ELSEVIER

Robotics and Autonomous Systems 30 (2000) 221–230

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

A comparative study of soft-computing methodologies in identification of robotic manipulators

M. Onder Efe, Okyay Kaynak [*]

Electrical and Electronics Engineering Department, Mechatronics Research and Application Center, Bogaziçi University, Bebek, 80815 Istanbul, Turkey

Received 12 April 1999

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. ©2000 Elsevier Science B.V. All rights reserved.

Keywords: Neural networks; Fuzzy systems; Identification; Robotics; Comparison

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 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 the combination of an expert's knowledge and massively parallel, and adaptive data processing architecture of the computationally intelligent approach adopted. The most popular members of the soft-computing methodologies are the neural networks and fuzzy inference systems. Neural networks provide the mathematical power of the brain whereas the fuzzy logic based mechanisms employ the verbal

[*] Corresponding author. Tel.: +90-212-287-2475; fax: +90-212-287-2465.
E-mail addresses: efemond@boun.edu.tr (M.O. Efe), kaynak@boun.edu.tr (O. Kaynak).

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 [5], and Funahashi [4] have shown that as long as the hidden layer comprises a 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 [7]. 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 an error backpropagation algorithm [8]. On the other hand, a novel approach has been presented in [11] 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 book the details of an 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 of a two degrees of freedom direct drive robotic manipulator.

In [2], several neural network based identification strategies are discussed on a three degrees of freedom anthropoid robot. Erbatur et al. [3] consider the standard fuzzy system model proposed by Wang [9,10] for the approximation of the inverse dynamics of the manipulator studied in this paper.

The identification procedure followed in all of the methods considered is as depicted in Fig. 1. During the training, the 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.

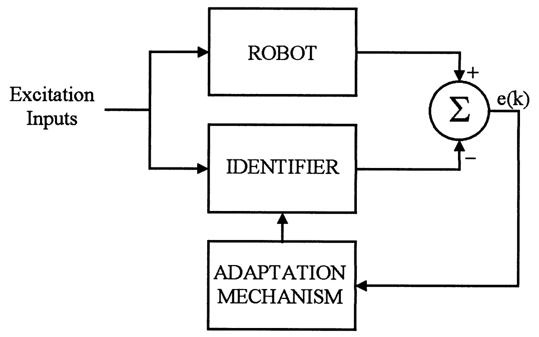

Fig. 1. Identification of a robotic manipulator.

In the next section the system under observation is introduced, the following section dwells on FNN based identification schemes. Next, the RBFNN approach is derived. Section 5 describes the application of Runge–Kutta neural network methodology for the identification of robotic manipulators. In Section 6 the ANFIS structure is elaborated. In Section 7, a comparison of the four approaches is presented. Conclusions constitute the last part of the paper.

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.

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

By assuming the angular positions and angular velocities as state variables of the system, a total of four first-order differential equations is obtained. The state

Table 1
Manipulator parameters

Motor 1 rotor inertia	0.267	I1
Arm 1 inertia	0.334	I2
Motor 2 rotor inertia	0.0075	I3
Motor 2 stator inertia	0.040	I3C
Arm 2 inertia	0.063	I4
Payload inertia	0.000	IP
Motor 1 mass	73.0	M1
Arm 1 mass	9.78	M2
Motor 2 mass	14.0	M3
Arm 2 mass	4.45	M4
Payload mass	0.00	Mp
Arm 1 length	0.359	L1
Arm 2 length	0.24	L2
Arm 1 CG	0.136	L3
Arm 2 CG	0.102	L4
Axis 1 friction	5.3	F1
Axis 2 friction	1.1	F2
Torque limit 1	245.0	
Torque limit 2	39.2	

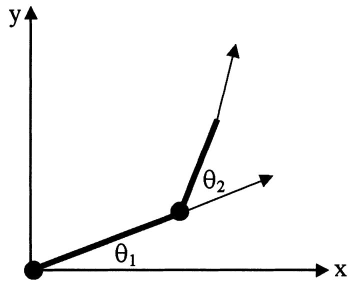

Fig. 2. Architecture of a 2 DOF planar manipulator.

