Identification and control of nonlinear dynamical systems using Levenberg-Marquardt learning algorithm for recurrent complex-valued neural networks

Ieroham S. Baruch, Victor Arellano Quintana and Edmundo P. Reynaud

***Abstract*—**In this work, a recursive Levenberg-Marquardt (LM) learning algorithm in the complex domain is developed and applied to the learning of an adaptive control scheme composed by Complex-Valued Recurrent Neural Networks (CVRNN). We simplified the derivation of the LM learning algorithm using a diagrammatic method to derive the adjoint CVRNN used to obtain the gradient terms. Furthermore, we apply the CVRNN control scheme for a particular case of a nonlinear, oscillatory mechanical plant to validate the performance of the adaptive neural controller and the learning algorithm. The obtained simulation results using a flexible robot arm confirm a good performance of the derived control schemes and learning algorithms to suppress the occurred robot oscillations and tracking error.

***Keywords*—**Complex-valued Levenberg-Marquardt learning, direct adaptive neural control, diagrammatic rules, recurrent complex-valued neural network topology, system identification of nonlinear oscillatory plants.

I. Introduction

THE rapid growth of available computational resources has led to the developments of a wide number of Neural Networks (**NN**) based modeling, identification, prediction and control applications [1-3]. Some other applications of neural and fuzzy-neural networks have been done for oscillatory chaotic systems, [4], [5]. The main NN property, namely the ability to approximate complex nonlinear relationships without prior knowledge of model structure, makes them a very attractive alternative to the classical modeling and control techniques [6-8]. Among several possible network architectures the ones most widely used are the Feed-Forward **NN** (**FFNN**) and Recurrent **NN** (**RNN**) [6]. In the last decade there has been a rise in applications using Recurrent Complex-Valued **NN**s (**RCVNN**) [9-15]. Most of them deal with oscillatory systems which, by their physical nature, are convenient to be treated in the complex domain, such as electromagnetic waves, light waves, images processing, electric power systems, evaporator systems and mechanical systems [9-11]. In [12], the authors derived a Complex-Valued Backpropagation (**CVBP**) algorithm used for pattern classification. However, the learning algorithm presented some problems because the activation functions presented singularity points in their domains. Some other articles [13], [14] and [15], propose different activation functions that avoid singularity points. To simplify the Levenberg-Marquardt (**LM**), learning for the **RCVNN**, the present work proposes the use of diagrammatic rules (see [16]) to construct an adjoint network and propagate the complex output error through it in order to obtain the weight adjustment, with two different **RCVNN** topologies considered, each with different activation functions avoiding singularities.

The based on optimization **LM** learning techniques is used for nonlinear oscillatory plant identification and oscillation suppression by means of a direct integral term (I-term) adaptive neural control using **RCVNN**. Lastly, some comparative simulation results of **RCVNN** identification and control of flexible-joint robot are given and discussed, and a validation stage is presented in order to confirm the good quality of the control scheme and the proposed learning algorithms.

I. S. Baruch is a corresponding author of the paper. He is with the Department of Automatic Control, CONVESTAV-IPN, D.F, Ave. IPN No 2508, Colonia San Pedro Zacatenco, A.P. 14-740, 07360 Mexico City, Mexico, Phone: (+52-55)5747-3800/42-29, Fax (+52-55)5747-7089, E-mail: baruch@ctrl.cinvestav.mx

Victor Arellano Quintana is Master in Science graduated in the Department of Automatic Control, CINVESTAV-IPN, D.F, Mexico. E-mail: varellano@ctrl.cinvestav.mx

E. P. Reynaud is a Master in Science student in the Department of Automatic Control, CINVESTAV-IPN, D.F, Mexico. E-mail: eperez@ctrl.cinvestav.mx

II. Topology and LM Learning of RCVNN

The general **RCVNN** topology in consideration is an extension of the real-valued Recurrent Neural Network topology, given in [8]. The considered **RCVNN** topology has real-valued input $U(k)$ and output $Y(k)$ signals, complex valued internal state $X(k)$ and hidden state $Z(k)$ vectors, and complex valued J, B, C weight matrices. It is defined as follows:

$$X(k+1) = JX(k) + BU(k) \quad (1)$$

$$J = \text{diag}(J_i), \quad |J_i| < 1, \quad i = 1, \ldots, N \quad (2)$$

$$E(k) = Y_P(k) - Y(k) \quad (3)$$

$$Z(k) = \Gamma[X(k)] \quad (4)$$

$$V(k) = CZ(k) \quad (5)$$

$$Y(k) = \Phi[V(k)] \quad (6)$$

The vectors and matrices dimensions of the **RCVNN** topology are given as follows: $J \in \mathbb{C}^{n\times n}$ is the feedback weight matrix, $B \in \mathbb{C}^{n\times m}$ is the input weight matrix, $C \in \mathbb{C}^{p\times n}$ is the output weight matrix, $X \in \mathbb{C}^n$ is the internal state vector, $Z \in \mathbb{C}^n$ is the hidden state vector, $U \in \mathbb{R}^m$ is the network input, $Y \in \mathbb{R}^L$ is the network output, $\Gamma[\cdot]$ a complex-valued activation function, and $\Phi[\cdot]$ a real-valued activation function $f(\cdot) = \tanh(\cdot)$; n, m, L are the number of internal states, inputs and outputs respectively.. The inequality in (2) is a stability preserving condition, imposed on the diagonal blocks of the matrix J. This condition is imposed also to all the weights of the matrices B, C.

