Systems Modeling Using Deep Elman Neural Network

Latifa Belhaj Salah
Control and Energy Management Laboratory (CEM-Lab)
University of Gabes, Tunisia
latifa.belhadjsalah@enis.tn

Fathi Fourati
Control and Energy Management Laboratory (CEM-Lab)
University of Sfax, Tunisia
fethi.fourati@ipeis.rnu.tn

***Abstract*—In this paper, the modeling of complex systems using deep Elman neural network architecture is improved. The emphasis is to retrieve better deep Elman structure that emulates perfectly such dynamic systems. To achieve this goal, sigmoid activation functions in the hidden and output layer nodes are chosen and data files on considered systems for modeling and validation steps are given. Simulation results prove the ability and the efficiency of a deep Elman neural network with two hidden layers in this task.**

Keywords-Elman neural network; recurrent neural network; deep learning; complex systems; modeling

I. INTRODUCTION

Recently, deep neural networks (including recurrent networks) have been successfully applied in several areas [1]: modeling and control of complex systems, fault detection, text understanding [2, 3], speech recognition [4, 5] and computer vision [6, 7]. The advantage of deep learning is reflected on the modeling of high-level abstractions from the data by its architecture which is consisted of several non-linear learning layers. Each layer replies to several levels of nonlinear abstraction. [8]. In [9], authors used recurrent neural networks with long-term memory units (LSTMs) to find a solution to the vanishing gradient problem observed in simple recurrent neural networks (RNNs) [9-10]. Authors in [11] used the pre-training and fine-tuning steps to ensure the effective training of deep learning. Deep learning has a significant success in big data since it can recover valuable information from complex processes [12]. In [13], authors have described a combination between deep learning and reinforcement learning for the prediction and control of intelligent laser welding. The effectiveness of using deep Elman RNN for modeling complex systems was demonstrated in [14]. In order to improve the performance of deep learning with multiplayer Perceptron (MLP), a new technique was used that requires a combination of adaptive learning rate and Laplacian score concept to vary the weights. In [15], authors proposed a new method for automatic modulation classification (AMC) based on unsupervised feed-forward deep learning. The performance of this approach was compared to conventional AMC techniques. In [16], authors used the application of end-to-end deep learning to solve the classification problem of speech emotion recognition. They showed the limitations and benefits of these architectures in speech recognition. In [17], authors used deep Elman RNN (ERNN) for acoustic modeling. They compared the performance of this technique with other RNN architectures like LSTM, GRU, and simplified LSTM [18-20]. In [21], authors developed two novel deep RNN models with LSTM units to predict building electricity consumption. In [22], authors proposed a new learning algorithm based on simplified convolutional neural network to achieve visual tracking with adaptive filtering of particles. In [23], authors used convolutional deep learning neural networks to detect and diagnose plant diseases from leaves images of healthy and diseased plants. In [24], authors exploited deep learning to solve the problem of large data analysis which is found in several areas. In [25], authors proposed the use of a new method based on deep belief networks and multiple models (DBNs-MMs) to detect faults of complex systems. The novelty in this paper is the use of a deep neural network that improves modeling systems tasks. In fact, comparing to results given in [26], an Elman neural network with two hidden layers gives more accuracy when modeling complex systems.

II. ELMAN NEURAL NETWORK

The Elman network is a type of recurrent network [27]. It has been applied in many areas such as dynamic system identification [28] and financial prediction [29]. In [27], a simple recurrent neural network was proposed: the input and output units are in contact with the external environment, in contrast to the context and hidden units. This network is characterized by context units that are used to save the previous hidden unit activations. Hidden unit activations at time *k* are returned to the context units and stored for the next training step. The Elman network is named partially recurrent network because the feed forward connections are modifiable and the recurrent connections are fixed. To train this kind of neural network, the back propagation algorithm can then be used and the hidden units activation functions can be linear or non-linear [17, 27, 30]. Figure 1 represents an Elman neural network architecture. The *l*-th input unit to the network is represented by $VE_l(k)$ and the *m*-th network output unit by $neto_m(k)$. The total input to the *i*-th first hidden layer unit is denoted as $net_i^1(k)$. The output of the *i*-th first hidden layer unit is denoted as $net_i^{11}(k)$. The output of the *j*-th context layer unit is $V_j^c(k)$. The total input to the *ii*-th second hidden layer unit is denoted

