A Comparative Study of Soft Computing Methodologies in Identification of Robotic Manipulators

M. Önder Efe , Okyay Kaynak
Mechatronics Research And Application Center,
Bogaziçi University, Bebek, Istanbul, Turkey

Abstract

This paper investigates the identification of nonlinear systems by utilizing soft-computing approaches. As the identification methods, Feedforward Neural Network architecture (FNN), Radial Basis Function Neural Networks (RBFNN), Runge-Kutta Neural Networks (RKNN) and Adaptive Neuro Fuzzy Inference Systems (ANFIS) based identification mechanisms are studied and their performances are comparatively evaluated on a two degrees of freedom direct drive robotic manipulator.

1 Introduction

Identification of systems has drawn a great interest because of the increasing needs in estimating the behavior of a system with partially known dynamics. Especially in the areas of control, pattern recognition and even in the realm of stock markets the system of interest needs to be known to some extent. A common property of real life systems is the fact that they have multiple variables, some of which are subjected to stochastic disturbances. Since the a system may have a complicated dynamic behavior, the varying environmental changes make the identification process much more difficult than the cases in which those changes are modeled deterministically. In the latter, the identification is performed at the cost of losing the reliability and preciseness. Therefore, deciding on what the future behavior of a system will be and performing an adaptive estimation is a formidable problem for real life systems.

Soft-computing is a practical alternative for solving computationally complex and mathematically intractable problems. The reason that lies behind this understanding is the fact that through the use of soft-computing methodologies, one can easily combine the natural system dynamics and an intelligent machine. In this respect, the intelligence stems from an expert's knowledge and, massively parallel and, adaptive data processing architecture of the computationally intelligent approach. The most popular members of the soft-computing methodologies are the neural networks and fuzzy logic. Neural networks provide the mathematical power of the brain whereas the fuzzy logic based mechanisms employ the verbal power. The latter allows the linguistic manipulation of input-state-output data. The most interesting applications offer an appropriate combination of these two approaches resulting in a hybrid system that both operates on linguistic descriptions of the variables and the numeric values through a parallel and fault tolerant architecture.

The mapping properties of artificial neural networks have been analyzed by many researchers. Hornik [1], and Funahashi [2] have shown that as long as the hidden layer comprises sufficient number of nonlinear neurons, a function can be realized with a desired degree of accuracy. This proof is followed by the study of Narendra and Parthasarathy [3]. In their pioneering work, they have debated how useful artificial neural networks can be for identification and control purposes. Their paper dwells on the realization of an unknown nonlinearity by artificial neural networks. The training is performed by the error backpropagation algorithm [4]. On the other hand, a novel approach has been presented in [5] where the neural network realizes the behavior of a set of ordinary differential equations utilizing the Runge-Kutta algorithm. The method is proved to be successful in predicting the future behavior accurately. In [6], Radial Basis Function Neural Networks are explained with their functional equivalence to Fuzzy Inference Systems (FIS). In the same reference the details of Adaptive Neuro Fuzzy Inference Systems (ANFIS) structure can be found, proposed as a core neuro-fuzzy model that can incorporate human expertise as well as adapt itself through repeated learning. This architecture has revealed a high performance in many applications. This paper considers the ANFIS structure for the identification procedure of a direct drive robotic manipulator.

The identification procedure followed in all of the methods considered is as depicted in Fig. 1. An excitation input is applied to both the robot system and the identifier. The output error is then used to update the parameters of the identifier. The excitation input cannot be selected randomly for any real system. Therefore, in the simulations the manipulator is kept under an external control loop while the identifier performance is being tested.

In the next section the system under observation is introduced, the following section dwells on FNN based identification schemes. Next, RBFNN approach is derived. The fifth section describes the application of Runge-Kutta neural network methodology for the identification of robotic manipulators. In the sixth section ANFIS structure is elaborated. Lastly the conclusions are presented.

