Recurrent Spiking Neural Networks the Third Generation in Identification of Systems

Nadia Adnan Shiltagh
Computer Department
College of Engineering
University of Baghdad

ABSTRACT
In this paper the modified identification method for nonlinear systems is proposed based on Recurrent Spiking Neural Networks (RSNN). Spike Response Model (SRM) has been employed in the modification method. The learning of the parameters of RSNN is based on modified backpropagation algorithm which is known as *SpikeProp*. In the identification of a variety of types of nonlinear systems, a coding equation is applied to convert real numbers into spike times. The RSNN structure is tested for the identification of the nonlinear systems. The simulation results show that the proposed modification method provides a good performance in terms of execution time and minimizing error in the training phase. .

Keywords
Spiking Neural Networks, Identification, nonlinear systems, SpikeProp

1. INTRODUCTION
Spiking neural networks are of the last generation. Compared to the other types of traditional neural networks models, spiking models have spike forwarding outputs rather than continuously varying outputs. This change comes as a generalization of the coding approaches and allows exact spike timing to be utilized as information carrier [**1**]. The last era of computational neuroscience shows spatial-temporal distribution of spikes in biological neurons. The third generation of neuron modeling (spiking neurons) is based on realization that the precise mechanism by which biological neuron encodes information is poorly understood. The models of spiking neurons have more advantages than the other models in terms of computationally efficient and biologically accurate [**2**]. The training method is one of the most important problems for applications of temporally encoded spiking neural networks. A supervised training rule, called SpikeProp based on error backpropagation (BP) is used in training method because the SpikeProp is mathematically correct without the help of the linearity assumption [3].

2. RECURRENT SPIKE NEURAL NETWORK (RSNN) MODEL
The Proposed model that is used to identify the system is based on Recurrent Spike Neural Network (RSNN) , the RSNN model which is shown in Figure 1 consist of two input neurons in the input layer , eight neurons in the hidden layer, and one neuron in output layer. Each neuron in the hidden layer has a constant self-feedback, and is activated based on Spike Response Model (SRM) .The SRM model is used to explain how the number of incoming spikes is processed to produce a new spike train leaving the neuron. In the proposed model assuming that any neuron generates at most one spike during the simulation interval. In Fig.(1) each synapse has its self-delay which is different from the delay of the other synapse, The neuron generates presynaptic single spike that increases or decreases the membrane potential. If the weighted sum of the incoming postsynaptic potentials generated by presynaptic neurons reaches a threshold value (σ), then the neuron fires [4,5].