Corresponding author: Latifa Belhaj Salah

as $net_{ii}^{2}(k)$. The output of the *ii*-th second hidden layer unit is denoted as $net_{ii}^{22}(k)$. The output of the *jj*-th second context layer unit is $V_{jj}^{c_1}(k)$. The total input to the *i..i*-th *(n-1*-th*)* hidden layer unit is denoted as $net_{i..i}^{n-1}(k)$.The output of the *i..i*-th *(n-1*-th*)* hidden layer unit is denoted as $net_{i..i}^{(n-1)(n-1)}(k)$. The output of the *j..j*-th *(n-1*-th*)* context layer unit is $V_{j..j}^{c_{n-1}}(k)$. The total input to the *i..ii*-th *(n*-th*)* hidden layer unit is denoted as $net_{i..ii}^{n}(k)$. The output of the *i..ii*-th *(n*-th*)* hidden layer unit is denoted as $net_{i..ii}^{nn}(k)$. The output of the *j..jj*-th *(n*-th*)* context layer unit is $V_{j..jj}^{c_n}(k)$. The total input to the *m*-th output layer unit is denoted as $net_{m}^{s}(k)$. $w_{j,i}^{c}(.)$, $w_{l,i}^{VE}(.)$, $w_{i..ii,m}^{neto}(.)$, $w_{i,ii}^{net^2}(.)$, $w_{jj,ii}^{c_1}(.)$, $w_{i..i,i..ii}^{net^n}(.)$, $w_{j..jj,i..ii}^{c_n}(.)$ are the weights of the links, respectively between the first context layer and the first hidden layer, the input layer and the first hidden layer, the *i..ii*-th *(n*-th*)* hidden layer and the output layer, the first hidden layer and the second hidden layer, the second context layer and the second hidden layer, the *(n-1)*-th hidden layer and the *n*-th hidden layer and the *n*-th context layer and the *n*-th hidden layer.

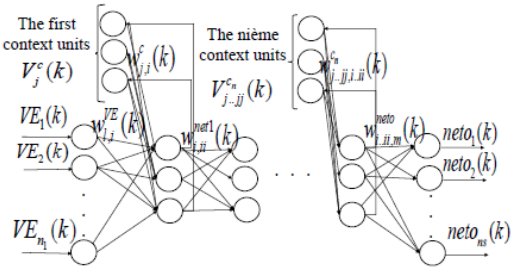

Fig. 1. Elman neural network architecture.

III. ELMAN NEURAL NETWORK LEARNING STEP

The training of the Elman neural network to emulate direct dynamics of a system amounts to minimize the squared error criterion defined as:

$$E_k = \frac{1}{2}\sum_{m=1}^{n_s}(VS_m(k) - neto_m(k))^2 \quad (1)$$

where *VS(k)* is the desired output vector.

In the training step the back propagation algorithm is used to adjust the Elman neural network connection weights, in order to emulate the dynamics of a system. In the case of many hidden layers, the Elman neural network is governed by the following equations:

$$net^1{}_i(k) = \sum_{j=1}^{n_c} w_{j,i}^{c}(k-1)V_j^c(k) + \sum_{l=1}^{n_{VE}} w_{l,i}^{VE}(k-1)VE_l(k) \quad (2)$$

$$net^{11}{}_i(k) = f(net^1{}_i) \quad (3)$$

$$V_j^c(k) = net^{11}{}_j(k-1) \quad (4)$$

$$net^2{}_{ii}(k) = \sum_{i=1}^{n_h} w_{i,ii}^{net^2}(k-1)net^{11}{}_i(k) + \sum_{jj=1}^{n_{c_1}} w_{jj,ii}^{c_1}(k-1)V_{jj}^{c_1}(k) \quad (5)$$

$$net^{22}{}_{ii}(k) = f(net^2{}_{ii}(k)) \quad (6)$$

$$V_{jj}^{c_1}(k) = net^{22}{}_{jj}(k-1) \quad (7)$$

$$net_{i..ii}^{n}(k) = \sum_{i..i=1}^{n_{h_{n-1}}} w_{i..i,i..ii}^{net^n}(k-1)net_{i..i}^{(n-1)(n-1)}(k) + \sum_{j..jj=1}^{n_{c_n}} w_{j..jj,i..ii}^{c_n}(k-1)V_{j..jj}^{c_n}(k) \quad (8)$$

$$net_{i..ii}^{nn}(k) = f(net_{i..ii}^{n}(k)) \quad (9)$$

$$V_{j..jj}^{c_n}(k) = net_{j..jj}^{nn}(k-1) \quad (10)$$

$$net_{m}^{s}(k) = \sum_{i..ii=1}^{n_{h_n}} w_{i..ii,m}^{neto}(k-1)net_{i..ii}^{nn}(k) \quad (11)$$

$$neto_m(k) = f(net_m^s) \quad (12)$$

where n_c, n_h, n_{VE} , n_{h_1}, n_{c_1} , $n_{h_{n-1}}$, n_{h_n}, n_{c_n} represent the number of units respectively in the first context layer, the first hidden layer, the input layer, the second hidden layer, the second context layer , the (*n*-1)th hidden layer, the *n*-th hidden layer and the *n*-th context layer.