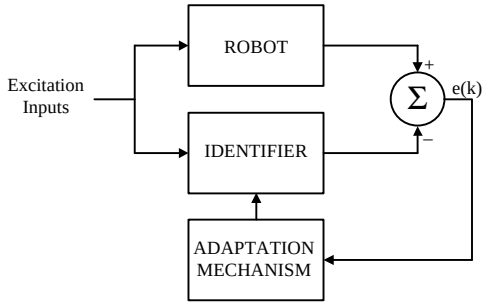


Fig. 1. Identification of a robotic manipulator

Table 1. Manipulator Parameters

Motor 1 Rotor Inertia	0.267	I1	Payload Mass	0.00	Mp
Arm 1 Inertia	0.334	I2	Arm 1 length	0.359	L1
Motor 2 Rotor Inertia	0.0075	I3	Arm 2 length	0.24	L2
Motor2 Stator Inertia	0.040	I3C	Arm 1 CG	0.136	L3
Arm 2 inertia	0.063	I4	Arm 2 CG	0.102	L4
Payload Inertia	0.000	IP	Axis 1 Friction	5.3	F1
Motor 1 Mass	73.0	M1	Axis 2 Friction	1.1	F2
Arm 1 Mass	9.78	M2	Torque Limit 1	245.0	
Motor 2 Mass	14.0	M3	Torque Limit 2	39.2	
Arm 2_Mass	4.45	M4			

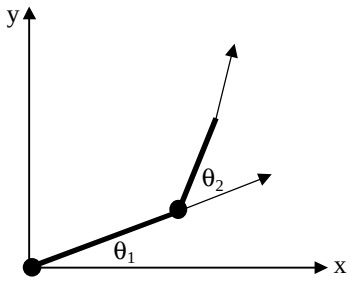


Fig. 2**.** Architecture of a 2-DOF planar manipulator

2 Two dof Direct Drive Robotic Manipulator Dynamics

A robotic manipulator is a proper candidate for the evaluation of the performance of identification mechanisms stated in the previous section. The main reason for this is the fact that the dynamics are highly involved, being comprised of coupled nonlinear differential equations. The general form of the manipulator dynamics is given by (1) and the nominal values of the parameters are summarized in Table 1 in standard units. The architecture of the manipulator is illustrated in Fig. 2.

$$M(\mathbf{q})\ddot{\mathbf{q}} + V(\mathbf{q},\dot{\mathbf{q}}) = \tau(t) - f \tag{1}$$

By assuming the angular positions and angular velocities as state variables of the system, total of four 1st order differential equations are obtained. The state varying inertia matrix and coriolis terms are given in (2).

$$M(\mathbf{q}) = \begin{bmatrix} p_1 + 2p_3\cos(q_2) & p_2 + 2p_3\cos(q_2) \\ p_2 + 2p_3\cos(q_2) & p_2 \end{bmatrix}$$

$$V(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} -\dot{q}_2(2\dot{q}_1 + \dot{q}_2)p_3\sin(q_2) \\ \dot{q}_1^2 p_3\sin(q_2) \end{bmatrix} \tag{2}$$

where $p_1 = 2.0857$, $p_2 = 0.1168$, $p_3 = 0.1630$.

3 Feedforward Neural Networks (FNN) for Identification

Identification procedure, in the most general sense, entails a matching between the system outputs and an identifier output and artificial neural networks, due to their ability to act as universal approximators, can very effectively be used for this purpose [1-2]. Narendra and Parthasarathy [3] have reported an extensive study on the use of these networks for identification and control purposes. In this paper, based on the diagram sketched in Fig. 1, the cost function given in (3) is minimized by propagating the output error back through the neural network, the architecture of which is depicted in Fig. 3. The update equations will not be derived here, for further details, [4] should be reviewed.

$$J = \frac{1}{2}\sum_{i=1}^{N}(d_i^p - y_i^p)^2 \tag{3}$$

$$\delta_j^{k+1,p} = (d_j^p - y_j^{k+1,p})\,\Psi'(S_j^{k+1,p}) \tag{4}$$

$$\delta_j^{k+1,p} = \left(\sum_{h=1}^{\#neurons_{k+2}} \delta_h^{k+2,p}\, w_{jh}^{k+1}\right)\Psi'(S_j^{k+1,p}) \tag{5}$$

In (3), y_i^p denotes the ith entry of pth pattern in neural network response, d_i^p denotes the ith entry of pth target vector. Equations (4) and (5) give the delta values for the output layer and hidden layer neurons respectively.