varying inertia matrix and coriolis terms are given in (2) and (3), respectively.

$$M(\theta) = \begin{bmatrix} p_1 + 2p_3\cos(\theta_2) & p_2 + p_3\cos(\theta_2) \\ p_2 + p_3\cos(\theta_2) & p_2 \end{bmatrix}, \quad (2)$$

$$V(\theta, \dot\theta) = \begin{bmatrix} -\dot\theta_2(2\dot\theta_1 + \dot\theta_2)p_3 \sin(\theta_2) \\ \dot\theta_1^2 p_3 \sin(\theta_2) \end{bmatrix}, \quad (3)$$

where $p_1 = 2.0857$, $p_2 = 0.1168$ and $p_3 = 0.1630$. The details and the derivation of the manipulator dynamics can be found in [1].

3. Feedforward neural networks (FNN) for system identification

An identification procedure, in the most general sense, entails a matching between the system outputs and an identifier output. Artificial neural networks, due to their ability to act as universal approximators, can very effectively be used for this purpose [4,5]. Narendra and Parthasarathy [7] 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 (4) 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, [8] should be reviewed.

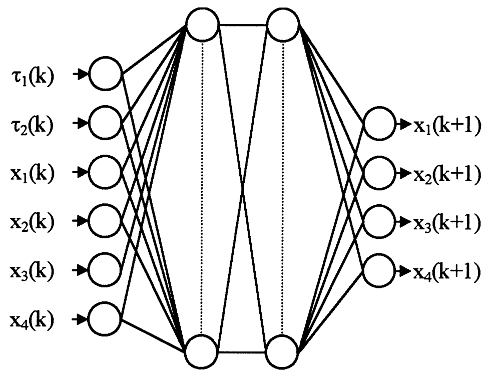

Fig. 3. FNN architecture.

$$J = \frac{1}{2}\sum_{i=1}^{N}(d_i^p - y_i^p)^2, \quad (4)$$

$$\delta_j^{k+1,p} = (d_j^p - y_j^{k+1,p})\Psi'(S_j^{k+1,p}), \quad (5)$$

$$\delta_j^{k+1,p} = \left(\sum_{h=1}^{\#\text{neurons}_{k+2}} \delta_h^{k+2,p} w_{jh}^{k+1}\right)\Psi'(S_j^{k+1,p}). \quad (6)$$

In (4), 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. Eqs. (5) and (6) give the delta values for the output layer and hidden layer neurons, respectively.

In (5) and (6), S_j^{k+1} denotes the net summation of the jth neuron in the $(k+1)$th layer, and 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 (7) is applied for each training pair.

$$\Delta w_{ij}^k = \eta\delta_j^{k+1,p} o_i^{k,p}, \quad (7)$$

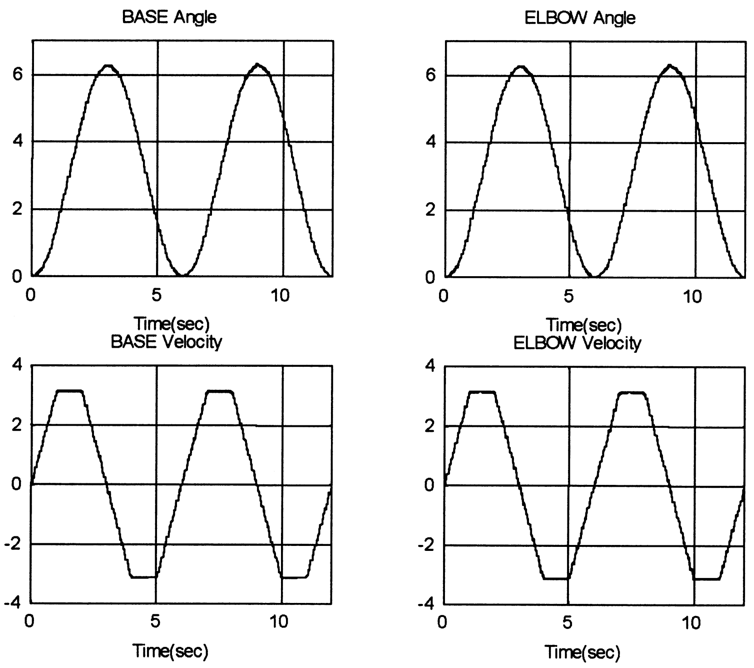

Fig. 4. Behavior of the plant under control.