The squared error at the network output is defined as in (1). For $w_{i..ii,m}^{neto}(k-1)$ the error gradient is:

$$\frac{\partial E_k}{\partial w_{i..ii,m}^{neto}(k-1)} = -(VS_m(k) - neto_m(k))f'_{net_m^s}net_{i..ii}^{nn}(k) \quad (13)$$

for $w_{l,i}^{VE}(k-1)$, $w_{j,i}^{c}(k-1)$, $w_{i,ii}^{net2}(k-1)$, $w_{jj,ii}^{c_1}(k-1)$, $w_{i..i,i..ii}^{net^n}(k-1)$ and $w_{j..jj,i..ii}^{c_n}(k-1)$.

$$\frac{\partial E_k}{\partial w_{l,i}^{VE}(k-1)} = -(VS_m(k) - neto_m(k))f'_{net_m^s}w_{i..ii,m}^{neto}(k-1) f'_{net_{i..ii}^n}w_{i..i,i..ii}^{net^n}(k-1)f'_{net_{i..i}^{n-1}}...f'_{net_{ii}^2}w_{i,ii}^{net^2}(k-1)f'_{net_i^1}VE_l(k) \quad (14)$$

$$\frac{\partial E_k}{\partial w_{j,i}^{c}(k-1)} = -(VS_m(k) - neto_m(k))f'_{net_m^s}w_{i..ii,m}^{neto}(k-1) f'_{net_{i..ii}^n}w_{i..i,i..ii}^{net^n}(k-1)f'_{net_{i..i}^{n-1}}...f'_{net_{ii}^2}w_{i,ii}^{net^2}(k-1)f'_{net_i^1}V_j^c(k) \quad (15)$$

$$\frac{\partial E_k}{\partial w_{i,ii}^{net2}(k-1)} = -(VS_m(k) - neto_m(k))f'_{net_m^s}w_{i..ii,m}^{neto}(k-1) f'_{net_{i..ii}^n}w_{i..i,i..ii}^{net^n}(k-1)f'_{net_{i..i}^{n-1}}...f'_{net_{ii}^2}net_i^{11}(k) \quad (16)$$

$$\frac{\partial E_k}{\partial w_{jj,ii}^{c_1}(k-1)} = -(VS_m(k) - neto_m(k))f'_{net_m^s}w_{i..ii,m}^{neto}(k-1) f'_{net_{i..ii}^n}w_{i..i,i..ii}^{net^n}(k-1)f'_{net_{i..i}^{n-1}}...f'_{net_{ii}^2}V_{jj}^{c_1}(k) \quad (17)$$

$$\frac{\partial E_k}{\partial w_{i..i,i..ii}^{net^n}(k-1)} = -(VS_m(k) - neto_m(k))f'_{net_m^s}w_{i..ii,m}^{neto}(k-1) f'_{net_{i..ii}^n}net_{i..i}^{(n-1)(n-1)} \quad (18)$$

$$\frac{\partial E_k}{\partial w^{net^n}_{i..i,i..ii}(k-1)} = -(VS_m(k) - neto_m(k))\, f'_{net^s_m} w^{neto}_{i..ii,m}(k-1)$$
$$f'_{net^n_{i..ii}} net^{(n-1)(n-1)}_{i..i} \quad (19)$$

In (13) $f'_{net^s_m}$ denotes the derivative of f representing net^s_m. The general weight modification in the gradient descent method is:

$$\Delta w = -\mu \frac{\partial E_k}{\partial w} \quad (20)$$

Where:

- The weight adjustment between the output and the n-th hidden layer are:

$$\Delta w^{neto}_{i..ii,m}(k) = \mu(VS_m(k) - neto_m(k))\, f'_{net^s_m} net^{nn}_{i..ii}(k) \quad (21)$$