We consider two particular **RCVNN** with different activation functions. In the same way as in the real-valued case, [8], we apply complex-valued diagrammatic rules so to derive an adjoint network for each case.

The performance index to be minimized is given by:

$$\zeta(k) = \frac{1}{2}\sum_j [E_j(k)]^2, \qquad \zeta = \frac{1}{N_e}\sum_k \zeta(k) \quad (7)$$

The instantaneous Means Squared Error (**MSE**) $\zeta(k)$ is used in real-time applications, while the total **MSE** ζ is used for one epoch N_e in off-line applications.

A. Topology and LM Learning of RCVNN with First Type Activation Function

The first type activation function is defined as follows:

$$f(z) = \tanh(z), \quad z \in \mathbb{C} \setminus \left\{ \begin{array}{c} z : z = 0 \pm \dfrac{2p-1}{2}\pi i, \\ \forall p \in \mathbb{N} \end{array} \right\} \quad (8)$$

This activation function has singularities in some points of the complex domain and because of this, we avoided them. The topology of **RCVNN** using this activation function is given on Fig.1.

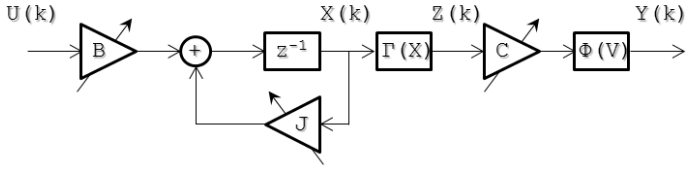


Fig. 1 Topology of the first type RCVNN

Applying the diagrammatic rules, [16], to the **RCVNN** topology, we obtain the adjoint **RCVNN** model, given on Fig.2. The adjoint **RCVNN** is used for the backward propagation of the output error signals of the BP algorithm.

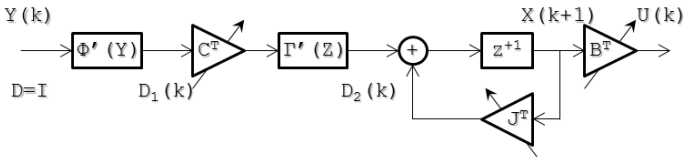


Fig. 2 Adjoint topology of the first type RCVNN

The complex-valued Levenberg-Marquardt (**CVLM**) algorithm for any weight vector W is described by the following equation:

$$W(k+1) = W(k) + P(k)\cdot DY[W(k)]\cdot E(k), \quad |W_j| < W_0 \quad (9)$$

Where: W_0 is a restricted region for the weight W_j. The gradient terms for the complex-valued network with the first type activation function are described by the following equations:

$$D_1(k) = \Phi'[Y(k)]\cdot D \quad (10)$$

$$D_2(k) = \Gamma'[Z(k)]\cdot C^*\cdot D_1(k) \quad (11)$$

$$DY[C(k)] := \partial Y(k)/\partial C(k) = D_1(k)\cdot Z^*(k) \quad (12)$$

$$DY[J(k)] := \partial Y(k)/\partial J(k) = D_2(k)\cdot X^*(k) \quad (13)$$

$$DY[B(k)] := \partial Y(k)/\partial B(k) = D_2(k)\cdot U^*(k) \quad (14)$$

Where the $(*)$ superscript denotes a complex conjugate and transposed vector, $D = I$ is a real-valued identity matrix input for the adjoint topology. The matrix P is computed recursively using the following equation:

$$P(k) = \alpha^{-1}[P(k-1) - P(k-1)\cdot\Omega_{W(k)}\cdot S_{W(k)}^{-1}\cdots \cdot\Omega_{W(k)}^*\cdot P(k-1)] \quad (15)$$

Where the matrices Ω_W and S_W are given by:

$$\Omega_{W(k)}^* = \begin{bmatrix} DY^*[W(k)] \\ 0 \quad \cdots \quad 1 \quad \cdots \quad 0 \end{bmatrix} \quad (16)$$

$$S_{W(k)} = \alpha\Lambda + \Omega_{W(k)}^*\cdot P(k-1)\cdot\Omega_{W(k)} \quad (17)$$

$$\Lambda^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix} \quad (18)$$

Again, the matrices P and S_W have dimensions $(N_W \times N_W)$ and (2×2) respectively, where N_W is the number of weights in the vector W. The matrix Ω_W has dimension $(N_W \times 2)$, the second row of Ω_W^* consists of

$(N_W - 1)$ zeroes and a unit element in the i-th position computed by:

$$i = k \cdot \mathrm{mod}(N_W) + 1 \tag{19}$$

The parameters for the algorithm should be restricted as follows:

$$\begin{aligned} 10^{-6} \le \rho \le 10^{-4}, \qquad 0.97 \le \alpha \le 1.00 \\ 10^{3} \le P(0) \le 10^{6} \end{aligned} \tag{20}$$

B. Topology and LM Learning of RCVNN with Second Type Activation Function

The second type activation function ([11], [14]) does not have any singularity points and it is defined by the following equation:

$$f(z) = \tanh\left[\mathrm{Re}(z)\right] + i \cdot \tanh\left[\mathrm{Im}(z)\right], z \in \mathbb{C} \tag{21}$$

This activation function doesn't have any singularities, so it doesn't need any restrictions in its domain as it did the first activation function. The block diagram of the **CVRNN** that uses this activation function is shown in Fig.3.

