases} \tag{5}$$

where w_{ij} is a weight that defines the behavior of the synapse, making the synapse inhibitory or excitatory depending on the neural model. t_{spike} is the firing time of the pre-synaptic neuron, τ is the decay time of the synapse output current.

3. Dynamic system control using spiking neural networks

In this section, the developed spiking neural network to be used in control systems is described. A classical feedback control scheme is used. An input coding is required in order to translate the control variable into spikes. The lay of neurons processes the control variable and generates an output consisting of a train of spikes. Finally, this output is decoded, providing the input to the plant (Fig. 1).

As mentioned previously, bio-inspired structures to define input coding, neural structure, and output coding are used in this work. They are described in the following sub-sections.

3.1. Input coding

The coding and decoding of the input and output variables, respectively, in the SNN is an aspect of great importance. In SNN, unlike in previous neural networks, it is possible to use temporal coding. For example, in Clawson Ferrari, Fuller, and Wood (2016), the authors use a so-called 'Population coding" for the input variables in which they are transformed into spike-times by using a number of input neurons. In particular, they use multiple local receptive fields to distribute an input continuous variable over multiple input neurons. Although the spikes are discrete, the neurons do not work with discrete values but with continuous input currents coming from the synapse. Likewise, the output is a continuous voltage corresponding to the potential of the membrane. Therefore, synapses are the component of the neural network that generate the DC current that attacks the neurons. Synapses behave similarly to a first-order system, i.e. they filter the signal, add a delay and pass from quasi-discrete spikes to a continuous value (Eq. (4)).

In this work, the coding of the input is performed by a fuzzification-alike process using gauss distributions (Bohte, La Poutré, & Kok, 2002). To do so, a number of Gaussian distributions that coincides with the number of neurons in the first layer (m) are distributed along the entire range of the input variable. The means of each of the distributions (σ_i) are obtained from Eq. (6). So defined, the outputs are bounded between zero and the maximum intensity value. The distributions are defined according to Eqs. (7) and (8):

$$\sigma(i) = \frac{n_{\max} - n_{\min}}{\beta(m-2)} \quad i = 1, \ldots, m \tag{6}$$

$$\varphi(i) = n_{\min} + (i - 1.5) \cdot \sigma(i) \quad i = 1, \ldots, m \tag{7}$$

$$I_m(n, i) = I_{\max} \cdot e^{-\frac{(n-\varphi(i))^2}{2\cdot\sigma(i)^2}} \quad i = 1, \ldots, m \tag{8}$$

where $I_m(n)$ is the input current to each input neuron, n is the input variable, i.e, the error, m is the number of input neurons, $n_{\min}$ and $n_{\max}$ are the limits of the input variables, $\beta = 1.5$ is a shape factor and $I_{\max}$ is the maximum value of the output current, which is defined by the programmer.

Therefore, all input neurons are activated by the input currents. However, the level of excitation of each neuron depends on how close the input value is to the mean of the distribution associated to the neuron. The main reason to use this type of coding is that no delay is introduced, which is crucial for control in real time applications.

An example of the coding of the input variable is included next. In this case, the input neurons used are of the type called 'regular spiking'. This type of neuron provides a spikes train of frequency proportional to the input current to the neuron. Three input neurons are used (N, Z and P). If the value of the input is low, close to zero or positive, the N, Z and P neurons are more excited, respectively. As can be seen in Fig. 2a, for a given input of a value equal to $0.2{\cdot}n_{\min}$, the input current to neurons N, Z and P are $I_{\max}$, $0.8{\cdot}I_{\max}$ and $0.4{\cdot}I_{\max}$. As mentioned before, the spike frequency is proportional to the input currents to each neuron, as it can be seen in Fig. 2b.

3.2. Neural network structure

The number of neurons and the way they are distributed is described next. Particular attention has been paid to the structure of the neural network as a key factor in its overall behavior. Some researchers create variable structures. This way, the number and connections between neurons have to be optimized for each particular application (Oniz & Kaynak, 2014). On the contrary, another solution in the literature is the use of a fixed structure with a high number of neurons (Oniz, Aras, & Kaynak, 2013). In our approach, our aim is to recreate simple structures found in living beings, where a low number of neurons is employed. These simple structures can be encountered in the Aplysia and the reflex acts in humans.

One of the most analyzed living being is the Aplysia, a slug with two types of circuits between neurons (Kandel, 2001) (see Fig. 3a). The first circuit, called modulator, connects interneuronal signals from other sensors with the control circuit, called the mediator. The mediator circuit performs the control functions and connects siphon sensory neurons with gill motor neurons. The mediator circuit works as follows. The sensorial neurons (SN) are activated by the external inputs that reach the living being, e.g., a tactile stimulus. This activation causes the SN to send an action potential to the motor neurons (MN). The modulator circuit also receives inputs from the tail sensors. It is important to note that these are not used directly to control the contraction of the gill, but to manage the learning process and provide an indirect control over motor neurons. Finally, MN control muscles by managing their elongation, in this case, the closing or opening of the gill. In this work, our control structure resembles the mediator circuit described above, with a series of neurons in the input (SN) and output (MN) stages. The modulator circuit is not required in our approach because this paper is focused on fast control, which makes the conditional learning process provided by the modulator circuit unnecessary.

As stated before, simple control structures control reflex acts in humans. As an example, a well-known reflex act, the patellar act, takes place when knocking on the patellar tendon. If so, a neural network activates the muscles of the leg and makes it lift. This control process is made without involving the brain, which makes the response of the body faster. This type of behavior is optimal for control applications. Like in the Aplysia example, there are sensory neurons, placed on the patellar tendon, and motor neurons that shrink and extend the muscles of the leg. In this particular case, motor neurons shrink the quadriceps and extend the hamstring muscle (see Fig. 3b). The neural network found

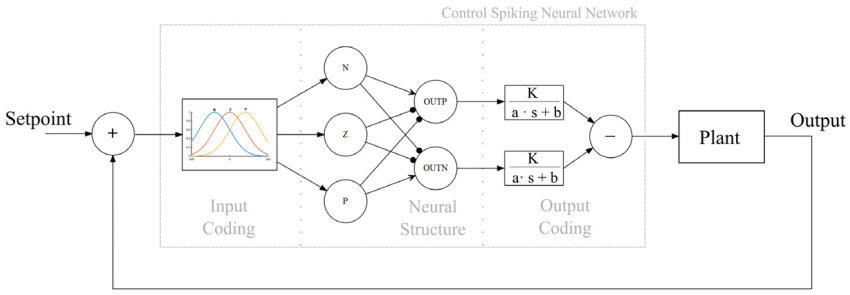

Fig. 1. Feedback control based on spiking neural networks divided in input coding, neural structure, and output coding.