- The weights adjustment between the first hidden layer and the input layer are:

$$\Delta w^{VE}_{l,i}(k) = \mu(VS_m(k) - neto_m(k))\, f'_{net^s_m} w^{neto}_{i..ii,m}(k-1)\, f'_{net^n_{i..ii}}$$
$$w^{net^n}_{i..i,i..ii}(k-1)\, f'_{net^{n-1}_{i..i}} \ldots f'_{net^2_{ii}} w^{net^2}_{i,ii}(k-1)\, f'_{net^1_i} VE_l(k) \quad (22)$$

- The weights adjustment between the first hidden and the context layers are:

$$\Delta w^{c}_{j,i}(k) = \mu(VS_m(k) - neto_m(k))\, f'_{net^s_m} w^{neto}_{i..ii,m}(k-1)\, f'_{net^n_{i..ii}}$$
$$w^{net^n}_{i..i,i..ii}(k-1)\, f'_{net^{n-1}_{i..i}} \ldots f'_{net^2_{ii}} w^{net^2}_{i,ii}(k-1)\, f'_{net^1_i} V^c_j(k) \quad (23)$$

- The weights adjustment between the first hidden layer and the second hidden layer are:

$$\Delta w^{net^2}_{i,ii}(k) = \mu(VS_m(k) - neto_m(k))\, f'_{net^s_m} w^{neto}_{i..ii,m}(k-1)$$
$$f'_{net^n_{i..ii}} w^{net^n}_{i..i,i..ii}(k-1)\, f'_{net^{n-1}_{i..i}} \ldots f'_{net^2_{ii}} net^{11}_i(k) \quad (24)$$

- The weights adjustment between the second hidden and the context layers are:

$$\Delta w^{c_1}_{jj,ii}(k-1) = \mu(VS_m(k) - neto_m(k))\, f'_{net^s_m} w^{neto}_{i..ii,m}(k-1)$$
$$f'_{net^n_{i..ii}} w^{net^n}_{i..i,i..ii}(k-1)\, f'_{net^{n-1}_{i..i}} \ldots f'_{net^2_{ii}} V^{c_1}_{jj}(k) \quad (25)$$

- The weights adjustment between the n-th hidden and context layers are:

$$\Delta w^{c_n}_{j..jj,i..ii}(k) = \mu(VS_m(k) - neto_m(k))$$
$$f'_{net^s_m} w^{neto}_{i..ii,m}(k-1)\, f'_{net^n_{i..ii}} V^{c_n}_{j..jj} \quad (26)$$

- The weights adjustment between the n-th hidden layer and the (n-1-th) hidden layer are:

$$\Delta w^{net^n}_{i..i,i..ii}(k) = \mu(VS_m(k) - neto_m(k))\, f'_{net^s_m}$$
$$w^{neto}_{i..ii,m}(k-1)\, f'_{net^n_{i..ii}} net^{(n-1)(n-1)}_{i..i} \quad (27)$$

IV. CONSIDERED SYSTEMS

Complex systems are generally multi-variable, described by non-linearity, characterized by uncertainties and subject to disturbances [31]. The goal of modeling a complex system consists of constructing a concise and efficient neural Elman structure to emulate the dynamics of a system [32, 33]. At first we consider a single input and a single output (SISO) system with a non-linear function $f(.)$, as in Figure 2.

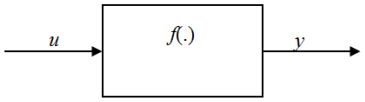

Fig. 2. A SISO sytem

Here,

$$y = \sin(u) = f(u) \quad (28)$$

The complex system to be modeled is a greenhouse with multi-inputs, multi-outputs (MIMO), disturbances and uncertainty. It is composed of sensors to measure internal and external climate. The greenhouse outputs are defined by the internal climate ("internal temperature and internal hygrometry"). It is equipped with actuators to control the internal climate, consisted with a sliding shutter with an opening between 0° and 35°, a heater which operating in on/off mode with a power of 5kW, a sprayer and a curtain with a length varying between 0 and 3m. The considered greenhouse is a classical one, it is characterized by physical quantities that constitute its functioning [34, 35]:

- Measurable but not controllable input: *Te* (external temperature (in ºC), *He* (external hygrometry in %), *Rg* (global radiant in W/m^2), *Vv* (wind speed in km/h).
- Measurable and controllable input: *Ch* (heating input varying between 0 and 1), *Ov* (sliding shutter in degrees), *Br* (sprayer varying between 0 and 1), *Om* (curtain in m).
- Outputs: *Ti* (internal temperature in *ºC*), *Hi* (internal hygrometry in %).