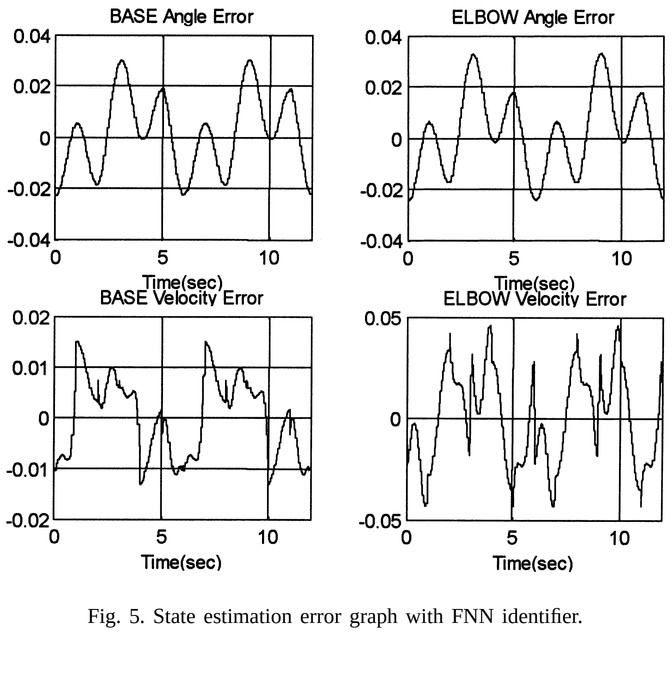

Fig. 5. State estimation error graph with FNN identifier.

In the simulations, the plant is kept under an external control loop, which enforces the manipulator to follow a trapezoidal velocity profile for both links as illustrated in Fig. 4. All four approaches discussed throughout the paper aim to estimate the behavior in Fig. 4. The state estimation error graphs for FNN based identification schemes are presented in Fig. 5.

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 an FIS, then there exists a functional equivalence between RBFNN and FIS [6]. In Fig. 6, an 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.

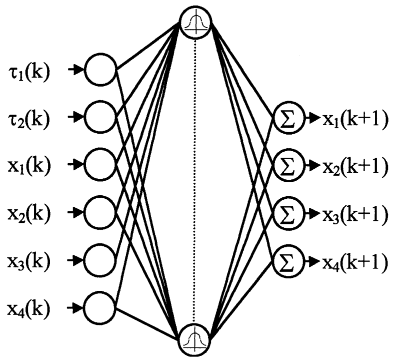

Fig. 6. RBFNN architecture.

In this approach, 12 hidden neurons are used. As is given by (8), each neuron output is evaluated from the multiplication of the outputs of individual Gaussians corresponding to each input. The overall response is evaluated through the multiplication of a hidden layer output vector by a matrix of appropriate dimensions. In fuzzy terms, this is equivalent to saying that a product inference rule is used with weighted sum defuzzifier. If the network output and hidden layer output vectors are denoted by $\boldsymbol{y}$ and $\boldsymbol{o}$, respectively, the activation level of the ith receptive unit (or firing strength of the ith rule) and the overall realization performed by the network can be given by (8) and (9), respectively.

$$o_i = \prod_{j=1}^{\#\text{inputs}} \exp\left(-\frac{(x_j - c_{ij})^2}{w_{ij}^2}\right), \tag{8}$$

$$\mathbf{y} = W_{\#\text{outputs}\times\#\text{hidden neurons}}\,\boldsymbol{o}. \tag{9}$$

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

$$\boldsymbol{\delta}_{\text{output}} = \boldsymbol{d} - \boldsymbol{y}, \tag{10}$$

$$\boldsymbol{\delta}_{\text{hidden}} = W^{\mathrm{T}}\boldsymbol{\delta}_{\text{output}}, \tag{11}$$

$$W_{ij}(k+1) = W_{ij}(k) + \eta\delta_{\text{output}\,i}\,o_j, \tag{12}$$

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

$$w_{ij}(k+1) = w_{ij}(k) + \eta\delta_{\text{hidden}\,i}\,o_j 2\frac{(x_j - c_{ij}(k))^2}{w_{ij}(k)^3}, \tag{14}$$

The state estimation error graphs for the RBFNN based identification scheme are presented in Fig. 7.