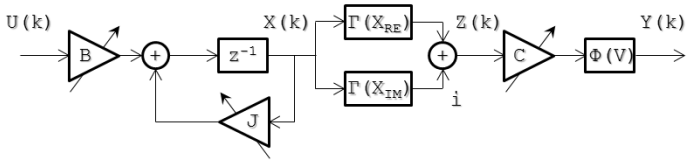


Fig.3 Block-diagram representation for the CVRNN with second type activation function

The description of the **CVRNN** with the constructed activation function (21) is given by the following equations:

$$X(k+1) = JX(k) + BU(k) \tag{22}$$

$$Z(k) = \Gamma[X_{\mathrm{Re}}(k)] + i\Gamma[X_{\mathrm{Im}}(k)] \tag{23}$$

$$V(k) = CZ(k) \tag{24}$$

$$Y(k) = \Phi[V(k)] \tag{25}$$

The dimensions and domains of each vector and matrix in this **CVRNN** are the same as in the previous network. Applying the complex-valued diagrammatic rules we obtain the adjoint network, shown in Fig.4.

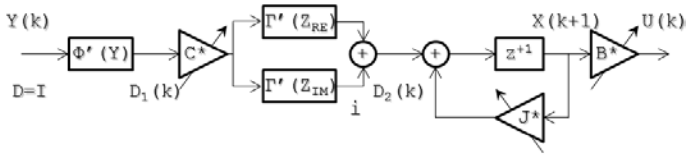


Fig.4 Block-diagram representation of the adjoint network for the CVRNN with second type activation function

From this adjoint network, we derive the gradient terms needed for the **CVLM** learning algorithm, which are described by the following equations:

$$D_1(k) = \Phi'[Y(k)] \cdot D \tag{26}$$

$$\begin{aligned} D_2(k) = (\Gamma'[Z_{\mathrm{Re}}(k)] \cdot \mathrm{Re}(C^*) \cdots \\ + \Gamma'[Z_{\mathrm{Im}}(k)] \cdot \mathrm{Im}(C^*)) \cdot D_1(k) \end{aligned} \tag{27}$$

$$DY[C(k)] := \partial Y(k) / \partial C(k) = D_1(k) \cdot Z^*(k) \tag{28}$$

$$DY[J(k)] := \partial Y(k) / \partial J(k) = D_2(k) \cdot X^*(k) \tag{29}$$

$$DY[B(k)] := \partial Y(k) / \partial B(k) = D_2(k) \cdot U^*(k) \tag{30}$$

Where: $D = I$ is a real-valued identity matrix input for the adjoint topology. Then we apply the **CVLM** equations given by (15)-(19) with parameters restricted by (20).

III. Complex-Valued Neural Identification and Control of Nonlinear Plants

A. Plant Identification

The application of the given models of **RCVNN** for the identification of nonlinear oscillatory plants is illustrated by Fig.5. The plant here is a flexible joint arm. The input signal of the plant is the same as the input signal of the **RCVNN** model. Here the desired target vector is the output of the plant and the identification objective is to adjust the complex weight parameters of the **RCVNN** in a way that the **RCVNN** output follows the plant output with minimum Mean Squared Error (**MSE)**. The **RCVNN** training is then validated by a generalization step, where an unknown input signal is used and a **MSE** is computed with fixed **RCVNN** weights.

The input plant signals used for system identification and generalization are chosen as a sum of sine functions with different frequency and amplitude. The input signal of **RCVNN** model and the output signal of the plant are discretized in other to perform the complex **LM** algorithm of learning and the **RCVNN** model generalization (see Fig.5).

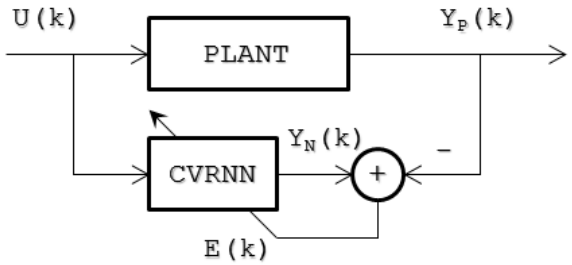


Fig. 5 Block-diagram of RCVNN plant identification scheme

B. Direct Feedforward Adaptive Neural Control with State Feedback

For the first adaptive neural controller, we use the control scheme represented in the diagram shown in Fig.6 [17].

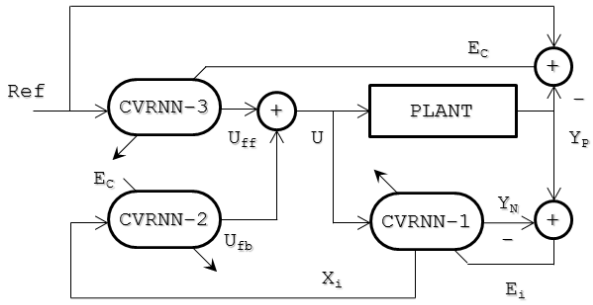


Fig.6 Block-diagram of the feedforward adaptive neural control with state feedback

In this control scheme we use three different **CVRNN**, each performing a specific task:

- *CVRNN-1*: Is used as a plant model identifier. It has the control signal U that is fed to the plant as its input and produces an identification output Y_N, which is compared to the plant output Y_P to produce the identification error signal used for the training of its own weight parameters. The identification error, given by the equation (3), is specified as follows.

$$E_i(k) = Y_P(k) - Y_N(k) \tag{31}$$

- *CVRNN-2*: This neural network is fed by the internal state vector X_i of the identification network *CVRNN-1* to produce a state feedback control signal U_{fb}. This network is trained with the control error signal E_C, which is obtained comparing the reference signal R with the output of the plant. The control error is given by the following equation:

$$E_C(k) = R(k) - Y_P(k) \tag{32}$$

- *CVRNN-3*: This neural network is trained with the control error signal (32) so it converges to the inverse model of the plant. The network is fed by the reference signal R to produce a feedforward control signal U_{ff}.