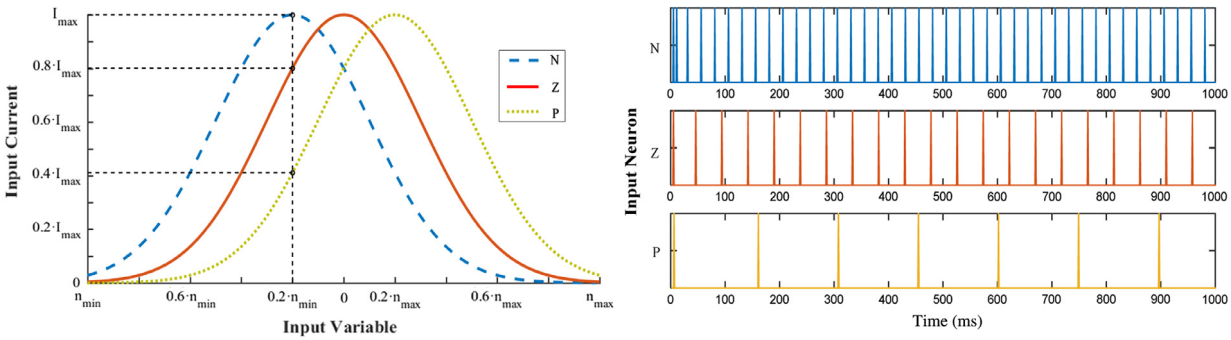

Fig. 2. (a) Coding for a constant current input. (b) Excitation level of the input neurons.

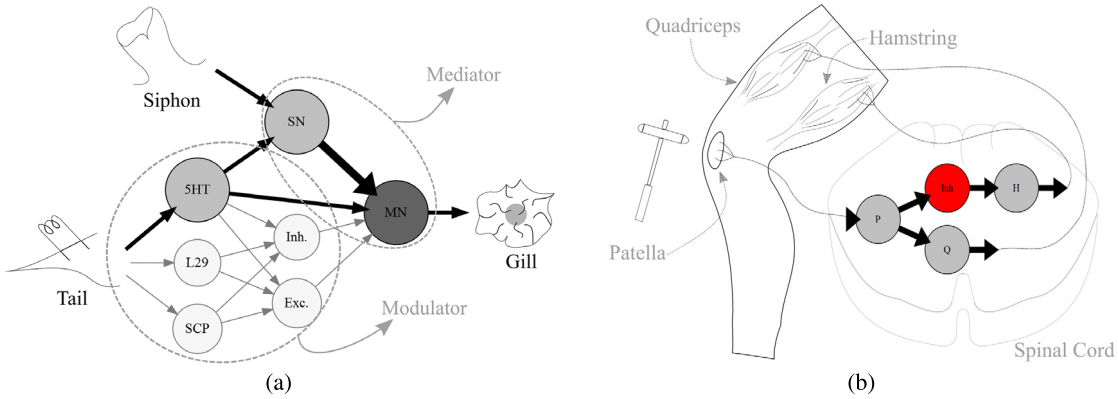

Fig. 3. (a) Structure of the Aplysia interneuronal system. (b) Patellar reflex interneuronal system placed in the spinal cord.

in the spinal cord oversees the control. It is based on one direct connection between the sensors and the quadriceps motor neuron. Besides, an interneuronal connection between the sensor and the hamstring inverts the signal due to its inhibitory behavior, Fig. 3b. This way, when the patellar tendon is knocked, the sensor neuron (P) activates the quadriceps (Q) and deactivates the hamstring (H) by means of the inhibitory neuron (Inh). As a result, the leg is lifted.

A generic control structure based on these simple neural networks is designed to control dynamic systems. The main objective is to maintain the error, i.e. the difference between the actual point of the system and the setpoint, next to the zero value. To do so, if the error is positive the control signal is decreased and if the error is negative the control signal is increased. Like in the nature example, in the proposed structure, the neurons are distributed to recreate this behavior. The network is composed of three input neurons, which are activated according to the error value, and two output neurons, that act as antagonist muscles by changing the control variable (Fig. 4). OUTP output causes an increase in the control signal while OUTN decreases it. Fig. 4 shows this behavior when negative, zero and positive neurons are activated.

3.3. Output coding

The decoding is the process in which the spike-train coming from the output neurons is converted into a continuous variable. An example of this type of decoding is shown in Kaiser et al. (2016), where a steering wheel decoder based on an agonist–antagonist muscle mode is used. In this work, a first-order system is used to perform the decoding (Eq. (9)). The decoding forces the neuron outputs to be within the available range of the values of the control action.

$$Muscle(s) = \frac{K}{a \cdot s + b} \tag{9}$$

where $K = 200$, $a = 1$ and $b = 5$. Fig. 5 shows the output of the control system for a sinusoidal setpoint. The first and second plots

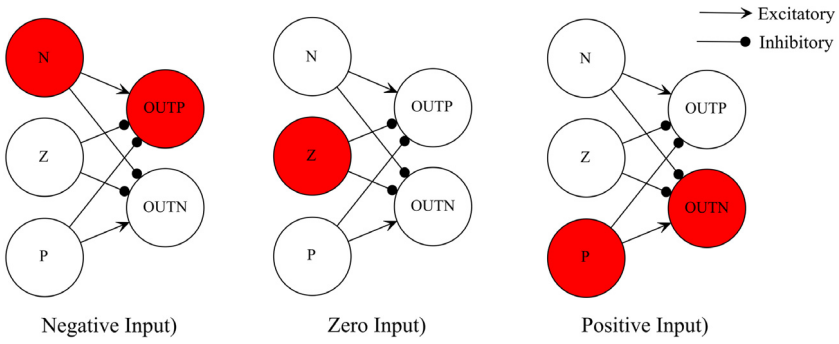

Fig. 4. Main structure of the neural network representing the stages of activation with different input levels.