5. Runge–Kutta neural networks (RKNN) for system identification

The Runge–Kutta method is a powerful way of solving the behavior of a dynamic system if the system

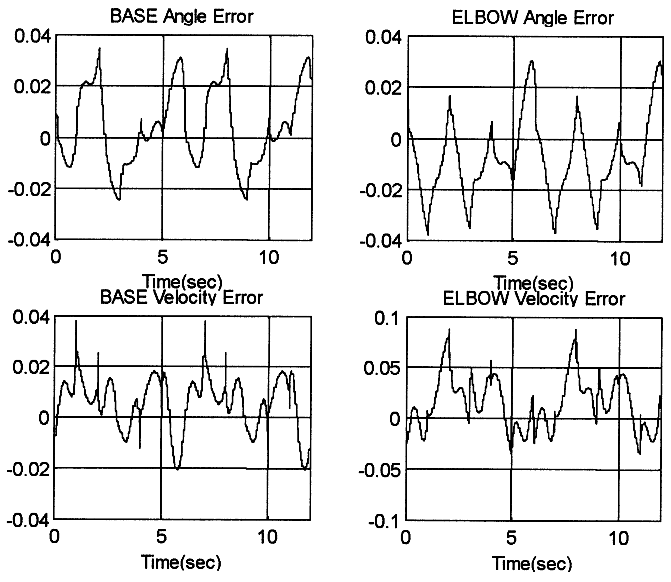

Fig. 7. State estimation error graph with RBFNN identifier.

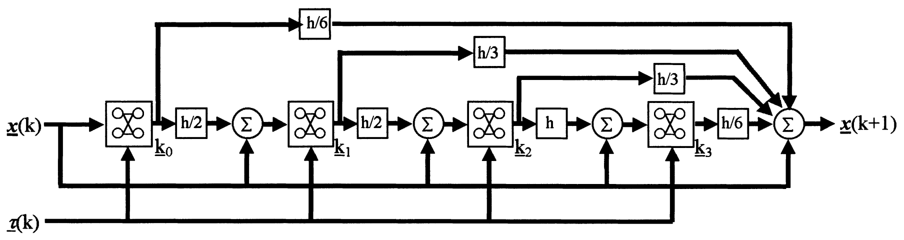

Fig. 8. Runge–Kutta neural network architecture.

is characterized by ordinary differential equations. In [11], the proposed method is applied to several problems. Mainly, the method is observed to be successful in estimating the system states given long enough time. It should be emphasized that the neural network architecture realizes the changing rates of the system states instead of the $[\boldsymbol{x}(\boldsymbol{k}), \boldsymbol{\tau}(\boldsymbol{k})] \Rightarrow [\boldsymbol{x}(k+1)]$ 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 [11] have simulated the approach with the RBFNN architecture and trained the neural network for priorily observed data. This paper considers the approach with FNN, with on-line tuning of the parameters. The RKNN architecture is illustrated in Fig. 8. In this figure, h denotes the Runge–Kutta integration stepsize.

Robot dynamics can be stated more compactly as given by (15). All of the neural networks appearing in Fig. 8 realize the vector function $\boldsymbol{f}$. 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 (16) through (25).