Except for the dimensions of each of the **CVRNN**, the three **RNNs** used the same topology and learning. The control signal is the sum of the feedforward and state feedback control signals, and is described by the following equation:

$$U_1(k) = U_{ff}(k) + U_{fb}(k) \tag{33}$$

C. Direct Feedforward Adaptive Neural Control with Integral Term and State Feedback

This control scheme is described by the block-diagram shown in Fig.7 [17]. We used three **CVRNN** for this control scheme: one as a plant model identifier, one as a state feedback controller, and one as a direct feedforward controller.

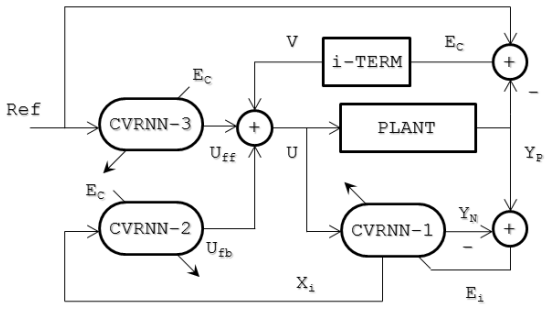


Fig.7 Block-diagram of the feedforward adaptive neural controller with integral term and state feedback

We add an integral term V to the control signal to eliminate the steady-state error presented in the plant output. This integral term is given by the following equation:

$$V(k+1) = V(k) + \tau \cdot K_i \cdot E_C(k) \tag{34}$$

Where: K_i is the gain of the integral-action and τ is the sampling time. The control signal for this scheme is the sum of three terms and is described by the following equation:

$$U_2(k) = U_{ff}(k) + U_{fb}(k) + \mathrm{V(k)} \tag{35}$$

This control scheme forces the plant to follow the reference signal with zero steady-state error. The stability of the whole system, for both control schemes, is assured by the restriction in the weight parameters of the state matrix J of the **CVRNN** and the boundedness of its activation functions.

IV. Simulation and Results

A. Plant Description

The plant subject to system identification in this work is an idealized nonlinear plant model of a flexible-joint robot arm, illustrated in Fig.8. The flexibility of the joint is caused by a harmonic drive, which is a type of gear mechanism with high torque transmission, low backlash and compact size.

The robot joint model consists of an actuator connected to a load through a torsional spring, which represents the joint flexibility. We consider the motor torque and the angular position of the link as the input signal $u(t)$ and the output

signal $y(t)$ respectively, making this a **SISO** system. The equations that describe the motion of the flexible joint are as:

$$J_l\ddot{\theta}_l + B_l\dot{\theta}_l + Mgl\sin\theta_l + k(\theta_l - \theta_m) = 0 \quad (36)$$

$$J_m\ddot{\theta}_m + B_m\dot{\theta}_m - k(\theta_l - \theta_m) = \mathrm{u} \quad (37)$$

Where: J_l, J_m are the link and motor inertial coefficients, B_l, B_m are the link and motor damping coefficients, k is the torsion stiffness coefficient of the harmonic drive gear, M is the mass of the link, L is the length between the shaft and the center of mass of the link, θ_l, θ_m are the angular positions of the link and the rotor of the motor respectively.

The input and output signals are discretized with a sampling period (τ) in order to use a discrete neural network approach for its identification. This is an oscillatory system, described by two nonlinear second order differential equations.

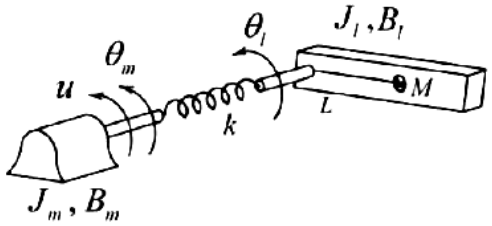


Fig.8 Flexible-joint robotic arm

B. Plant Identification

We test the **CVLM** learning algorithm applied to a **CVRNN** with the second activation function, for nonlinear oscillatory plant identification with a simulation using MATLAB. The simulation has two stages: in the learning stage, weights are adjusted until convergence to a steady value and the **CVRNN** output matches the output of the plant; in the generalization stage, we fix the weight parameters and apply a different input for the plant and the **CVRNN**, to validate its learning by comparing both outputs. Next, we make a comparison between the **CVBP** and the **CVLM** learning algorithms.