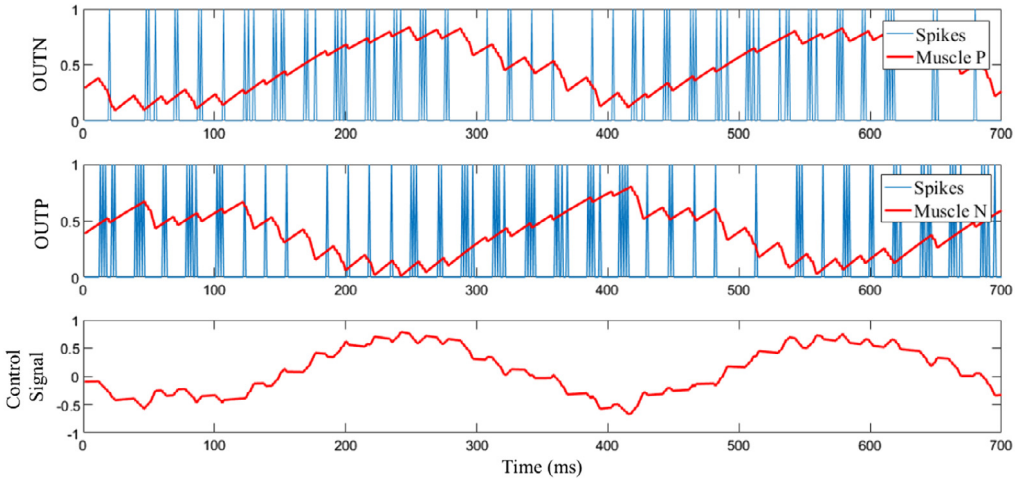

Fig. 5. OUTP and OUTN represent the triggering of the two output neurons and the action result of the muscles after the first order system. Control signal is the input of the plant as the result of the agonist–antagonist response of OUTP and OUTN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(OUTP and OUTN) include the spiking neuron outputs (in blue) and the result after the decoding (Muscle P and Muscle N, in red). Thus, the spikes from the motor neurons excite the muscles by means of its decodification. The control signal is obtained by adding both muscles outputs. Focusing on the output of the neurons, the agonist and antagonist behavior of the neurons can be observed as both signals tend to be opposed.

4. Supervised learning based on genetic algorithm

As it was shown in Fig. 4, the structure of the SNN is composed of three input neurons, two output neurons and the synapses that connect them. All input neurons are of the same type. Similarly, all output neurons are also of the same type. However, input and output neurons are of different types. This way, these structures reproduce natural systems, where sensor neurons and motor neurons are different types.

Once the structure of the neural network has been defined, a learning process is required to optimize the network's performance. The neural network is defined by a set of parameters that regulates its response to the inputs. These parameters describe the type of neuron, both for the input and the output set. For each set, a total of 4 parameters (a, b, c and d in Eqs. (1)–(3)). In addition, the synapse weights are also optimized. Three learning processes have been carried out: (1) synapse weights optimization, (2) neuron types optimization and (3) synapse weights and neuron types optimization.

A supervised learning system is used to find the optimal value of the network parameters. The learning process consists of updating the parameters that define the network by comparing the output of the network with a desired output. To do so, an optimization process is required to find the set of parameters that produces the closest solution to the desired one (Fig. 6). Therefore, the optimization has to minimize an error function. In this work, this function is the sum of the absolute errors (e) through the entire simulation, called IAE.

$$IAE = \sum_{t=0}^{t=t_{max}} |e(t)| \,. \tag{10}$$

4.1. Optimization procedure

The optimization process has been carried out using an algorithm based on differential evolution. The algorithm was developed by this research group (Cabrera, Castillo, Carabias, & Ortiz, 2015). The main advantage of this algorithm is that it can easily be modified to adapt it to this problem. Besides, its programming, using parallel execution, makes it quite appropriate for this application.

As it can be seen in Fig. 7, the optimization begins with the creation of a random initial population of NP individuals (step 1). Each individual (or chromosome X) is a set of parameters whose values are randomly generated within the searching space, each set being a possible solution to the problem. New population are created by evaluating the objective function (step 2) (Eq. (10)), selecting couples of individuals (step 3) and applying reproduction and mutation operators iteratively (step 4). The third step is individual

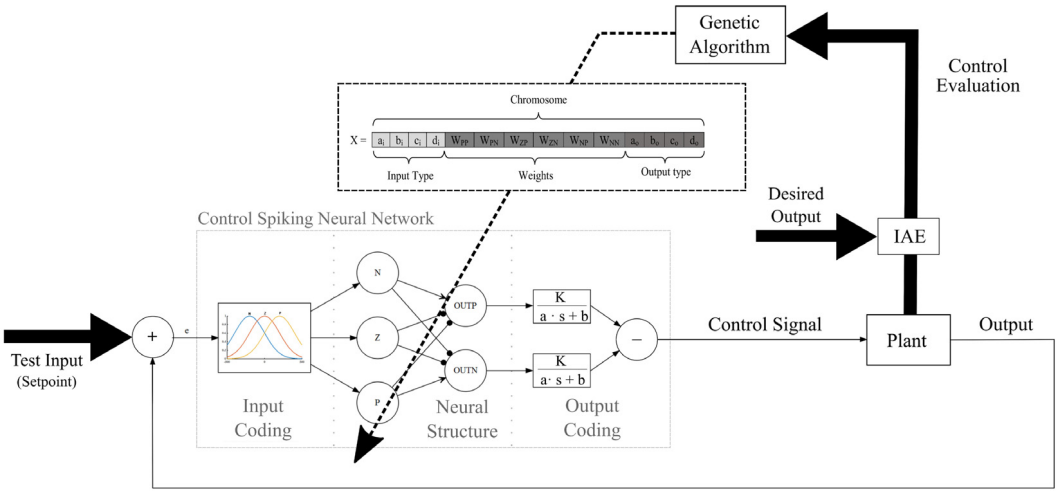

Fig. 6. Scheme of the optimization process using genetic algorithms.

Table 1
Neural weight initialization.

Symbol	Description	Random value range
W_{PP}	Weight (In Positive–Out Positive)	[0,40]
W_{PN}	Weight (In Positive–Out Negative)	[−20,0]
W_{ZP}	Weight (In Zero–Out Positive)	[−20,0]
W_{ZN}	Weight (In Zero–Out Negative)	[−20,0]
W_{NP}	Weight (In Negative–Out Positive)	[−20,0]
W_{NN}	Weight (In Negative–Out Negative)	[0,40]

selection for reproduction. In this work we use a scheme known as differential evolution (Storn & Price, 1997). In this procedure, the selection of individuals for reproduction is carried out as follows: one of the parents is the *i* individual of the current population (X_i). To generate the other parent, called *V* parent, the best member of the current population (X_{best}) and two randomly chosen members (X_{r1} and X_{r2}) are selected and combined according to Eq. (11), where *F* measures the deviation from the best member.

$$V = X_{best} + F(X_{r1} - X_{r2}). \tag{11}$$

The fourth step in the algorithm is crossover and mutation. In the crossover operator, the two parents (X_i and V) chosen for reproduction generate a new offspring (X_i^{New}). In this process, genes of parents X_i and V are interchanged with a *CP* probability. Simultaneously with reproduction, mutation is carried out with probability *MP*. Mutation is an operator consisting of a random change (*range*) of a gene during reproduction. The optimization process ends when a predefined maximum number of iterations is reached.