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau}), \tag{15}$$

$$\boldsymbol{x}(i+1) = \boldsymbol{x}(i) + \tfrac{1}{6}h(\boldsymbol{k}_0 + 2\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + \boldsymbol{k}_3), \tag{16}$$

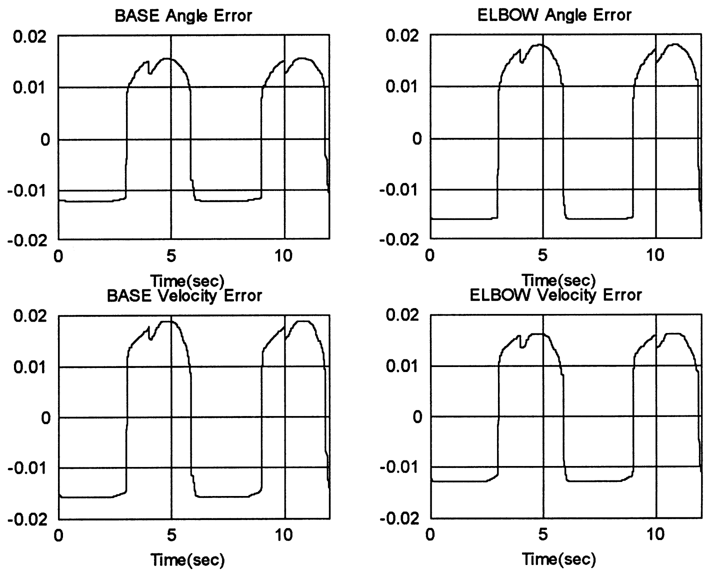

Fig. 9. State estimation error graph with RKNN identifier.

$$k_0 = N(x;\phi) = N(x_0;\phi), \tag{17}$$

$$k_1 = N\left(x + \tfrac{1}{2}hk_0;\phi\right) = N(x_1;\phi), \tag{18}$$

$$k_2 = N\left(x + \tfrac{1}{2}hk_1;\phi\right) = N(x_2;\phi), \tag{19}$$

$$k_3 = N(x + hk_2;\phi) = N(x_3;\phi), \tag{20}$$

where ϕ is a generic parameter of neural network. Fig. 8 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 partial derivative, except the first one, will concern two terms. The rule is summarized below for the fourth-order Runge–Kutta approximation.

$$\frac{\partial k_0}{\partial\phi} = \frac{\partial N(x_0;\phi)}{\partial\phi}, \tag{21}$$

$$\frac{\partial k_1}{\partial\phi} = \frac{\partial k_1}{\partial x_1}\frac{\partial x_1}{\partial k_0}\frac{\partial k_0}{\partial\phi} + \frac{\mathrm{d}k_1}{\mathrm{d}\phi} \tag{22}$$

$$\frac{\partial k_2}{\partial\phi} = \frac{\partial k_2}{\partial x_2}\frac{\partial x_2}{\partial k_1}\frac{\partial k_1}{\partial\phi} + \frac{\mathrm{d}k_2}{\mathrm{d}\phi}, \tag{23}$$

$$\frac{\partial k_3}{\partial\phi} = \frac{\partial k_3}{\partial x_3}\frac{\partial x_3}{\partial k_2}\frac{\partial k_2}{\partial\phi} + \frac{\mathrm{d}k_3}{\mathrm{d}\phi}, \tag{24}$$

$$\Delta\phi(i) = \frac{\eta h}{6}(d^{\mathrm{T}}(i) - x^{\mathrm{T}}(i)) \times \left(\frac{\partial x_0}{\partial\phi} + 2\frac{\partial x_1}{\partial\phi} + 2\frac{\partial k_2}{\partial\phi} + \frac{\partial k_3}{\partial\phi}\right). \tag{25}$$

In (25), η represents the learning rate and $d(i)$ represents the measured state vector of the plant at time index i. Simulation results for the RKNN methodology are presented in Fig. 9.