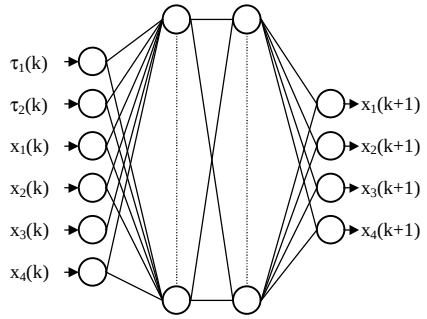


Fig. 3**.** FNN architecture.

In (4) and (5), S_j denotes the net summation of the j^{th} neuron in the $(k+1)^{th}$ layer, ψ denotes the nonlinear activation function attached to each neuron in the hidden layer. After the evaluation of the delta values during the backward pass, the weight update rule given in (6) is applied for each training pair.

$$\Delta w_{ij}^{k} = \eta \delta_j^{k+1,p} o_i^{k,p} \tag{6}$$

Extensive simulation studies were carried out with this structure. However, due to limitations of space, these are not presented. Only those obtained with RKNN and ANFIS structures are given at the end of the paper.

4 Radial Basis Function Neural Networks (RBFNN) for System Identification

In most of the literature, RBFNN are considered as a smooth transition between fuzzy logic and neural networks. Structurally, RBFNN are composed of receptive units (neurons) which act as the operators providing the information about the class to which the input signal belongs. If the aggregation method, number of receptive units in the hidden layer and the constant terms are equal to those of a Fuzzy Inference System (FIS), then there exists a functional equivalence between RBFNN and FIS [6]. In Fig. 5, a RBFNN structure is illustrated. Each neuron in the hidden layer provides a degree of membership value for the input pattern with respect to the basis vector of the receptive unit itself. The output layer is comprised of linear combiners. Neural network interpretation makes RBFNN useful in incorporating the mathematical tractability, especially in the sense of propagating the error back through the network, while the Fuzzy System interpretation enables the incorporation of the expert knowledge into the identification procedure. The latter may be crucial in assigning the initial value of the RBFNN parameter vector to a vector that is close to the optimal one. This results in faster convergence in parameter space if the system dynamics is known. In this paper, the parameter vector is randomly initialized.

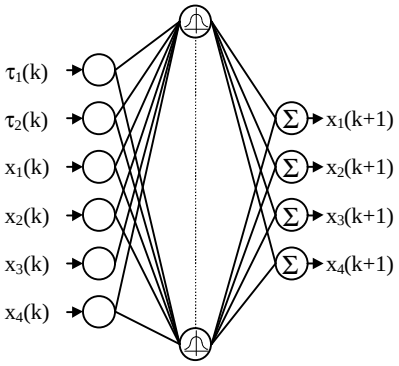


Fig. 5. RBFNN architecture.

In this approach, 12 hidden neurons are used. As is given by (7), each neuron output is evaluated from the multiplication of the outputs of individual Gaussians corresponding to each input. The overall response is evaluated through multiplication of hidden layer output vector by a matrix of appropriate dimensions. In fuzzy terms, this is equivalent to say that, product inference rule is used with weighted sum defuzzifier.
If the network output and hidden layer output vectors are denoted by $\underline{y}$ and $\underline{o}$ respectively, the activation level of i^{th} receptive unit (or firing strength of each rule) and the overall realization performed by the network can be given by (7) and (8) respectively.

$$o_i = \prod_{j=1}^{\#inputs} \exp\left(\frac{(x_j - c_{ij})^2}{w_{ij}^2}\right) \tag{7}$$

$$\underline{y} = W_{\#outputs\ x\ \#hidden\ neurons}\,\underline{o} \tag{8}$$

According to the error backpropagation rule, the parameter update law can be summarized as follows:

$$\underline{\delta}_{output} = \underline{d} - \underline{y} \tag{9}$$

$$\underline{\delta}_{hidden} = W^T \underline{\delta}_{output} \tag{10}$$

$$W_{ij}(k+1) = W_{ij}(k) + \eta \delta_{output\ i} o_j \tag{11}$$

$$c_{ij}(k+1) = c_{ij}(k) + \eta \delta_{hidden\ i} o_j 2\left(\frac{x_j - c_{ij}(k)}{w_{ij}(k)}\right)^2 \tag{12}$$

$$w_{ij}(k+1) = w_{ij}(k) + \eta \delta_{hidden\ i} o_j 2\frac{(x_j - c_{ij}(k))^2}{w_{ij}(k)^3} \tag{13}$$