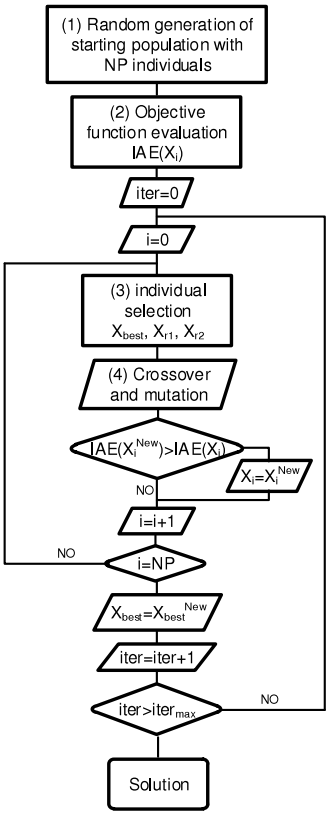

Fig. 7. Block diagram of the genetic algorithm.

4.1.1. Weight optimization

The optimization of the synapse weights requires a chromosome of 6 parameters. The genes are the weight of the connections between the 3 input neurons and the two output neurons. The initial range of each gene is defined according to its excitatory or inhibitory behavior (Table 1), as it was defined in the structure of the network (Fig. 4).

4.1.2. Type optimization

A chromosome of 8 parameters is needed when optimizing the neuron type. In this case, the genes are the parameters that describe the input and output neurons, respectively. Regular Spiking, Chattering, Fast Spiking, Thalamo-Cortial and other neuron types are characterized depending on the neuron parameters. The initiation ranges have been established with the help of the Izhikevich model (Izhikevich, 2003).

Finally, a chromosome of 14 parameters is required when optimizing the neuron types and the synapses weights. The range of each gene is the same as those established Tables 1 and 2.

4.2. Training

A sinusoidal signal with a variable amplitude and frequency that includes constant value periods is used to perform the learning

Table 2
Neural type initialization.

Symbol	Description	Random value range
a_i	Input neuron type	[0.02,0.1]
b_i	Input neuron type	[0.2,0.25]
c_i	Input neuron reset	[−65,−50]
d_i	Input neuron recovery	[0.05,8]

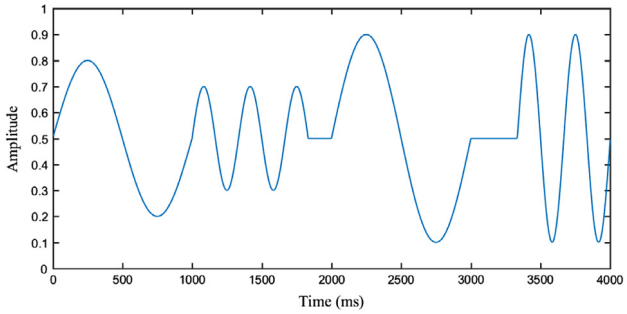

Fig. 8. Training signal based on a variable sine in frequency and amplitude with constant areas.

process of the neural network, Fig. 8, Webb et al. (2011). The advantage of using such a training signal is that a good frequency and amplitude response is achieved because the system is exposed to different frequencies and amplitudes. Besides, the addition of constant zones improves the output of the system when a steady input is applied by diminishing the response oscillations.

5. Simulations and results

The performance of the control system based on SNN has been evaluated by simulations with two different applications. The first one is the input current control of a DC motor to achieve a desired output torque. Next, a non-linear model that reproduces the human arm is controlled.

5.1. DC motor controller

Two differential equations are used to model the behavior of a DC motor. The first one models the mechanical part (Eq. (12)) while the second one reproduces the electrical part of the motor (Eq. (13)).

$$J\frac{d^2\theta}{dt^2} = iK_t - b\frac{d\theta}{dt} \tag{12}$$

$$L\frac{di}{dt} = -Ri + V - K_e\frac{d\theta}{dt}. \tag{13}$$

A non-linearity is introduced by means of an input saturation voltage of ±10 V and a dead zone of ±1.2 V. The control variable is the input current to the rotor. The output torque of the motor can be obtained by multiplying the input current by the torque constant. Thus, the output torque is controlled if the input current is regulated. The parameters of the genetic algorithm were the following: $NP = 100$, number of iterations (iter) $= 100$, $CP = 0.4$, $MP = 0.2$, range $= 0.5$, $F = 0.6$. Where the parameters and the values used in the simulations are detailed in Table 3.

5.1.1. Results (optimization comparison)

As stated in Section 4, there are 3 possible optimization processes: synapse weights, neuron types or both optimizations. Simulations have been conducted in order to compare the three possibilities. As shown in Table 4, the higher the number of variables to be optimized, the lower the sum of the absolute errors obtained. This is due to the fact that more parameters of the neural network are available to be tuned by the optimization algorithm, thus exploring a higher number of potential solutions to the problem. Therefore, solutions with a lower IAE can be found, which means that the controller can provide a more accurate output. However, the increase of chromosome size leads to a higher computational cost for the same number of iterations. Despite the fact that the synapse weights and neuron types optimization requires a higher computational effort, this optimization is selected due to the better performance of the network. The algorithm was run using a standard PC with a 3.07 GHz Intel (R) Core TM 950 processor.

Table 3
DC motor parameters.

Parameters	Notation	Units	Value
Moment of inertia of the rotor	J	kg m^2	0.01
Motor friction constant	b	N s m	0.1
Torque constant	Kt	N m/A	0.01
Electromotive force constant	Ke	V/s	0.01
Winding resistance	R	Ω	1
Inductance	L	H	0.5

Fig. 9 shows the response of the controlled system when the inputs are the learning signal (a) and the step input (b). In this application example, the best system response is achieved when the optimization of the neural network includes the parameters that define the neuron type and the synapse weights, as summarized in Table 4. Weight and type optimizations provide the lowest IAE with the best compromise between overshoot and rise time. On the contrary, weight optimization yields the controller that makes the system have the highest rise time. This can cause delays in system response as seen in the system output to the training signal (Fig. 9a). Therefore, the system requires more time to reach output values close to the desired values. However, it has less overshoot producing a smoother response. Finally, type optimization makes the system have the shortest rise time but the highest overshoot. Besides, output oscillations are observed.