As a comparison measure, we use the total **MSE** for learning and generalization stages. This section describes the simulation settings used and the results obtained. The input signals used on the learning stage $u_L(t)$ and generalization stage $u_G(t)$ are given by:

$$u_L(t) = \sin\left(\tfrac{1}{10}t\right) + 0.5\sin\left(\tfrac{1}{25}t\right) \quad (38)$$

$$u_G(t) = 0.5\sin\left(\tfrac{1}{10}t\right) + 0.8\sin\left(\tfrac{1}{20}t\right) \quad (39)$$

For the **CVRNN** we used dimensions $n = 3, m = 1, L = 1$, with initial conditions of the internal states vector $X(0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$, random initial conditions for each weight parameter in the interval $[-0.5,0.5]$; simulation time of $T = 500s$ and sampling time $\tau = 0.01$. For the **CVBP** algorithm we used the parameters: $\eta = 0.05$ and $\alpha = 0.005$. For the **CVLM** algorithm we used the parameters: $\alpha = 0.9775$, $\rho = 1\times10^{-4}$, $P_J(0) = 1\times10^{6}$, $P_B(0) = 1\times10^{4}$ and $P_C(0) = 1\times10^{5}$. For both cases, the second activation function is used.

For the **CVBP** algorithm, Fig.9, a) shows the plant and the neural network outputs, and b) the total **MSE** for the learning stage. Fig.10 a), b) shows the same signals for the generalization stage.

For the **CVLM** algorithm, Fig.11, a) shows the plant and the neural network outputs, and b) the total **MSE** for the learning stage. Fig.12, a), b) shows the same signals for the generalization stage.

For both algorithms, we observe a fast convergence of the neural network output to the plant output, while the **MSE** shows a decreasing behavior, for the learning stage. For the generalization stage, we observe a good performance of the output of the network and the **MSE** in general.

Table I shows the final **MSE** of the simulations for the **CVBP** and **CVLM** learning algorithms, for both learning and generalization stages.

Table I. Final MSE of both learning algorithms for the learning and generalization stages

	CVBP	**CVLM**
Learning	6.11×10^{-4}	0.18×10^{-4}
Generalization	25.44×10^{-4}	20.84×10^{-4}

We observed from the total **MSE** for both stages that the **CVLM** learning algorithm has a better performance compared to the **CVBP** learning algorithm.

We also observe that the **CVLM** learning algorithm tends to be more sensible to the initial conditions of its weight parameters. Nevertheless, this sensibility doesn't affect the performance and convergence of the learning stage for the **CVRNN**.

C. Plant Adaptive Neural Control Simulation and Results

We tested the **CVLM** learning algorithm applied to both adaptive neural control schemes proposed in [17] for a nonlinear, oscillatory mechanical plant using MATLAB.

The reference signal used for both simulations is given by the following equation:

$$R(t) = 0.5\sin\left(\tfrac{1}{25}t\right) + 0.3\sin\left(\tfrac{1}{15}t\right) \quad (40)$$

For both control schemes, the *CVRNN-1,3* have dimensions $n_{1,3} = 3, m_{1,3} = 1, L_{1,3} = 1$, while the *CVRNN-2* has dimensions $n_2 = 3, m_2 = 3, L_2 = 1$ with initial conditions of the internal states vector $X(0) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$, random initial conditions for each weight parameter in the interval $[-0.5,0.5]$; simulation time of $T = 500s$ and sampling time $\tau = 0.01$.