6. Adaptive neuro-fuzzy inference systems (ANFIS) for system identification

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. The fuzzifier outputs the firing strengths for each rule. The vector of firing strengths is normalized. The resulting vector is

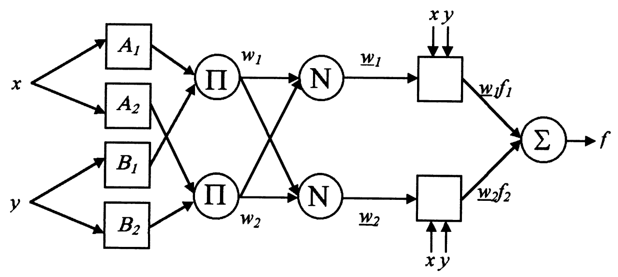

Fig. 10. Adaptive neuro-fuzzy inference system (ANFIS) for two rules.

defuzzified by utilizing the first-order Sugeno model. The procedure is explained briefly in (26)–(30) for the ANFIS architecture illustrated in Fig. 10. Construction of a simple rule base is as follows:

IF $\boldsymbol{x}$ is $\boldsymbol{A}_1$ and $\boldsymbol{y}$ is $\boldsymbol{B}_1$ THEN $f_1 = p_1 x + q_1 y + r_1$

IF $\boldsymbol{x}$ is $\boldsymbol{A}_2$ and $\boldsymbol{y}$ is $\boldsymbol{B}_2$ THEN $f_2 = p_2 x + q_2 y + r_2$

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

$$w_1 = \mu_{A_1}(x)\mu_{B_1}(y), \tag{26}$$

$$w_2 = \mu_{A_2}(x)\mu_{B_2}(y), \tag{27}$$

$$\boldsymbol{w}_{n1} = \frac{w_1}{w_1 + w_2}, \tag{28}$$

$$\boldsymbol{w}_{n2} = \frac{w_2}{w_1 + w_2}, \tag{29}$$

$$f = \boldsymbol{w}_{n1}(p_1 x + q_1 y + r_1) + \boldsymbol{w}_{n2}(p_2 x + q_2 y + r_2), \tag{30}$$

The ANFIS output is clearly a linear function of the adjustable defuzzifier parameters. At the adjustment of the $[\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}]^{\mathrm{T}}$ vector, the gradient descent method is applied. In the 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. 11.

7. Comparison of the four approaches

In order to study the performances of the four approaches described above, a series of simulation studies has been performed and some performance measures are tabulated in Tables 2–4. In Table 2, each column represents the error bound for a variable, e.g., B_{p} stands for the position error bound for the base link, E_{v} is for the velocity error bound for the elbow link and so on. The use of RKNN and ANFIS architectures result in a good estimation performance, as is clearly seen by the respective position and velocity error bounds for both links. Table 3 compares another distinguishing property of the approaches namely the

Table 2
Comparison of the estimation methods

	B_{p}	E_{p}	B_{v}	E_{v}	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-training times

	Time (h)
FNN	0.5
RBFNN	96
RKNN	–
ANFIS	0.4

Table 4
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

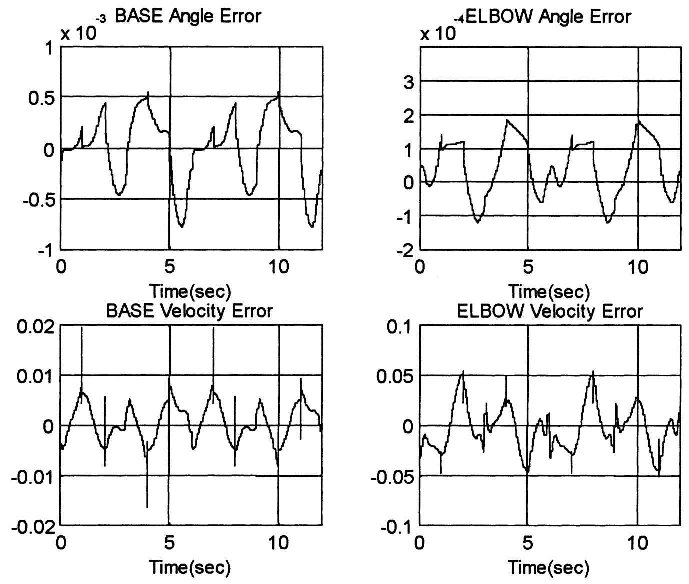

Fig. 11. State estimation error graph with ANFIS identifier.