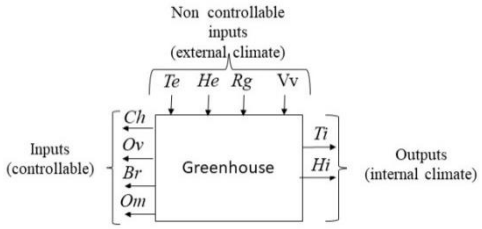

Fig. 3. Greenhouse functional bloc diagram.

V. SIMULATION RESULTS

Three neural network structures are considered for both systems: the first is with one hidden and context layers, the second one is constituted of two hidden and context layers and the third one has three hidden and context layers. Table I presents the deep Elman neural network caracteristics modeling the non-linear process.

TABLE I. DEEP ELMAN NEURAL NETWORK -NON-LINEAR PROCESS.

Parameters	One hidden and context layers	Two hidden and context layers	Three hidden and context layers
n_1	1	1	1
n_s	1	1	1
n_{c_1}	4	4	4
n_{h_1}	4	4	4
n_{c_2}		4	4
n_{h_2}		4	4
n_{c_3}			4
n_{h_3}			4
iterations	3000	3000	10000
Learning coefficient	0.2	0.2	0.4

TABLE II. DEEP ELMAN NEURAL NETWORKW-GREENHOUSE

Parameters	One hidden and context layers	Two hidden and context layers	Three hidden and context layers
n_1	8	8	8
n_s	2	2	2
n_{c_1}	4	4	4
n_{h_1}	4	4	4
n_{c_2}		4	4
n_{h_2}		4	4
n_{c_3}			4
n_{h_3}			4
iterations	10000	10000	20000
Learning coefficient	0.1	0.2	0.4

Figures 4-6 represent the evolution of the criterion (1) for the first non-linear process. Table II presents deep Elman neural network characteristics modeling the greenhouse. Figures 7-9 present the evolution of the criterion *Ek* in the case of the second process (greenhouse). From the Figures and for the two considered systems we conclude that the error E_k is lower in the case of two hidden and context layers network. Thus, the convergence of the error is faster in the case of the neural structure with a single hidden and context layer than the other neural structures.

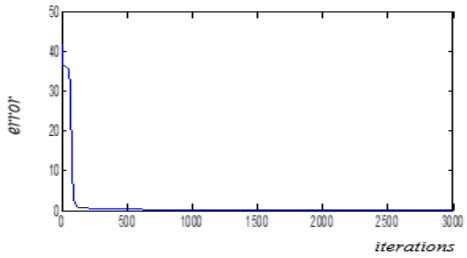

Fig. 4. Evolution of *Ek* in the case of one hidden layer network.

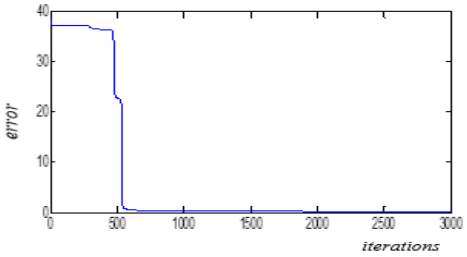

Fig. 5. Evolution of *Ek* in the case of two hidden layers network.

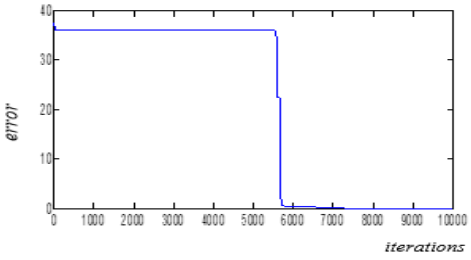

Fig. 6. Evolution of *Ek* in the case of three hidden layers network.

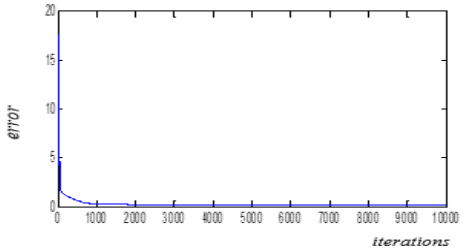

Fig. 7. Evolution of *Ek* in the case of one hidden layer network.

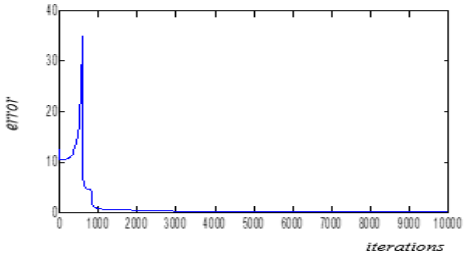

Fig. 8. Evolution of *Ek* in the case of two hidden layers network.