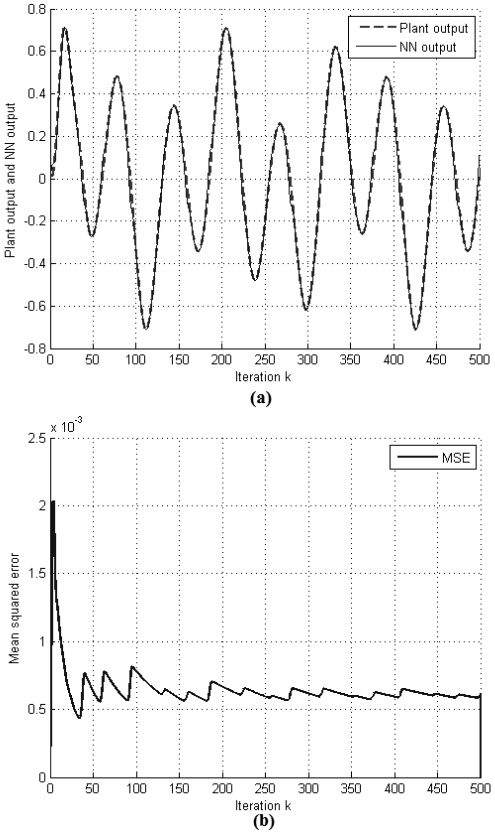


Fig.9 CVBP learning stage, a) Plant output and NN output signals, b) total MSE

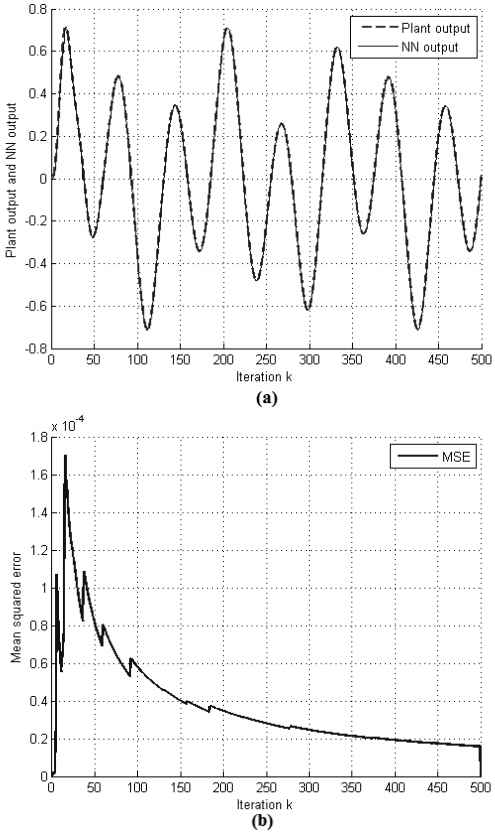


Fig.11 CVLM learning stage, a) Plant output and NN output signals, b) total MSE

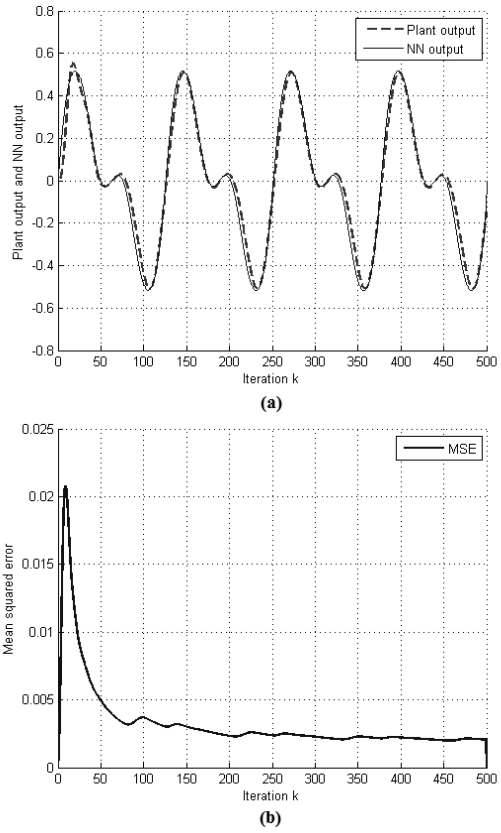


Fig.10 CVBP generalization stage, b) Plant output and NN output signal, b) total MSE

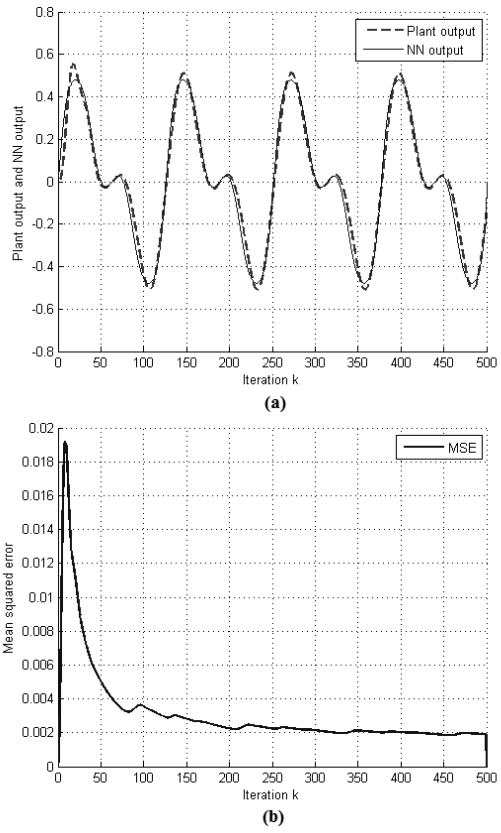


Fig.12 CVLM generalization stage, (b) Plant output and NN output signal, (b) total MSE

For the **CVBP** algorithm we used the parameters: $\eta = 0.05$ and $\alpha = 0.005$. For the **CVLM** algorithm we used the parameters: $P_J(0) = P_B(0) = P_C(0) = 1\times10^2$, $\alpha = 0.98$, and $\rho = 1\times10^{-4}$. For both cases, the second activation function is used. For the control scheme with integral term, an integral gain of $K_i = 0.05$ is used.

For the **CVBP** algorithm, Fig.13, a) shows the plant output and reference signal, and b) the instantaneous **MSE** for the first control scheme. Fig.14 a), b) shows the same signals for the second control scheme with integral term.

For the **CVLM** algorithm, Fig.15, a) shows the plant output and reference signal, and b) the instantaneous **MSE** for the first control scheme. Fig.16 a), b) shows the same signals for the second control scheme with integral term.

From these figures we observe that, in general, both control schemes have good quality in driving the mechanical plant to the reference signal, while the control scheme with the integral term shows a better performance than the other. Furthermore, we observe that the second control scheme has a much better performance than the first one; this can be observed from the value of the final **MSE** for both control schemes.

Table II shows the final **MSE** of the simulations for the **CVBP** and **CVLM** learning algorithms, for both adaptive neural control schemes.

Table II. Final MSE of both learning algorithms and both control schemes

	CVBP	**CVLM**
1st Scheme	81.10×10^{-4}	61.90×10^{-4}
2nd Scheme	1.19×10^{-4}	0.91×10^{-4}

We observed from the total **MSE** for both schemes that the **CVLM** learning algorithm has a better performance compared to the **CVBP** learning algorithm.

We also observe that adding the integral term of the error changes drastically the performance of the adaptive controller, eliminating any tracking error.