5 Runge-Kutta Neural Networks (RKNN) for System Identification

Runge-Kutta method is a powerful way of solving the behavior of a dynamic system if the system is characterized by ordinary differential equations. In [5], the proposed method is applied to several problems. Mainly, the method is observed to be successful in estimating the system states in long run. It should be emphasized that the neural network architecture realizes the changing rates of the system states instead of the $[\underline{\mathbf{x}}(k), \underline{\tau}(k)] \Rightarrow [\underline{\mathbf{x}}(k+1)]$ mapping. Therefore, the RKNN approach alleviates the difficulties introduced by the discretization methods. As known, the first order discretization brings large approximation errors. Wang and Lin has simulated the approach with RBFNN architecture and trained the neural network for priorily observed data [5]. This paper considers the approach with FNN with on-line tuning of the parameters.

The RKNN architecture is illustrated in Fig. 6. In this figure, *h* denotes the Runge-Kutta integration stepsize.

Robot dynamics can be stated more compactly as given by (14). All of the neural networks appearing in Fig. 6, realize the vector function *f*. Therefore, for fourth order Runge-Kutta approximation, the overall scheme is comprised of four times repeatedly connected neural network blocks and corresponding stage gains. The update mechanism is based on the error backpropagation. The derivation for FNN based identification scheme is given in (15) through (24).

$$\dot{\underline{x}} = \underline{f}(\underline{x},\underline{t}) \tag{14}$$

$$\underline{x}(i+1) = \underline{x}(i) + \frac{h}{6}\left(\underline{k}_0 + 2\underline{k}_1 + 2\underline{k}_2 + \underline{k}_3\right) \tag{15}$$

$$\underline{k}_0 = N(\underline{x};\phi) \qquad = N(\underline{x}_0;\phi) \tag{16}$$

$$\underline{k}_1 = N(\underline{x} + \frac{1}{2}h\underline{k}_0;\phi) = N(\underline{x}_1;\phi) \tag{17}$$

$$\underline{k}_2 = N(\underline{x} + \frac{1}{2}h\underline{k}_1;\phi) = N(\underline{x}_2;\phi) \tag{18}$$

$$\underline{k}_3 = N(\underline{x} + h\underline{k}_2;\phi) = N(\underline{x}_3;\phi) \tag{19}$$

where, ϕ is a generic parameter of neural network. Figure 6. clarifies how the error backpropagation rule is applied. There are two paths to be considered in this propagation. The first is the direct connection to the output summation, the other is through the FNN stages of the architecture. Therefore, each derivation, except the first one, will concern two terms. The rule is summarized below for the fourth order Runge-Kutta approximation.

$$\frac{\partial \underline{k}_0}{\partial \phi} = \frac{\partial N(\underline{x}_0;\phi)}{\partial \phi} \tag{20}$$

$$\frac{\partial \underline{k}_1}{\partial \phi} = \frac{\partial \underline{k}_1}{\partial \underline{x}_1}\frac{\partial \underline{x}_1}{\partial \underline{k}_0}\frac{\partial \underline{k}_0}{\partial \phi} + \frac{\partial \underline{k}_1}{\partial \phi} \tag{21}$$

$$\frac{\partial \underline{k}_2}{\partial \phi} = \frac{\partial \underline{k}_2}{\partial \underline{x}_2}\frac{\partial \underline{x}_2}{\partial \underline{k}_1}\frac{\partial \underline{k}_1}{\partial \phi} + \frac{\partial \underline{k}_2}{\partial \phi} \tag{22}$$

$$\frac{\partial \underline{k}_3}{\partial \phi} = \frac{\partial \underline{k}_3}{\partial \underline{x}_3}\frac{\partial \underline{x}_3}{\partial \underline{k}_2}\frac{\partial \underline{k}_2}{\partial \phi} + \frac{\partial \underline{k}_3}{\partial \phi} \tag{23}$$

$$\Delta\phi(i) = \frac{\eta h}{6}\left(\underline{d}^T(i) - \underline{x}^T(i)\right)\left(\frac{\partial \underline{k}_0}{\partial \phi} + 2\frac{\partial \underline{k}_1}{\partial \phi} + 2\frac{\partial \underline{k}_2}{\partial \phi} + \frac{\partial \underline{k}_3}{\partial \phi}\right) \tag{24}$$

In (24), η represents the learning rate and $\underline{d}(i)$ represents the measured state vector of the plant at time index *i*. Simulation results for RKNN methodology are presented in Fig. 8.