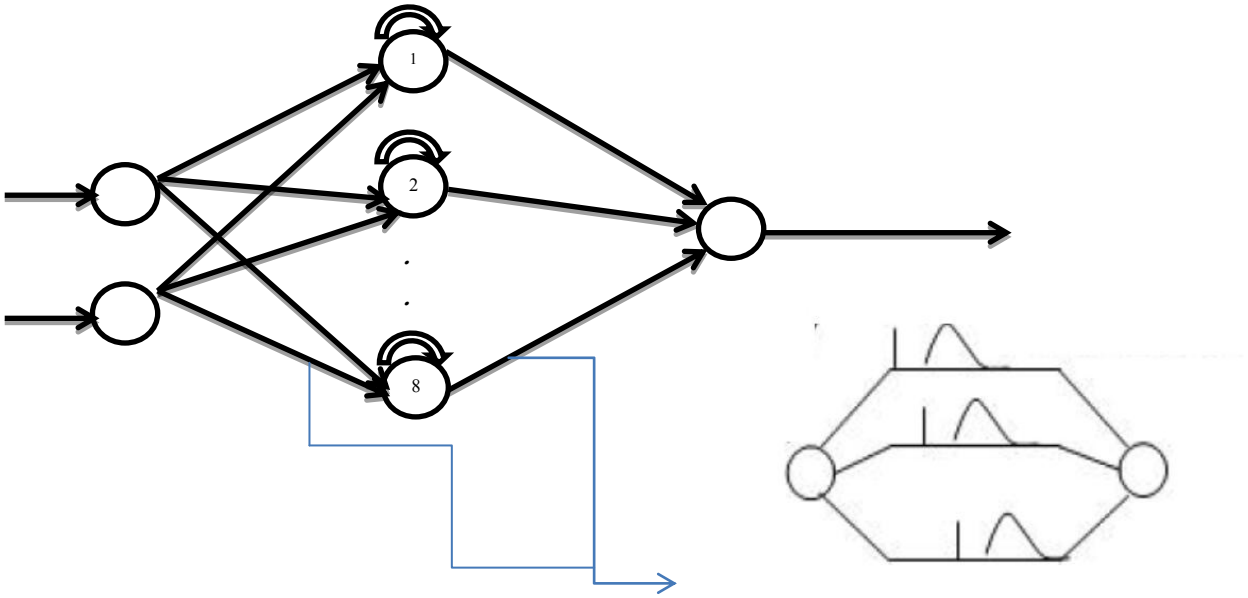

Fig 1: Recurrent Sipke Neural Network Model

3. THE TRAINING ALGORITHM

The training algorithm is based on back propagation algorithm which is modified in order to implement it on the proposed model. So the inputs to the model are the past input and output of the nonlinear plant that need to identified it, the structure of identification model is shown in Figure 2. The error between the output of the plant and output of RSNN is used to update the weights, as explained earlier not all the weights are updated at the same time as in the traditional Back Propagation algorithm, because each neurons are firing in time different from the time that other neuron fire it. And if the summations of weights reach a threshold value then this neuron is firing. Eq.(1) shows the spike response function that is used in training algorithm [6]

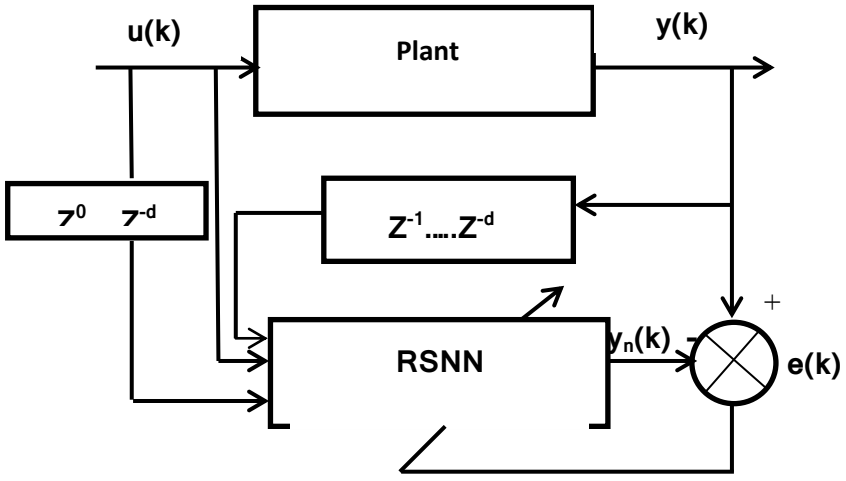

Fig 2: The Identification Model

$$\rho(t) = \begin{cases} \frac{t}{\tau} e^{1-\frac{t}{\tau}}, & t > 0 \\ 0 & t \leq 0 \end{cases} \qquad (1)$$

Where $\rho(t)$ is the spike response function, τ is called decay time constant that models membrane potential rise and decay time. The response of one synaptic is shown in Figure 3

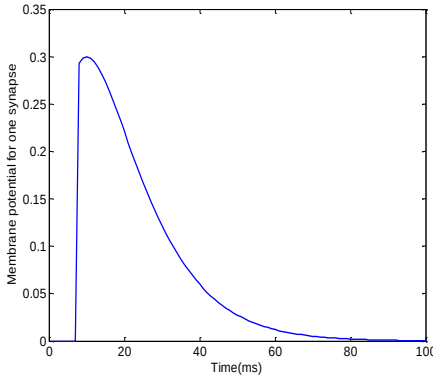

Fig 3: the postsynaptic membrane potential

The internal state of the postsynaptic neuron can be described by the following spike response function

$$g(t) = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ij}^{k} \rho(h) + g(t-1) \qquad (2)$$

where *N* is the number of presynaptic neurons, *K* is the number of synapses, $h = t - t_i - d^k$, t_i is firing time of spikes, d^K is delay, and *w* is the weight coefficient between presynaptic neuron *i* and postsynaptic neuron *j* for the k^{th} synapse. $g(t)$ is the state variable. In order to increase the capability of the network to identify the nonlinear system a self-feedback is added to the neuron in the hidden layer as shown in Fig. (1) Which acts as short term memory so the input to the network is depended only on the present input and output of the system that needs to identify it, therefore the network doesn't need too many past state of input-output information about the system that identify it.