V. CONCLUSIONS

In this work we used an array of Complex-Valued Recurrent Neural Networks to construct a control scheme applied to a nonlinear, oscillatory mechanical plant. These neural networks were trained using a complex-valued version of the Recursive Levenberg-Marquardt algorithm, where the local gradient terms were easily obtained from an adjoint topology, instead of using an algebraic approach. The neural network used for plant identification worked with great performance, which gave us the possibility of using its internal states to produce a state feedback control signal. In general, the control schemes present an acceptable performance, and were improved by introducing an integral term of the error. The comparative simulation results confirm the good quality of the LM neural identification and control scheme.

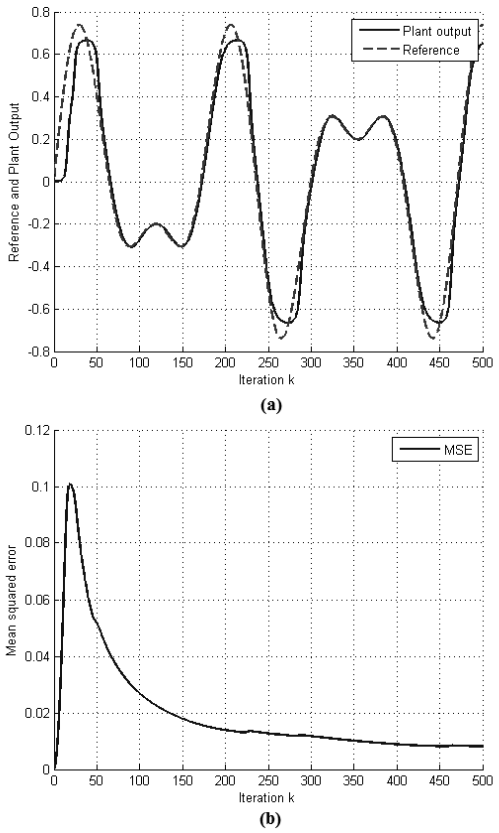


Fig.13 CVBP applied to the first control scheme, a) Plant output and reference signals, b) total MSE

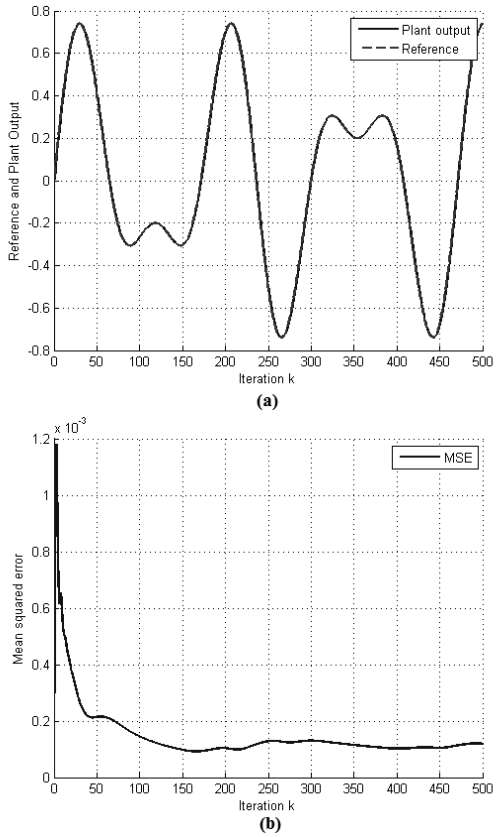
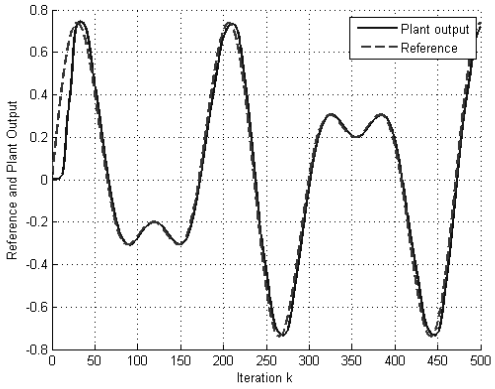
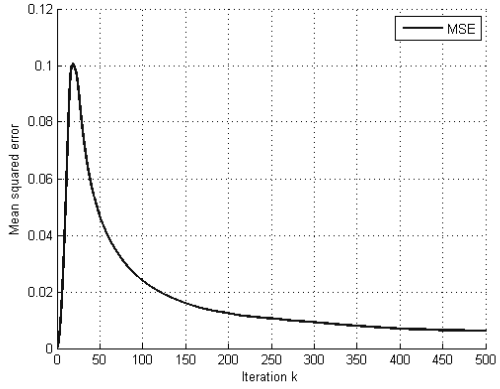


Fig.14 CVBP applied to the second control scheme, a) Plant output and reference signals, b) total MSE

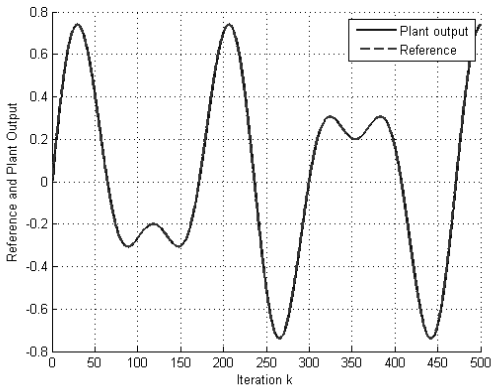


(a)

