Neural Networks Approaches for Computing the Forward Kinematics of a Redundant Parallel Manipulator

H. Sadjadian , H.D. Taghirad *Member, IEEE* and A. Fatehi

***Abstract*—**In this paper, different approaches to solve the forward kinematics of a three DOF actuator redundant hydraulic parallel manipulator are presented. On the contrary to series manipulators, the forward kinematic map of parallel manipulators involves highly coupled nonlinear equations, which are almost impossible to solve analytically. The proposed methods are using neural networks identification with different structures to solve the problem. The accuracy of the results of each method is analyzed in detail and the advantages and the disadvantages of them in computing the forward kinematic map of the given mechanism is discussed in detail. It is concluded that ANFIS presents the best performance compared to MLP, RBF and PNN networks in this particular application.

***Keywords*—**Forward Kinematics, Neural Networks, Numerical Solution, Parallel Manipulators.

I. Introduction

OVER the last two decades, parallel manipulators have been among the most considerable research topics in the field of robotics. These robots are now applied in real-life applications such as force sensing robots, fine positioning devices, and medical applications [1]-[2].

As in the case of conventional serial robots, kinematics analysis of parallel manipulators is also performed in two phases. In forward or direct kinematics the position and orientation of the mobile platform is determined given the leg lengths. This is done with respect to a base reference frame. In inverse kinematics we use position and orientation of the mobile platform to determine actuator lengths. It is known that unlike serial manipulators, inverse position kinematics for parallel robots is usually simple and straightforward. In most cases, joint variables (actuator displacements) may be computed independently using the given pose of the movable platform. The solution to this problem is in most cases uniquely determined. But forward kinematics of parallel manipulators is generally very complicated. Its solution usually involves systems of nonlinear equations which are highly coupled and in general have no closed form and unique solution. Different approaches are provided in literature to solve this problem either generally or in special cases. There are also numerous cases in which the solution to this problem is provided for a special or novel architecture. In general, different solutions to this problem can be found using numerical approaches, analytical approaches, and closed form solution for special architectures [3]-[4].

Manuscript received on March 18, 2005.

Authors are with the K.N. Toosi University of Technology, Electrical Engineering Department, Advanced Robotics and Automated System (ARAS), P.O. Box 16315-1355, Tehran, IRAN. (phone: +98-21-846-8094; fax: +98-21-846-2066; e-mail: sadjadian@alborz.kntu.ac.ir and (Taghirad,Fatehi)@kntu.ac.ir).

In this paper, Four different types of neural networks; multilayer perceptron (MLP), Radial Basis Function neural network (RBF), polynomial neural networks (PNN) and adaptive-network-based fuzzy inference system (ANFIS) have been successfully used to solve the forward kinematics problem in a 3DOF actuator redundant hydraulic parallel manipulator, which generalizes the application of such networks to spatial parallel mechanisms. The performances of such networks are compared in detail for the above problem. The paper is organized as following. Section 2 contains the mechanism description. Kinematic modelling of the manipulator is discussed in section 3, where inverse and forward kinematics is studied and the need for appropriate method to solve the forward kinematics is justified. In section 4, different methods to solve the forward kinematics problem are discussed; First, two different but mostly common neural networks, MLP and RBF, are used to estimate the forward kinematic map of the given mechanism. In the third method a polynomial neural network is provided to approximate the nonlinear map with required precision. Then in the forth experience ANFIS is applied for forward kinematics solution. In section 5, these methods are simulated and compared regarding the problem in hand in order to identify the benefits and drawbacks of each scheme.

II. Mechanism Description

A three DOF actuator redundant hydraulic parallel manipulator is used as the basis of our study. The mechanism is designed by Hayward [5]-[7], borrowing design ideas from biological manipulators and specially the biological shoulder. The interesting features of the mechanism and its similarity to human shoulder have made it a unique design, which can serve as a basis for a good experimental setup for parallel robot research. A picture of the mechanism, which is currently under experimental studies in ARAS Robotics Lab, is shown in Fig. 1. The mobile platform is constrained to spherical motions. Four high performance hydraulic piston actuators are used to give three degrees of freedom in the mobile platform. Each actuator includes a position sensor of LVDT type and an embedded Hall Effect force sensor. Simple elements, like spherical and universal joints, are used in the structure.