6 Adaptive Neuro Fuzzy Inference Systems (ANFIS) For System Identification

Adaptive Neuro-Fuzzy Inference Systems (ANFIS) constitute an appropriate combination of neural and fuzzy systems. This hybrid combination enables to deal with both the verbal and the numeric power of intelligent systems. As is known from the theory of fuzzy systems, different fuzzification and defuzzification mechanisms with different rule base structures can propose various solutions to a given task. This paper considers the ANFIS structure with first order Sugeno model containing 25 rules. Gaussian membership functions with product inference rule are used at the fuzzification level. Fuzzifier outputs the firing strengths for each rule. The vector of firing strengths is normalized. The resulting vector is defuzzified by utilizing the first order Sugeno model. The procedure is explained briefly in (25) through (29) for the ANFIS architecture illustrated in Fig. 7.

Construction of a simple rule base is as follows:

IF $\boldsymbol{x}$ ***is*** $\boldsymbol{A_1}$ *and* $\boldsymbol{y}$ ***is*** $\boldsymbol{B_1}$ *THEN* $f_1 = p_1x + q_1y + r_1$

IF $\boldsymbol{x}$ ***is*** $\boldsymbol{A_2}$ *and* $\boldsymbol{y}$ ***is*** $\boldsymbol{B_2}$ *THEN* $f_2 = p_2x + q_2y + r_2$

Depending on the system in hand, the parameters of the membership functions can be initialized so that the convergence speed is increased.

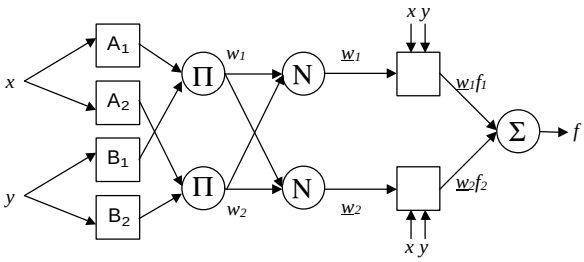


Fig. 7. Adaptive Neuro-Fuzzy Inference System (ANFIS) for two rules

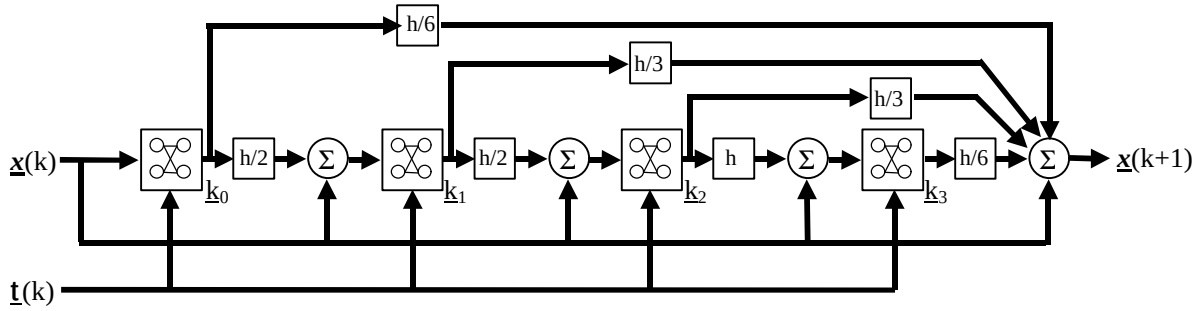


Fig. 6. Runge-Kutta neural network architecture.

$$w_1 = \mu_{A_1}(x)\mu_{B_1}(y) \quad (34)$$

$$w_2 = \mu_{A_2}(x)\mu_{B_2}(y) \quad (35)$$

$$\underline{w}_1 = \frac{w_1}{w_1 + w_2} \quad (36)$$

$$\underline{w}_2 = \frac{w_2}{w_1 + w_2} \quad (37)$$

$$f = \underline{w}_1(p_1 x + q_1 y + r_1) + \underline{w}_2(p_2 x + q_2 y + r_2) \quad (38)$$

The ANFIS output is clearly a linear function of the adjustable defuzzifier parameters. At the adjustment of $[\underline{p}\ \underline{q}\ \underline{r}]^T$ vector, gradient descent method is applied.