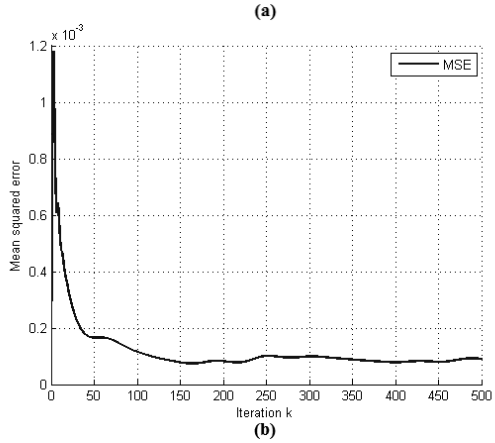


(b)

Fig.15 CVLM applied to the first control scheme, a) Plant output and reference signals, b) total MSE



(a)



(b)

Fig.16 CVLM applied to the second control scheme, a) Plant output and reference signals, b) total MSE

ACKNOWLEDGMENT

Master in Science graduate Victor Arellano Quintana and Master in Science student Edmundo Pérez Reynaud would like to thank CONACyT, Mexico, for the student grant received during his studies at the Department of Automatic Control, CINVESTAV-IPN, Mexico.

REFERENCES

1. R. A. Amer, G. A. Morsy, H. A. Yassin, SCG Stability Enhancement Using STATCOM Based-ANN Controller. WSEAS Transactions On Systems And Control 6 (9), 325-338 (Sept. 2011)
2. Youpeng Zhang, Hongsheng Su, Turbo-Generator Vibration Fault Diagnosis Based On PSO-BP Neural Networks, WSEAS Transactions On Systems And Control 5 (1), 37-47 (Jan.. 2010)
3. Ryad Zemouri, Rafael Gouriveau, Paul Ciprian Patic, Combining ARecurrent Neural Network And A PID Controller For Prognostic Purpose: AWay To Improve The Accuracy Of redictions, WSEAS Transactions on Systems and Control 5 (5) 353-371(May. 2010)
4. Da Lin, Xingyuan Wang, Fuzhong Nian, Yonglei Zhang. Dynamic Fuzzy Neural Networks Modeling And Adaptive BacksteppingTracking Control Of Uncertain Chaotic Systems, Neurocomputing, 73 (11-18), 2873-2881 (2010)
5. Hao Zhang, Xingyuan Wang, Xiaohui Lin, Chongxin Liu, Stability And Synchronization For Discrete-Time Complex-Valued Neural Networks With Time Varying Delays, PLOS ONE, 9 (4)_e93838.pdf, (2014)
6. S. Haykin, Neural Networks: A Comprehensive Foundations (Macmillan College, New York, 1994)
7. K.S. Narendra, K. Parthasarasi, Identifications And Control Of Dynamic Systems Using Neural Networks, IEEE Transactions of Neural Networks, 1 (1) 4-27 (1990)
8. Baruch, I.S., Mariaca-Gaspar, C.R.: A Levenberg-Marquardt Learning Applied For Recurrent Neural Identification And Control Of A Wastewater Treatment Bioprocess. International Journal of Intelligent Systems, ISSN 0884-8173, 24, 1094-1114 (2009)
9. A. Hirose, Complex-Valued Neural Networks, (2nd ed., S. i. C. Intelligence, Ed. Springer Verlag, 2012, vol. 400)
10. A. Hirose, Motion Controls Using Complex-Valued Networks With Feedback Loops, in: Proc. IEEE International Conference on Neural Networks, (vol. 1, San Francisco, CA, 1993, pp. 156-161).
11. A. Minin, Y. Chistyakov, E. Kholodova, H. G. Zimmermann, A. Knoll, Complex-Valued Open Recurrent Neural Network For Power Transformer Modeling, International Journal of Applied Mathematics and Informatics, 6 (1), 41-48 (2012)
12. H. Leung, S. Haykin, The Complex Back-PropagationAlgorithm, IEEE Transactions on Signal Processing, 39 (9), 2101-2104 (1991)
13. C. Woo, D. S. Hong, Adaptive Equalization Using The Complex Back-PropagationAlgorithm, in: Proc. of IEEE International Conference on Neural Networks, (vol. 4, Washington DC, 2136-2141, (1996)
14. N. Miklos, B. Salik, Neural Networks With Complex Activations And Connection Weights, Complex Systems 8, 115-126 (1994)
15. F. Nava, I. Baruch, A. Poznyak, B. Nenkova, Stability Proofs Of Advanced Recurrent Neural Networks Topology And Learning, Comptes Rendus (Proceedings of the Bulgarian Academy of Sciences) 57 (1), 27-32 (2004)
16. E. Wan, F. Beaufays, Diagrammatic Method For Deriving And Relating Temporal Neural Networks Algorithms, Neural Computation 8, 182-201(1996)
17. I. S. Baruch, E. P. Reynaud. "Control of Nonlinear Dynamical Systems Using Levenberg-Marquardt Learning Algorithm for Recurrent Complex-Valued Neural Networks". In *Proc. of the 19th International Conference on Systems, part of CSCC '15,* Recent Advances in Systems, Zakynthos Island, Greece. ISSN: 1790-5117, ISBN: 978-1-61804-321-4. p.398-403, (2015).