The genetic algorithm has been programmed in MATLAB©. SIMULINK© has been used to model the complete controlled system. After the optimization, the values shown in Table 5 were obtained. A robust and easy-to-implement control is achieved with only 5 neurons and 6 synapses.

5.1.2. Comparison

In order to verify the performance of the proposed control based on SNN, a comparative study between a nonlinear autoregressive moving average neural network controller (NARMA-L2) (Valluru & Singh, 2016) and the best solution found in the previous section has been carried out. The NARMA-L2 controller transforms nonlinear system dynamics into linear dynamics by canceling the nonlinearities. There are two basic steps in the NARMA-L2 controller design: system identification and controller design. In the first step, the nonlinear autoregressive moving average (NARMA) model is used to identify the system. Next, the controller is designed using the identified model of the plant.

Similar parameters to the ones used in Spüler et al. (2015) were used for the design of the controller. First, the system was identified with a multi-layer neural network architecture. The number of delayed plant inputs and outputs were set at 1. Unlike in the previously mentioned reference, 5 neurons were used in the hidden layer in this comparison, similar to the number of neurons used in the SNN controller. The sampling interval was 0.01 s. Second, in the controller design stage, a gradient descent with a momentum and adaptive learning rate backpropagation algorithm was used to train the controller. The rest of parameters for the training were: 1000 training samples, maximum and minimum

Table 4
Optimization comparative of the integral of the absolute error value.

Optimization	Variables optimized	IAE (Training signal)	Rise time	Overshoot	Mean time per iteration
Weight	6	0.3507	94 ms	2.4%	1.1149 s
Type	8	0.2209	63 ms	31.9%	1.1434 s
Weight + Type	14	0.1701	70 ms	14.3%	1.2351 s

Table 5
Neural network configuration.

Symbol	Description	Weight	Type	Weight + Type
a_i	Input neuron type	0.02[a]	0.0251	0.0359
b_i	Input neuron type	0.2[a]	0.0705	0.0775
c_i	Input neuron reset	−65[a]	−60.7982	−58.9570
d_i	Input neuron recovery	8[a]	3.3497	2.6539
a_o	Output neuron type	0.02[a]	0.0464	0.3983
b_o	Output neuron type	0.2[a]	−1.1142	−0.2492
c_o	Output neuron reset	−65[a]	−43.6111	−96.1849
d_o	Output neuron recovery	8[a]	15.1540	39.3966
W_{PP}	Weight (In Positive–Out Positive)	130	20[a]	691.8236
W_{PN}	Weight (In Positive–Out Negative)	−22.3	−10[a]	−17.8650
W_{ZP}	Weight (In Zero–Out Positive)	20.07	−10[a]	−29.4566
W_{ZN}	Weight (In Zero–Out Negative)	24.91	−10[a]	19.8571
W_{NP}	Weight (In Negative–Out Positive)	2.32	−10[a]	−56.5727
W_{NN}	Weight (In Negative–Out Negative)	105.51	20[a]	551.5980

[a] Fixed for that optimization.

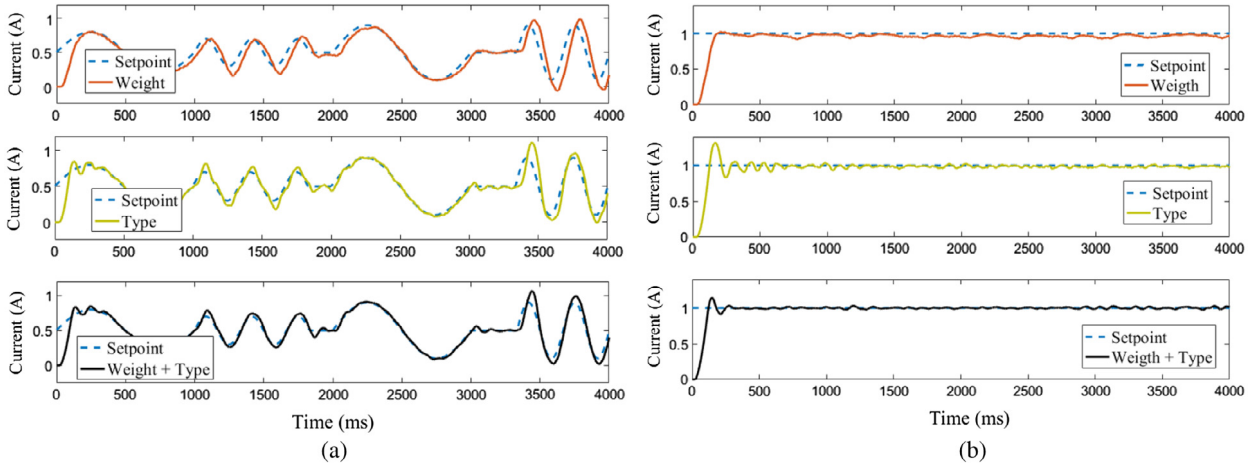

Fig. 9. Setpoint comparison between different optimization method learning processes by genetic algorithm. (a) Training signal response (b) Step response.

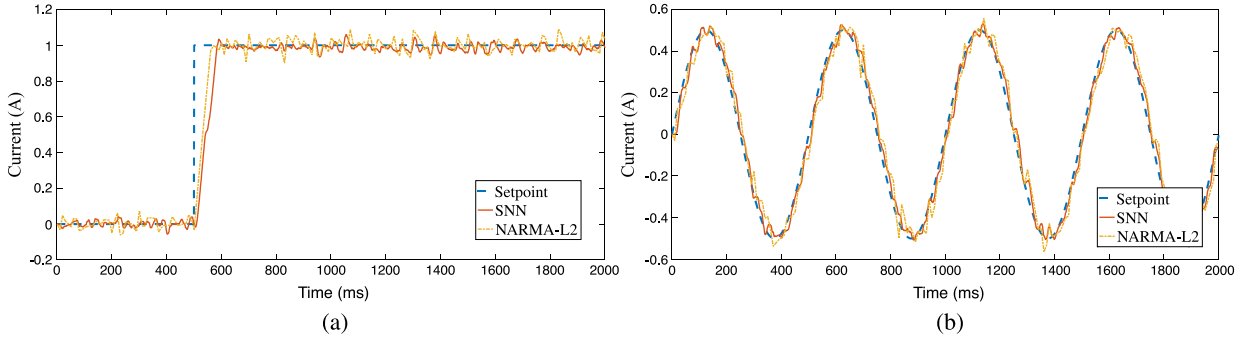

Fig. 10. (a) Step response comparison between NARMA-L2 and SNN. (b) Sine response comparison between NARMA-L2 and SNN.

plant inputs were set at +10 and −10, respectively, and maximum and minimum interval values were 0.0001 and 0.00001 s.