need for pre-training. In this table, no figure is given for the RKNN method because it operates on-line. ANFIS and FNN architectures require approximately the same amount of pre-training time. The RBFNN approach requires by far the longest pre-training time. The overall assessment of the approaches is given in Table 4, where H denotes high, M denotes medium and L denotes low. The approaches are evaluated for four different comparison measures. The first row clearly recommends the use of RKNN or ANFIS. The second row accounts for the need for pre-training time. In this sense, RKNN is the best approach. On the other hand, from the viewpoint of operational simplicity, FNN and RBFNN are the simplest approaches. The last row considers the ability to incorporate expert knowledge. As is known from the theory of fuzzy systems, the philosophy of the ANFIS architecture best fulfills this task because the design starts with the rules in the form of IF–THEN statements which are the best means for expressing the expert–machine interaction mathematically.

8. 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 two degrees of freedom direct drive robotic manipulator model.

All four 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 for the RKNN approach, all methods need a pre-training phase.

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

Acknowledgements

This work supported in part by Foundation for Promotion of Advanced Automation Technology, FANUC grant and Bogaziçi University Research Fund (Project No: 97A0202).

References

[1] Direct Drive Manipulator R&D Package User Guide, Integrated Motions Incorporated, 704 Gillman Street, Berkeley, CA.

[2] M.O. Efe, O. Kaynak, A comparative study of neural network structures in identification of nonlinear systems, Mechatronics 9 (3) (1999) 287–300.

[3] K. Erbatur, O. Kaynak, I.J. Rudas, An inverse dynamics based robot control method using fuzzy identifiers, in: Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Tokyo, 1997.

[4] K. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (1989) 183–192.

[5] K. Hornik, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[6] J.-S.R. Jang, C.-T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, PTR Prentice Hall, Upper Saddle River, NJ, 1997, pp. 335–368.

[7] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Transactions on Neural Networks 1 (1) (1990) 4–27.

[8] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland and the PDP Research Group (Eds.), Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.

[9] L.X. Wang, Adaptive Fuzzy Systems and Control, Design and Stability Analysis, PTR Prentice Hall, Upper Saddle River, NJ, 1994, pp. 29–48.

[10] L.X. Wang, A Course in Fuzzy Systems and Control, PTR Prentice Hall, Upper Saddle River, NJ, 1997, pp. 168–179.

[11] Y.-J. Wang, C.-T. Lin, Runge-Kutta neural network for identification of dynamical systems in high accuracy, IEEE Transactions on Neural Networks 9 (2) (1998) 294–307.

M. Onder Efe received the B.Sc. and M.S. degrees from the Electronics and Communications Engineering Department, Istanbul Technical University, Turkey, in 1993 and from the Systems and Control Engineering Department, Bogaziçi University, Istanbul, Turkey, 1996, respectively. He is currently working towards the Ph.D. degree at the Bogaziçi University, Electrical and Electronics Engineering Department. Since 1996, he has been working at the Bogaziçi University, Mechatronics Research and Application Center as a Research Assistant. His research interests include artificial neural networks, fuzzy inference systems, variable structure control, stabilizing training strategies for computationally intelligent systems and robotics.

O. Kaynak received his B.Sc. (First Class, Honours) and Ph.D. degrees in Electronics and Electrical Engineering from the University of Birmingham, UK, in 1969 and 1972, respectively. After spending seven years in industry, in January 1979 he joined the Department of Electrical and Electronics Engineering of Bogaziçi University, Istanbul, Turkey, where he is presently a Full Professor. He has served as the Chairman of the Computer Engineering Department (3 years) and as the Director of Biomedical Engineering Institute (1 year) at this university. Currently, he is the Chairman of the Electrical and Electronic Engineering Department and the holder of the UNESO Chair on Mechatronics and the Director of the Mechatronics Research and Application Center. He has held long term visiting professor/scholar positions at various institutions in Japan, Germany, USA and Singapore. His current research interests are in the field of Intelligent Control and Mechatronics. He has published two books, edited two, and also a conference proceedings. Dr. Kaynak is presently a Vice President of the IEEE (The Institute of Electrical and Electronics Engineers) Industrial Electronics Society and Associate Editor of *IEEE Transactions on Industrial Electronics*.