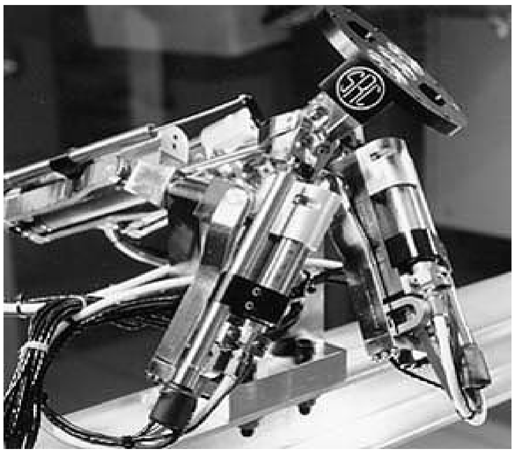
Fig.1.The hydraulic shoulder manipulator in movement

A complete analysis of such a careful design will provide us with good results regarding the structure itself and its performance. From the structural point of view, the shoulder mechanism which, from now on, we call it "the Hydraulic Shoulder" falls into an important class of robotic mechanisms called parallel robots. In these robots, the end effector is connected to the base through several closed kinematic chains. The motivation behind using these types of robot manipulators was to compensate for the shortcomings of the conventional serial manipulators such as low precision, low stiffness, error accumulation and load carrying capability. However, they have their own disadvantages, which are mainly smaller workspace and many singular configurations. The hydraulic shoulder, being a parallel structure, has the general features of these structures. It can be thought of as a shoulder for a light weighed seven DOF robotic arm, which can carry loads several times its own weight. The workspace of such a mechanism can be considered as part of a sphere surface. The orientation angles are limited to vary between $-\pi/6$ and $\pi/6$.

III. Kinematic Modeling

The hydraulic shoulder is kinematically over constrained. The inverse kinematics problem is easily solved, given the orientation of the mobile plate, similar to general parallel robots. The inverse kinematics problem has a unique solution, in our case, meaning that the hydraulic shoulder cannot be optimized by choosing between the solutions. Fig. 2 depicts a geometric model for the mechanism which will be used for its kinematics derivation. The parameters used in kinematics can be defined as:

$$l_b = \left\|\overrightarrow{CA_i}\right\| \quad l_p = \left\|\overrightarrow{CC_1}\right\| \quad l_d = \left\|\overrightarrow{C_1P_i}\right\|_{y1} \quad l_k = \left\|\overrightarrow{C_1P_i}\right\|_{z1}$$

α: The angle between CA_4 and y_0

C: Center of the reference frame

C_1: Center of the moving plate

ρ_i: Actuator lengths i=1, 2, 3, 4

P_i: Moving endpoints of the actuators

A_i: Fixed endpoints of the actuators

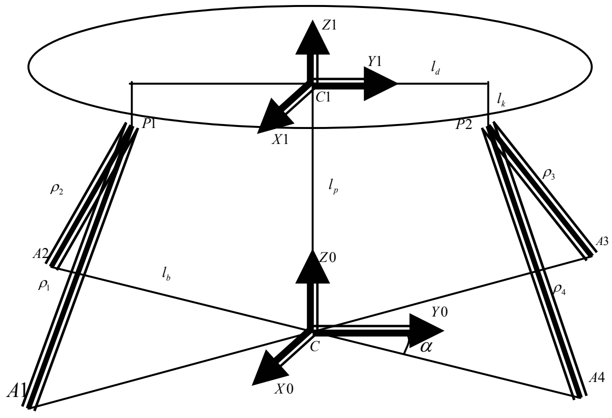

Fig. 2. A geometric model for the hydraulic shoulder

Two coordinate frames are defined. The base frame $X_0Y_0Z_0$ is centered at C (rotation center) with its Z_0-axis perpendicular to the plane defined by $A_1A_2A_3A_4$ and an X_0 axis parallel to the bisector of angle $\angle A_1CA_4$. The second frame, namely $X_1Y_1Z_1$ is centered at C_1 (center of the moving plate) with its Z_1 axis perpendicular to the line defined by the actuators moving end points (P_1P_2) and horizontal Y axis along C_1P_2.