A step and a sinusoidal input to the electric motor were used to perform the comparisons. Zero mean Gaussian noise with variance 0.0015 was added to the current output sensor used to simulate real conditions. As Fig. 10 shows, the SNN based controller provided significantly better results than the NARMA-L2 controller when the input was a sinusoidal signal. However, the response to the step input yielded a higher IAE (Table 6). This was due to a slightly slower response. Nevertheless, the output after the system

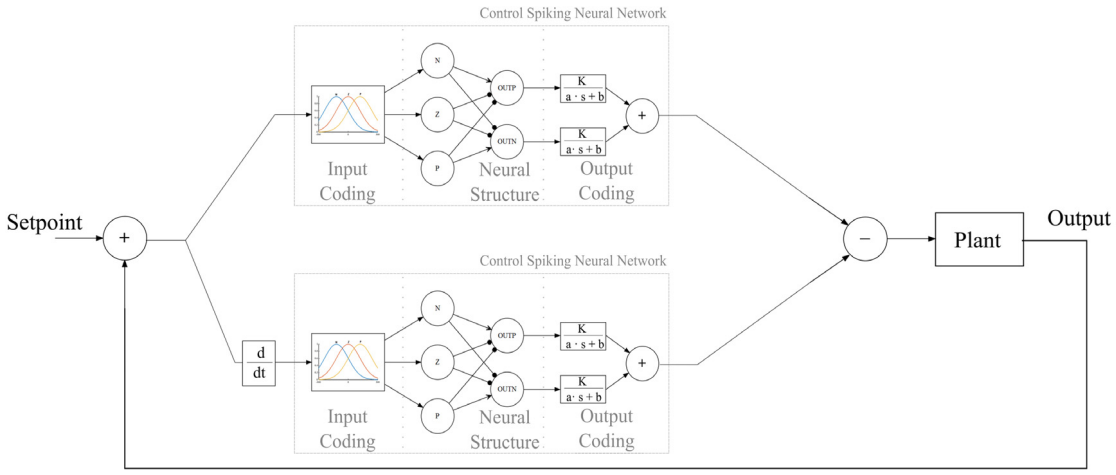

Fig. 11. Control structure based on two control SNN with input for error and error difference.

Table 6
Comparison between NARMA-L2 and SNN.

Controller	Rise time	IAE (Step)	IAE (Sine)
SNN	61 ms	0.07852	0.06426
NARMA-L2	50 ms	0.07429	0.09719

reaching values close to the desired setpoint was less affected by the noise compared to the NARMA controller.

5.2. Arm movement control

A further example of the use of a complex system control based on SNN is included in this section. A non-linear model of an arm reproducing the agonist–antagonist behavior of natural muscles in the elbow is used to evaluate the performance of the proposed controller. The model of the muscles is based on the one introduced by Hill (1938). In this example, the muscles are modeled using elastic and contractile elements. To do so, passive and an active torques are applied to control the elbow joint movement. Thus, the torques generated by the muscles can be modeled using the following equations.

- A passive torque, which models the elastic behavior of the muscle. This behavior is reproduced by means of a spring–damping system. This torque is not externally controlled by any activation signal. This passive torque is obtained from the following equation:

$$M_p(\theta) = k_0\theta + b_0\frac{d\theta}{dt} + k_1(e^{k_2\theta} - 1) \tag{14}$$

  where k_0, b_0, k_1 and k_2 are defined in Table 7.
- An Active torque, which models the contractile behavior of the muscle. As opposed to passive torque, this torque is regulated by an external trigger signal from motor neurons. Therefore, the equation that models this torque:

$$M_a(\theta) = Act \cdot \left(e^{-(\frac{\theta - MXoo}{MXsh})^2} + MXsl \cdot \theta\right) \cdot M_{\max} \tag{15}$$

  where $MXoo$, $MXsh$, $MXsl$ and M_{max} are defined in Table 8.

The control system regulates the activation signal (Act) to make the arm follow the setpoint positions. The activation signal is bounded between -1 and 1. The absolute value indicates the degree of activation. The direction of movement is represented with the sign. A positive sign means a lifting movement while a negative signs reproduces the arm extension.

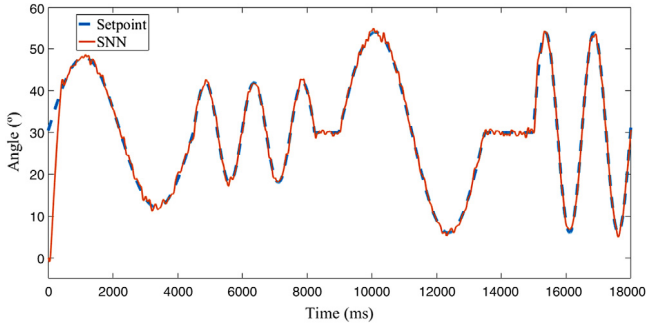

Fig. 12. Angular position of the arm. Comparison between the training signal and the output obtained by the SNN network.

Table 7
Passive values.

Symbol	Description	Value flexion	Value extension
k_0	Spring constant	3 N/m	3 N/m
b_0	Damping constant	0.3 N s/m	0.3 N s/m
k_1	Stiff-spring constant	−0.0074 Nm	0.0023 Nm
k_2	Stiff-spring exp. constant	$-5.7296\ \text{deg}^{-1}$	$6.7407\ \text{deg}^{-1}$

Finally, the equation that models the behavior is the following:

$$J_b\frac{d^2\theta}{dt^2} = V\left(\dot{\theta}\right) \cdot M_a(\theta) - M_p(\theta) - mgl\cos\theta. \tag{16}$$

As proposed by Hill, active torque $M_a(\theta)$ is multiplied by factor V that depends on the arm angular speed. In general, this factor decreases the active torque as the speed increases. Where J_b ($0.0704\ \text{kg m}^2$) is the inertia moment of the arm, m (1.65 kg) is the arm mass and l (0.179 m) is the distance from the center of gravity to the joint.