One major distinction between types of SNNs is in regard to the number of times a neuron can fire. If a neuron can fire only once, there is no need to model what happens to a neuron after it fires, thus greatly simplifying the mathematical modelling. In the more complex case, if a neuron can fire more than once, then typically a kernel is used to model the period of absolute and relative refractoriness. *Refractoriness* refers to the inability of a neuron to immediately fire again once it has initially fired [7]

The learning rate that is used in the training algorithm is adaptive which means that it is not constant value in order to have accurate results, the adaptive eq. is as described in eq.(3) [3]. This eq. is called heuristic rule .The heuristic rule can update the learning rate in every learning step

$$(t) = \begin{cases} a * \gamma(t-1) & if\ e(n) < e(n-1) \\ b * \gamma(t-1) & if\ e(n) > k * e(n-1) \\ \gamma(t-1) & otherwise \end{cases} \qquad (3)$$

where *a*, *b* and *k* are all parameters. γ is the learning rate and *e* is the error function which can be defined as eq. (4)

$$e(t) = (t_j^a - t_j^d) \qquad (4)$$

Where t_j^a *and* t_j^d are the actual firing time and desired spike time respectively

then the derivatives of eq. (1) is

$$\rho \cdot (h) = \rho(h) * [\frac{1}{h} - \frac{1}{\tau}] \qquad (5)$$

The weights are adaptive based on error the propagated from output to hidden to input layer, this can be represented according to [4,5]

$$w_{ij}(t+1) = w_{ij}(t) + \gamma \frac{\partial e}{\partial w_{ij}^n} \qquad (6)$$

$$\frac{\partial e}{\partial w_{ij}^n} = \frac{\partial e}{\partial t_i^a} * \frac{\partial t_i^a}{\partial g(t)} * \frac{\partial g(t)}{\partial w_{ij}^n} \qquad (7)$$

The set of input-output variables that used to input to the RSNN must be encoding in to spike time as in eq. (8)

$$t_i(g) = t_{max} - round(t_{min} + \frac{(g - g_{min})(t_{max} - t_{min})}{(g_{max} - g_{\min})}) \qquad (8)$$

Where $t_i(g), t_{max}, and\ t_{min}$ are the current, the maximum and the minimum spike times respectively and g , $g_{max} and\ g_{\min}$ are the current , the maximum, and the minimum values of the input variables.

4. SIMULATION RESULTS

The program for RSNN is coded on MATLAB version 2010a. The hardware configuration of the computer used is Intel Core I7 processor with 1 T Hz CPU, 4 GB RAM and the operating system used is Windows 7 Home Premium.
The training algorithm is implemented on the RSNN to identify the nonlinear plant described by [5]

$$y(k) = 0.72y(k-1) + 0.025y(k-2)u(k-1) + 0.01u^2(k-2) + 0.2u(k-3) \qquad (9)$$

Where y(k) and u(k) are the plant output and input respectively . the trainig is implemented off line by applying random input to the plant which descibed by eq. (9) . after training the network is tested by applying another input which is described in eq. (10)

$$u(k) = \begin{cases} \sin\left(\frac{\pi k}{30}\right) & 0 < k < 250 \\ 1 & 250 \le k < 500 \\ \sin\left(\frac{2k}{80}\right) & 500 \le k < 750 \\ 0.3\sin\left(\frac{\pi k}{25}\right) + 0.1\sin\left(\frac{\pi k}{32}\right) + 0.6\sin\left(\frac{\pi k}{10}\right) & 750 \le k \le 1000 \end{cases} \qquad (10)$$