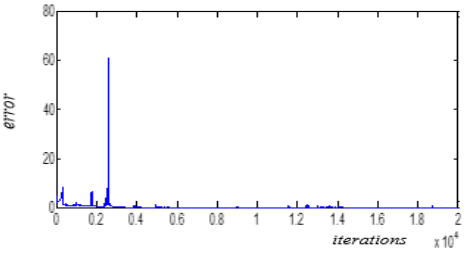

Fig. 9. Evolution of *Ek* in the case of three hidden layers network.

In order to compare and validate the three neural models the criterion (29) is considered:

$$E_t = \sum_{k=1}^{nb}\sum_{i=1}^{n_s} abs\left(y_i^m(k) - y_i(k)\right) \qquad (29)$$

where, *nb* is the operating interval, *k* is the sample time, *ns* is the number of outputs, $y_i(k)$ is the *i*-th output of the system at time k and $y_i^m(k)$ is the *i*-th output of the neural model at time *k*. In the case of the first system, *nb*=1000.

Figures 10-12 represent the evolution of the process output (continuous line) and the neural model output (dashed lines).

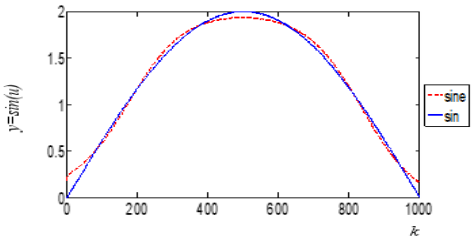

Fig. 10. Evolution of the process and the one hidden layer neural model outputs. *Et*=59.496.

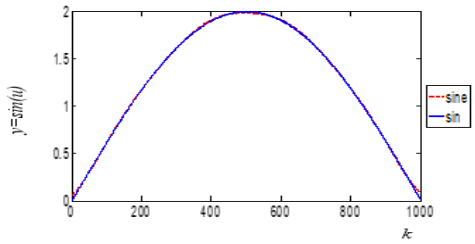

Fig. 11. Evolution of the process and the two hidden layer neural model outputs. *Et*=57.8332.

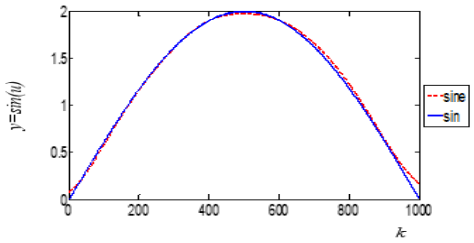

Fig. 12. Evolution of the process and the three hidden layer neural model outputs. *Et*=58.9947.

In the case of the second system (greenhouse), we have a data file of the whole parameters' values acting on the greenhouse during one day. The sampling time is one minute, so in one day (24 hours), we obtain a data file with 1440 lines. We divided the data file in two parts, each part constituted of 720 lines. The first part was used for learning (training) and the second was used for the validation. In our case:

$VE_l(k)=[Ov(k),Ch(k),Br(k),Om(k),Te(k),He(k),Rg(k),Vv(k)]^T$,$neto_m(k)=[Ti(k)\ Hi(k)]$. Here nb=720.

Figures 13 and 14 represent the evolution of the real internal climate (temperature and hygrometry) with continuous lines and the one hidden layer neural model outputs with dashed lines and using the validation data file part. Here *Et*=88.7962.

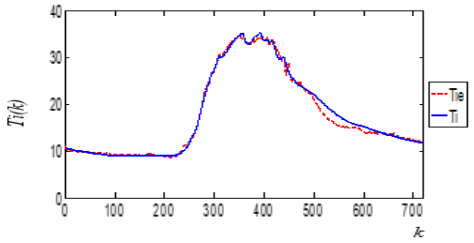

Fig. 13. Evolution of the internal temperature.

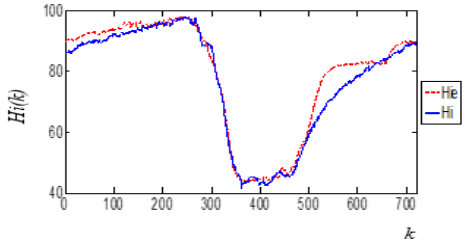

Fig. 14. Evolution of the internal hygrometry.