In robotic manipulator identification problem, the fuzzifier possesses six inputs, the rule base contains 25 rules and the defuzzifier has four outputs. The simulation results are presented in Fig. 9.

7 Conclusions

This study analyzes the performance of soft-computing methodologies from the point of system identification. In the assessment level, the estimation performance, together with the training times are considered as the primary comparison measures. Numerous simulations are performed on a direct drive robotic manipulator model.

All four of the approaches are tested for the same command signal. For the tracking error performance, ANFIS showed the best performance. On the other hand, RBFNN and FNN are the simplest approaches in the sense of computational complexity. Another important criterion is the requirement to a priori knowledge. Except the RKNN approach, all methods need a pre-training phase.

The contribution of this paper is to show the identification performance of ANFIS structure and to demonstrate the elegant performance of the RKNN approach with online operation and with ordinary feedforward neural network stages.

The work is in progress in the direction of improving the tracking performance through the use of combining ANFIS and RKNN approaches.

8 Acknowledgments

This work supported in part by Foundation for Promotion of Advanced Automation Technology, FANUC grant.

Table 2. Comparison of the methods.

	FNN	RBFNN	RKNN	ANFIS
Estimation performance	L	L	M	H
Pre-training time	L	H	-	L
Operational simplicity	H	H	M	M
Expert knowledge incorporation	L	M	L	H

Table 2. Comparison of the estimation methods.

	Bp	Ep	Bv	Ev	Σ
FNN	40e-3	40e-3	20e-3	50e-3	150e-3
RBFNN	40e-3	40e-3	40e-3	10e-3	160e-3
RKNN	20e-3	20e-3	20e-3	20e-3	80e-3
ANFIS	1e-3	0.2e-3	20e-3	10e-3	31.2e-3

Table 3. Comparison of the pre-estimation times.

	Time(hours)
FNN	0.5
RBFNN	96
RKNN	-
ANFIS	0.4

References

[1] Hornik, K., "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, pp 359-366, 1989.

[2] Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, pp 183-192, 1989.

[3] Narendra, K. S., Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, March 1990.

[4] Jang, J.-S. R., C.-T.Sun, "Neuro-Fuzzy Modelling and Control", Proc. IEEE, 83, 378-406, 1995.

[5] Wang, Y-J., Lin C-T., "Runge-Kutta Neural Network for Identification of Dynamical Systems in High Accuracy", *IEEE Transactions on Neural Networks*, vol. 9., no. 2, pp. 294-307, March 1998.

[6] Jang, J.-S. R., Sun, C.-T., Mizutani, E., Neuro-Fuzzy and Soft Computing, PTR Prentice Hall, 1997.

[7] Wang, L., *Adaptive Fuzzy Systems and Control, Design and Stability Analysis*, PTR Prentice Hall, 1994

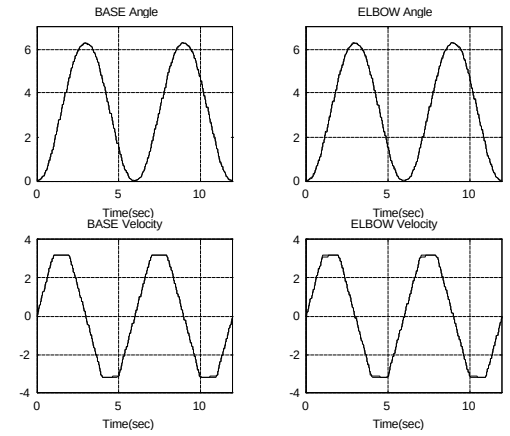
[8] Passino, K. M., "Intelligent Control for Autonomous Systems", *IEEE Spectrum*, 32. 55-62, 1996

[9] Erbatur K., Kaynak, O., Rudas I., "An Inverse Dynamics Based Robot Control Method Using Fuzzy Identifiers", IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 16-20 June, 1997, Tokyo, Japan