In this case, a single neural network does not perform an accurate and robust control due to the complexity of this system. Therefore, two neural networks are required to control arm movement. The first network is devoted to process the error. The error variation is the input to the second neural network. This network provides a satisfactory performance without increasing significantly the complexity of the control system. The final structure is shown in Fig. 11.

5.2.1. Results

Simulations have been carried out with the proposed network structure. Both, neural types and synapses weights have been

Table 8
Active values.

Symbol	Description	Value flexion	Value extension
MXoo	Angle at maximum moment	0°	0°
MXsh	Gaussian-type "shape" function	1.65	1.65
MXsl	"slope" parameter	0.1	0.1
M_{max}	Maximum moment	60 N m	50 N m

Table 9
Neural network configuration.

Symbol	Description	Value	Symbol	Description	Value
a_i	Input neuron type	0.9223	a_{di}	Input derivate neuron type	0.0295
b_i	Input neuron type	0.0036	b_{di}	Input derivate neuron type	0.1644
c_i	Input neuron reset	−67.8297	c_{di}	Input derivate neuron reset	−64.2742
d_i	Input neuron recovery	9.8868	d_{di}	Input derivate neuron recovery	4.4094
a_o	Output neuron type	0.1160	a_{do}	Output derivate neuron type	0.1226
b_o	Output neuron type	1.0064	b_{do}	Output derivate neuron type	0.5458
c_o	Output neuron reset	−48.8686	c_{do}	Output derivate neuron reset	−57.9161
d_o	Output neuron recovery	5.3373	d_{do}	Output derivate neuron recovery	−1.9128
W_{PP}	Weight (In Positive–Out Positive)	28.5117	Wd_{PP}	Weight derivate (In Positive–Out Positive)	54.1964
W_{PN}	Weight (In Positive–Out Negative)	−31.5076	Wd_{PN}	Weight derivate (In Positive–Out Negative)	−59.3492
W_{ZP}	Weight (In Zero–Out Positive)	−33.5738	Wd_{ZP}	Weight derivate (In Zero–Out Positive)	−5.3129
W_{ZN}	Weight (In Zero–Out Negative)	12.2756	Wd_{ZN}	Weight derivate (In Zero–Out Negative)	−20.5018
W_{NP}	Weight (In Negative–Out Positive)	−8.2379	Wd_{NP}	Weight derivate (In Negative–Out Positive)	−65.7671
W_{NN}	Weight (In Negative–Out Negative)	32.1006	Wd_{NN}	Weight derivate (In Negative–Out Negative)	39.9408

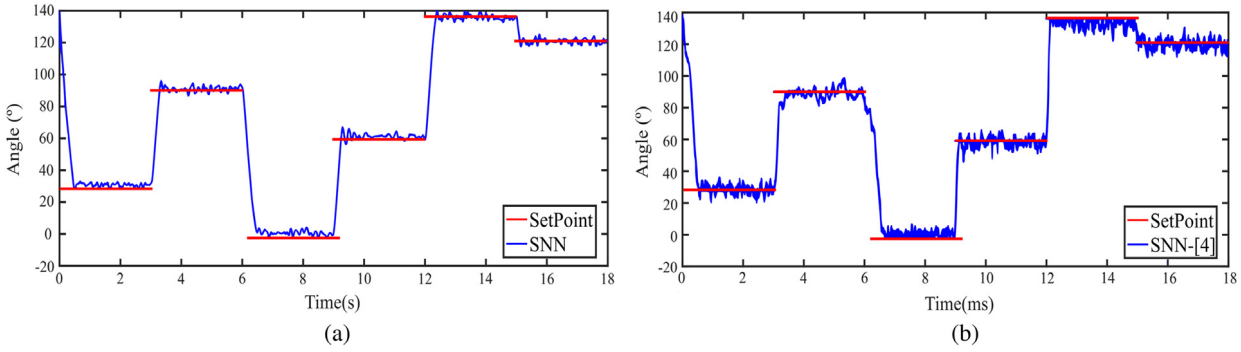

Fig. 13. Comparative between our reduced SNN (a) and the SNN from Spüler et al. (2015) (b) for the same setpoint.

optimized. Thus, a total of 28 variables have been optimized, giving rise to the values in Table 9.

In the first simulation, the input to the arm is the signal used in the learning process. The system response can be observed in Fig. 12. The initial delay is due to the difference between the initial arm position and the initial value of the reference signal. After that, no delay is observed between both signals. Some noise is still observed in the system response due to the decoding process, as mentioned in previous simulations.

5.2.2. Comparison

The effectiveness of the proposed SNN-based control is illustrated in comparison with the control described in Spüler et al. (2015). These authors use a higher number of neurons and a more complex architecture. The output of the system can be observed in Fig. 13a. Similarly, Fig. 13b includes the response of the system with the control proposed in this paper. Some oscillations can be observed in the controlled systems. These oscillations are due to inherent spike behavior of the neural network introduced by the decoding process. As it can be seen, similar rise time and settling times are obtained in both cases. However, system stability is clearly improved with the new proposal. It has to be mentioned that the control is carried out with only 10 neurons while 304 neurons are used by Spüler et al. (2015). Furthermore, Spüler et al. (2015) used the same signal for training and simulations while our system is trained with the signal shown in Fig. 8. This indicates a good ability of our proposal to adapt to non-learned conditions.

The hardware capabilities of both proposals are evaluated next to ensure that the time response comparison can be performed. In the first case, 1000 ms of simulation takes 987 ms of computer time while in our approach it takes 291 ms. The lower simulation time is due to the reduced number of neurons used, which implies fewer mathematical operations. Therefore, both systems can be run in real time systems with a loop time of 1 ms and, thus, the time response comparison can be performed.

6. Conclusions

A spiking neural network (SNN) based on biological systems for control purposes has been described in this work. Input coding imitating the sensory neurons of biological networks has been developed. The encoding of the input is performed by a fuzzification-alike process using gauss distributions. In addition, output coding reproducing the agonist–antagonist behavior of muscles has been proposed. This way, neuron output has been decoded by a first order system that adapts its value to be within the available range of the control action signal values.