The structure of SRNN that is used is 3-5-1, and the learing rate γ is initiate at 0.01 and it will adaptive according to eq. (3). The $\boldsymbol{d^K}$ parameter is defined the number of synapses in each connection.The othar parameters that are used in taining phase are given in table 1.Figure 4 shows the output of SRNN and the output of plant after traning. It is clear from the figure that the SRNN can identify the unkown plant effeciently. The Sum Square Error (SSE) is shown in figure 5. Figure 6 explains the output of the spike response function of the connection between first neuron in input layer and the first neuron of hidden layer with delay to illustrate that the updating of each weight connection is updating after firing and when the output of the spike response function exceeds the threshold value. Figure 7 shows the output of spike response function for all connections and the sum of the spike

response function, it is clear from the figure that each connection between neuron in input layer and the neuron in the hidden layer is delayed by d^K, so in this example the d^K is 6, which means that more accurate of identification system is result but with more execution time. The increasing of execution time is not very important matter because the identification process is done offline.

Table 1: The simulation parameters in traning phase

Parameters	Value
Initial weighs	[-1,1]
t_{max},	10
t_{min}	-10
d^K	6
τ	4
σ	2

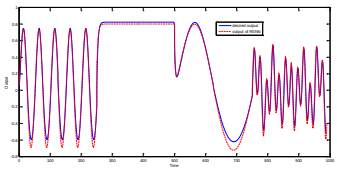

Fig 4: The desired output and the SRNN output

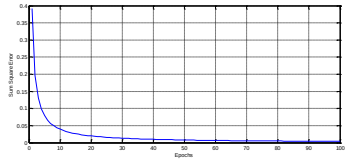

Fig 5: The Sum Square Error with the Number of Epochs

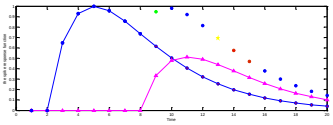

Fig 6: The output of spike response function

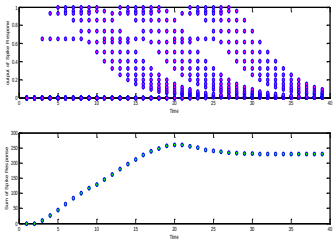

Fig 7: The output of spike response function for all connections and the sum of the spike response function

5. CONCLUSION

In this paper, the modified structure of Recurrent Spiking Neural Network is presented to identify the nonlinear system. The structure is based on recurrence, the past values of the state variables in the hidden layer, to increase the memory of the spike neuron in order to identify the unknown system. The structure is trained using the SpikePro training algorithm with some modification in the output of the state variables. The implementation of the proposed structure on the nonlinear system shows the accuracy of it in identification. In addition this structure, with modified training method, shows the minimization of the error between desired and actual values. The efficiency of the spiking neural network is demonstrated in that the spike neural does not need to update all weights at the same time, however, only the firing neuron which passes the threshold value is updated; this is entirely different from traditional neural networks which need to update all the weights at the same time. Also one can notice from the results that the number of epochs is few in comparison with traditional neural networks.

6. REFERENCES

[1] I. BOGDANOV R. MIRSU V. TIPONUT 2009,"Matlab Model for Spiking Neural Networks**,** Proceedings of the 13th WSEAS International Conference on SYSTEMS,

[2] H. Raza , G. Tsegaye and R. Biswas, 2012 Fuzzy Based Modified SHL algorithm for Spiking Neural Networks, *International Journal of Computer Applications ,Vol.41– No.5, March 2012*

[3] F. Huijuan, L. Jiliang and W. Fei,2012 Fast Learning in Spiking Neural Networks by Learning Rate Adaptation**,** *Chinese Journal of Chemical Engineering*.

[4] R. H.Abiyev, O. Kaynak ,*and* Y. Oniz,, 2012 Spiking Neural Networks for Identification and Control of Dynamic Plants, The 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics July 11-14, Kaohsiung, Taiwan**,**2012

[5] Y. Oniz, O. Kaynak ,*and* R. H.Abiyev, 2013,Spiking Neural Networks for the Control of a Servo System, International Conference on Mechatronics (ICM), IEEE, Feb. 27 2013-March 1 2013

[6] V. Thiruvarudchelvan, J. Crane and T. Bossomaier, 2013, Analysis of SpikeProp Convergence with Alternative Spike Response Functions", IEEE, 2013.

[7] S. McKennoch, D. Liu, and L. Bushnell, 2006, Fast Modifications of the SpikeProp Algorithm, IJCNN '06. International Joint Conference on Neural Networks, 2006