Figures 15 and 16 represent the evolution of the real internal climate (temperature and hygrometry) with continuous lines and the two hidden layers neural model outputs with dashed lines and using the validation data file part (*Et*=82.4499). Figures 17 and 18 represent the evolution of the real internal climate with continuous lines and the three hidden layers neural model outputs with dashed lines and using the validation data file part (*Et*=90.5219).

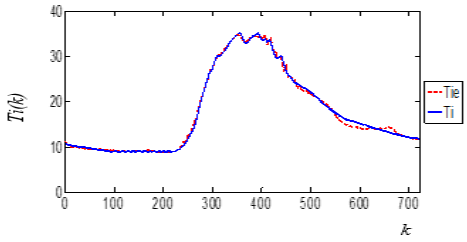

Fig. 15. Evolution of the internal temperature.

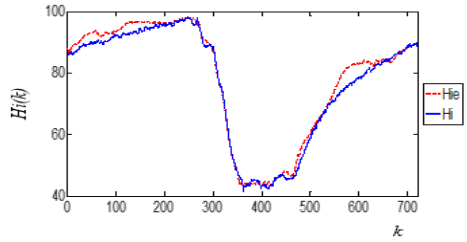

Fig. 16. Evolution of the internal hygrometry.

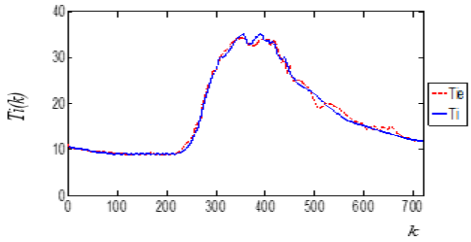

Fig. 17. Evolution of the internal temperature.

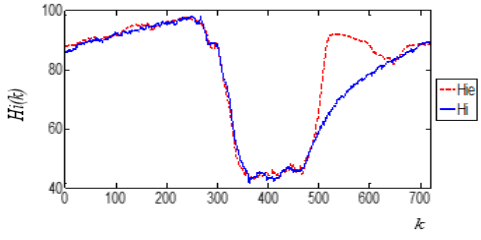

Fig. 18. Evolution of the internal hygrometry.

From the previous figures and results, we conclude that the total error is lower in the case of two hidden and context layers than the one of the other neural networks. This Elman neural structure gives better performance results for the modeling and validation steps.

VI. CONCLUSION

In this paper, we trained a deep Elman neural network to improve the modeling of complex systems. After observing the simulation results, we concluded that the direct model reproduces the dynamic behavior of two processes with acceptable performance. We showed that an Elman network with two hidden and two context layers is the best and the most efficient structure for modeling the complex process. The obtained model will be used in a control oriented task.

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview", Neural Networks, Vol. 61, pp. 85-117, 2015

[2] N. Majumder, S. Poria, A. Gelbukh, E. Cambria, "Deep learning-based document modeling for personality detection from text", IEEE Intelligent Systems, Vol. 32, pp. 74-79, 2017

[3] Z. Jiang, L. Li, D. Huang, L. Lin, "Training word embeddings for deep learning in biomedical text mining tasks", IEEE International Conference on Bioinformatics and Biomedicine, Washington, DC, USA, November 9-12, 2015

[4] L. Deng, G. Hinton, B. Kingsbury, "New types of deep neual network learning for speech recognition and related applications: an overview", IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, Canada, May 26-31, 2013

[5] D. Chen, B. K. W. Mak, "Multitask learning of deep neural networks for low-resource speech recognition", IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol. 23, No. 7, pp. 1172-1183, 2015

[6] S. M. S. Islam, S. Rahman, M. M. Rahman, E. K. Dey, M. Shoyaib, "Application of deep learning to computer vision: a comprehensive study", 5th International Conference on Informatics, Electronics and Vision, Dhaka, Bangladesh, May 13-14, 2016

[7] N. Kruger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez, L. Wiskott, "Deep hierarchies in the primate visual cortex: what can we learn for computer vision", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, No. 8, pp. 1847-1871, 2013

[8] M. Chengcai, G. Xiaodong, W. Yuanyuan, "Fault diagnosis of power electronic system based on fault gradation and neural network group", Neurocomputing, Vol. 72, pp. 2909-2914, 2009

[9] S. Hochreiter, J. Schmidhuber, "Long short-term memory", Neural Computing, Vol. 91, pp. 735-780, 1997

[10] Z. C. Lipton, J. Berkowitz, C. Elkan, "A critical review of recurrent neural networks for sequence learning", available at: https://arxiv.org/abs/1506.00019, 2015