Supervised learning based on evolutionary algorithms has been developed which has allowed obtaining the parameters of the neural network that best adapt to the system to be controlled. Finally, the operation of the proposed network has been evaluated by means of simulations with two dynamic systems. Superior performance of the proposed control has been demonstrated when

compared to non-linear PID control and a control reported in the literature.

It should be noted that, depending on the complexity of the control system, it might be necessary to use neural networks with more inputs, that is, with a higher degree of system response information. This has been shown by the control of the movement of an arm using an agonist–antagonist muscle model. In this example, the best performance of the proposed network has been demonstrated. The results show better control of the system with a faster response, less noise and a much lower number of neurons compared to other approach.

Finally, it is worth mentioning that, although the results obtained are encouraging, further research will be focused on application of this type of networks for control of high complexity systems. The work reported here is exploratory. Future studies should include different types of learning, such as unsupervised learning, and the use of inter-neurons to establish a more complex control logic.

Acknowledgment

This work was supported in part by the Spanish Innovation Science Ministry under Grant TRA2015-67920-R.

References

Arena, P., Fortuna, L., Frasca, M., & Patané, L. (2009). Learning anticipation via spiking networks: Application to navigation control. *IEEE Transactions on Neural Networks*, *20*(2), 202–216.

Balderas, David, & Rojas, Mario (2016). In Dr. Pedro Ponce (Ed.), *Human movement control, automation and control trends*. InTech. http://dx.doi.org/10.5772/63720.

Belatreche, A., Maguire, L. P., McGinnity, M., & Wu, Q. X. (2003). A method for supervised training of spiking neural networks. In *Proceedings of IEEE cybernetics intelligence challenges and advances, CICA* (pp. 39–44), Reading, UK.

Bienenstock, E. L., Cooper, L. N., & Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, *2*, 32–48.

Bohte, S. M., La Poutré, H., & Kok, J. N. (2002). Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks. *IEEE Transactions on Neural Networks*, *13*(2), 426–435.

Bohtea, S. M., Koka, J. N., & La Poutr, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*, 17–37.

Bouganis, A., & Shanahan, M. (2010). Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity. In *International joint conference on neural network* (pp. 1–8). http://dx.doi.org/10.1109/IJCNN.2010.5596525.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: MIT Press.

Cabrera, J. A., Castillo, J. J., Carabias, E., & Ortiz, A. (2015). Evolutionary optimization of a motorcycle traction control system based on fuzzy logic. *IEEE Transactions on Fuzzy Systems*, *23*(5).

Carrillo, R. R., Ros, E., Boucheny, C., & Olivier, J.-M. C. (2008). A real- time spiking cerebellum model for learning robot control. *BioSystems*, *94*(1), 18–27.

Chadderdon, L., Neymotin, S. A., Kerr, C. C., & Lytton, W. W. (2012). Reinforcement learning of targeted movement in a spiking neuronal model of motor cortex. *PLoS One*, *7*(10), e47251.

Clawson, T. S., Ferrari, S., Fuller, S. B., & Wood, R. J. (2016). Spiking neural network (SNN) control of a flapping insect-scale robot. In *IEEE 55th conference on decision and control, CDC* (pp. 3381–3388).

Froemke, R. C., & Dan, Y. (2002). Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature*, *416*, 433–438.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press.

Hill, A. V. (1938). The heat of shortening and the dynamic constants of muscle. *Proceeding of the Royal Society (London), Series B*, *126*, 136–195.

Hulea, M., & Caruntu, C. F. (2014). Spiking neural network for controlling the artificial muscles of a humanoid robotic arm. In *18th int. conf. syst. theory, control comput. ICSTCC 2014* (pp. 163–168).

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, *15*(5), 1063–1070.

Jin, X., Luján, M., Plana, L. A., Davies, S., Temple, S., & Furber, S. B. (2010). Modeling spiking neural networks on SpiNNaker. *Computer Science and Engineering*, *12*(5), 91–97.

Kaiser, J., Vasquez, J. C., Hubschneider, C., Wolf, P., Weber, M., & Hoff, M. et al., (2016). Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks. In *Proceedings of the 2016 IEEE int. conf. on simulation, modeling and programming for autonomous robots* (pp. 127–134).

Kandel, E. R. (2001). The molecular biology of memory storage: A dialogue between genes and synapses. *Science (80-.).*, *294*(5544), 1030–1038.

Meng, M., Wang, X., & Wang, X. (2017). Adaptive SNN torque control for tendon-driven fingers. In M. Fei, S. Ma, X. Li, X. Sun, L. Jia, & Z. Su (Eds.), *Communications in computer and information science*: *Vol. 761. Advanced computational methods in life system modeling and simulation. LSMS 2017, ICSEE 2017*. Singapore: Springer.

Oniz, Y., Aras, A. C., & Kaynak, O. (2013). Control of antilock braking system using spiking neural networks. In *IECON proceedings (industrial electronics conference)* (pp. 3422–3427).

Oniz, Y., & Kaynak, O. (2014). Variable-structure-systems based approach for online learning of spiking neural networks and its experimental evaluation. *Journal of the Franklin Institute*, *351*(6), 3269–3285.

Ponulak, F., & Kasiński, A. (2010). Supervised learning in spiking neural networks with ReSuMe sequence learning, classification, and spike shifting. *Neural Computation*, *22*, 467–510.

Ruf, B., & Schmitt, M. (1997). Learning temporally encoded patterns in networks of spiking neurons. *Neural Processing Letters*, *5*(1), 9–18.

Spüler, M., Nagel, S., & Rosenstiel, W. (2015). A spiking neuronal model learning a motor control task by reinforcement learning and structural synaptic plasticity. In *International joint conference on neural network* (pp. 1–8). http://dx.doi.org/10.1109/IJCNN.2015.7280521.

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.

Valluru, S. K., & Singh, M. (2016). Trajectory control of DC shunt motor by NARMA Level-2 neuro controller. In *2016 IEEE 1st international conference on power electronics, intelligent control and energy systems, ICPEICES*, (pp. 1–6), Delhi.

Webb, A., Davies, S., & Lester, D. (2011). Spiking neural PID controllers. In B. L. Lu, L. Zhang, & J. Kwok (Eds.), *Lecture notes in computer science*: *Vol. 7064. Neural information processing. ICONIP 2011*. Berlin, Heidelberg: Springer.

Winters, J. M., & Stark, L. (1985). Analysis of fundamental human movement patterns through the use of in-depth antagonistic muscle models. *IEEE Transactions on Biomedical Engineering*, *BME-32*(10), 826–839.