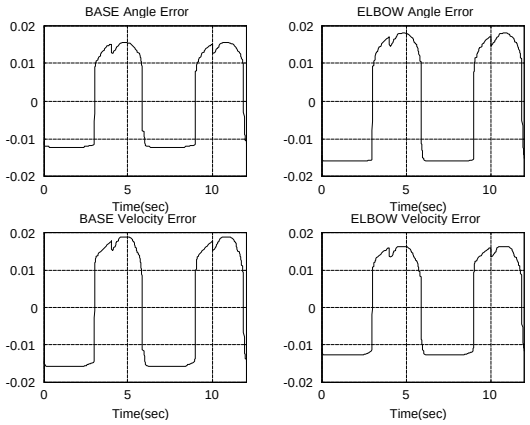
[10] Zadeh, L., A., "Fuzzy Sets", *Information and Control*, Vol. 8, 1965

[11] Miller, W. T., Sutton, R. S., Werbos, P. J., *Neural Networks for Control*, MIT Press, 1991.

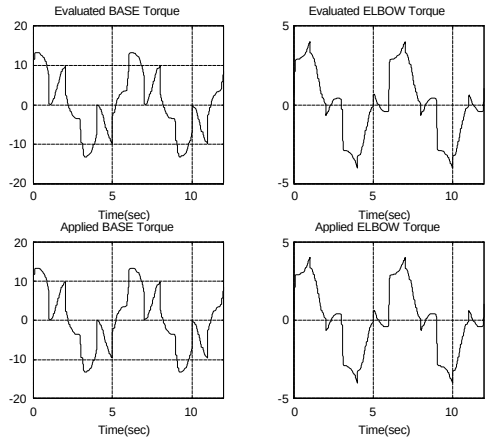
[12] Efe, M. Ö., "Identification and Control of Nonlinear Dynamical Systems Using Neural Networks," M.S. Thesis, Boğaziçi University, 1996.



a. State tracking graph

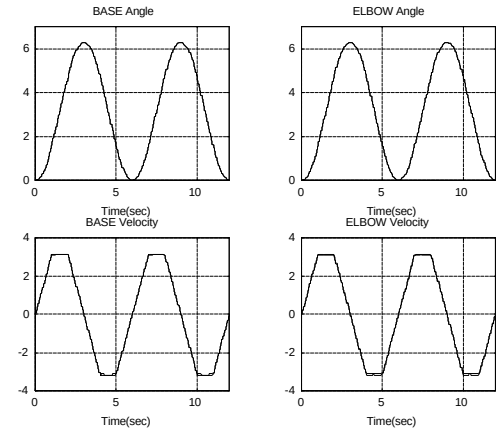


b. State tracking error graph

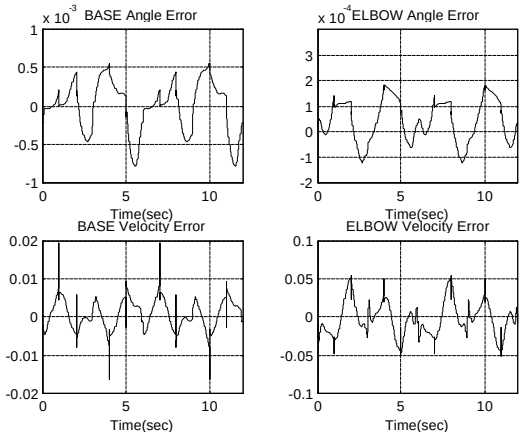


b. State tracking error graph

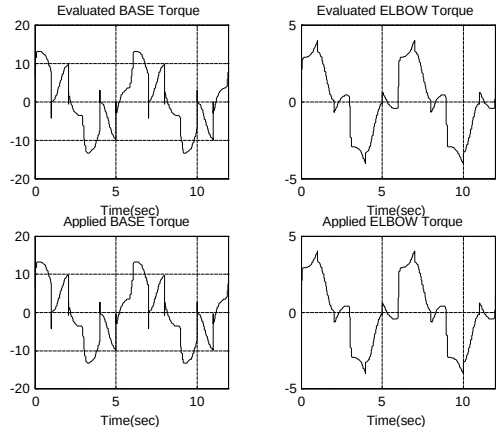
Fig. 8. Simulation results for RKNN structure.



a. State tracking graph



b. State tracking error graph



c. Applied torque graph

Fig. 9. Simulation results for ANFIS structure.