[11] G. E. Hinton, R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", Science, Vol. 313, pp. 504-507, 2006

[12] X. Chen, X. Lin, "Big data deep learning: challenges and perspectives", IEEE Access, Vol. 2, pp. 514-525, 2014

[13] J. Gunther, P. M. Pilarski, G. Helfrich, H. Shen, K. Diepold, "Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning", Mechatronics, Vol. 34, pp. 1-11, 2016

[14] B. Chandra, R. K. Sharma, "Deep learning with adaptive learning rate using Laplacian score", Expert Systems with Applications, Vol. 63, pp. 1-7, 2016

[15] A. Ali, F. Yangyu, "Unsupervised feature learning and automatic modulation classification using deep learning model", Physical Communication, Vol. 25, No. 1, pp. 75-84, 2017

[16] H. M. Fayek, M. Lech, L. Cacedon, "Evaluating deep learning architectures for Speech Emotion Recognition", Neural Networks, Vol. 92, pp. 60-68, 2017

[17] S. Achanta, S. V. Gangashetty, "Deep Elman Recurrent Neural Networks for Statistical Parametric Speech Synthesis", Speech Communication", Vol. 93, pp. 31-42, 2017

[18] H. Zen, H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis", in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4470-4474, IEEE, 2015

[19] Y. Fan, Y. Qian, F. L. Xie, F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks", Interspeech 2014, Singapore, September 14-18, 2014

[20] Z. Wu, S. King, "Investigating gated recurrent networks for speech synthesis", 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, Shanghai, China, March 20-25, 2016

[21] A. Rahman, V. Srikumar, A. D. Smith, "Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks", Applied Energy, Vol. 212, pp. 372-385, 2018

[22] X. Qian, L. Han, Y. Wang, M. Ding, "Deep learning assisted robust visual tracking with adaptive particle filtering", Signal Processing: Image Communication, Vol. 60, pp. 183-192, 2018

[23] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis", Computers and Electronics in Agriculture, Vol. 145, pp. 311-318, 2018

[24] Q. Zhang, L. T. Yang, Z. Chen, P. Li, "A survey on deep learning for big data", Information Fusion, Vol. 42, pp. 146-157, 2018

[25] R. Hao, C. Yi, J. Qu, Y. Xin, T. Qiu, "A novel adaptive fault detection methodology for complex system using deep belief networks and multiple models: A case study on cryogenic propellant loading system", Neurocomputing, Vol. 275, pp. 2111-2125, 2018

[26] F. Fourati, M. Chtourou, "A greenhouse control with feed-forward and recurrent neural networks", Simulation Modelling Practice and Theory, Vol. 15, No. 8, pp. 1016-1028, 2007

[27] J. L. Elman, "Finding structure in time", Cognitive Science, Vol. 14, No. 2, pp. 179-211, 1990

[28] D. T. Pham, X. Liu, "Dynamic system modeling using partially recurrent neural networks", Journal of Systems Engineereing", Vol. 2, No. 2, pp. 90-97, 1992

[29] K. Kamijo, T. Tanigawa, "Stock price pattern recognition-a recurrent neural network approach", International Joint Conference on Neural Networks, San Diego, CA, USA, June 17-21, 1990

[30] D. T. Pham, X. Liu, "Training of Elman networks and dynamic system modelling", International Journal of Systems Science Vol. 27, No. 2, pp. 221-226, 1996

[31] A. Yan, W. Wang, C. Zhang, H. Zhao, "A fault prediction method that uses improved case-based reasoning to continuously predict the status of a shaft furnace", Information Sciences, Vol. 259, pp. 269-281, 2014

[32] F. Baghernezhad, K. Khorasani, "Computationally intelligent strategies for robust fault detection, isolation, and identification of mobile robots", Neurocomputing, Vol. 171, pp. 335-346, 2016

[33] H. B. Huang, X. R. Huang, R. X. Li, T. C. Lim, W. P. Ding, "Sound quality prediction of vehicle interior noise using deep belief networks", Applied Acoustics,Vol. 113, pp. 149-161, 2016

[34] D. Psaltis, A. Sideris, A. A. Yamamura, "A multilayer neural network Controller", IEEE International Conference on Neural Networks, San Diego, California, June 21-24, 1987

[35] M. Souissi, Modelisation et Commande du Climat d'une Serre Agricole, PhD Thesis, University of Tunis, Tunis, 2002 (in French)