A. Inverse Kinematics

In modeling the inverse kinematics of the hydraulic shoulder we must determine actuator lengths (ρ_i) as the joint space variables given the task space variables, namely θ_x, θ_y and θ_z as the orientation angles of the moving platform. First we note that the fixed end points of the actuators (A_i) can be written in the base frame as:

$$\begin{aligned} A_1^0 &= (l_b\sin\alpha \quad -l_b\cos\alpha \quad 0),\\ A_2^0 &= (-l_b\sin\alpha \quad -l_b\cos\alpha \quad 0),\\ A_3^0 &= (-l_b\sin\alpha \quad l_b\cos\alpha \quad 0),\\ A_4^0 &= (l_b\sin\alpha \quad l_b\cos\alpha \quad 0), \end{aligned} \tag{1}$$

Also:

$$P_1^1 = (0 \quad -l_d \quad -l_k),\ P_2^1 = (0 \quad l_d \quad -l_k), \tag{2}$$

These must be transferred to the base frame using the rotation matrix R_1^0;

$$P_i^0 = R_1^0 P_i^1, \tag{3}$$

where:

$$S_{3\times3} = R_1^0 = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x) \tag{4}$$

The rotation matrix components are computed as following:

$$\begin{aligned} S_{11} &= \cos(\theta_z)\cos(\theta_y)\\ S_{21} &= \sin(\theta_z)\cos(\theta_y)\\ S_{31} &= -\sin(\theta_y)\\ S_{12} &= \cos(\theta_z)\sin(\theta_y)\sin(\theta_x) - \sin(\theta_z)\cos(\theta_x)\\ S_{22} &= \sin(\theta_z)\sin(\theta_y)\sin(\theta_x) + \cos(\theta_z)\cos(\theta_x)\\ S_{32} &= \cos(\theta_y)\sin(\theta_x)\\ S_{13} &= \cos(\theta_z)\sin(\theta_y)\cos(\theta_x) + \sin(\theta_z)\sin(\theta_x)\\ S_{23} &= \sin(\theta_z)\sin(\theta_y)\cos(\theta_x) - \cos(\theta_z)\sin(\theta_x)\\ S_{33} &= \cos(\theta_y)\cos(\theta_x) \end{aligned} \tag{5}$$

So we have:

$$\mathrm{p}_1^0=\begin{pmatrix}-l_d s_{12}-l_k s_{13}\\ -l_d s_{22}-l_k s_{23}\\ -l_d s_{32}-l_k s_{33}\end{pmatrix},\ \mathrm{p}_2^0=\begin{pmatrix}l_d s_{12}-l_k s_{13}\\ l_d s_{22}-l_k s_{23}\\ l_d s_{32}-l_k s_{33}\end{pmatrix} \qquad (6)$$

The final step is to translate the resulting vectors P_i^0 by l_p along the Z axis. Having P_i^0 and A_j^0 in hand, the actuator lengths $\left\|\overrightarrow{\mathrm{P}_i\mathrm{A}_j}\right\|$ can be easily computed as:

$$\rho_j=\sqrt{(\mathrm{p}_x-\mathrm{a}_x)^2+(\mathrm{p}_y-\mathrm{a}_y)^2+(\mathrm{p}_z-\mathrm{a}_z)^2}\ , \qquad (7)$$

where:

$$\mathrm{p}_i^0=[\mathrm{p}_x\quad \mathrm{p}_y\quad \mathrm{p}_z]^T \quad , \quad \mathrm{A}_j^0=[\mathrm{a}_x\quad \mathrm{a}_y\quad \mathrm{a}_z], \qquad (8)$$

are defined in (6) and (1) respectively. From (7) and (8), the actuator lengths (ρ_i) are exactly computable by the orientation angles of the moving platform, θ_x, θ_y and θ_z, and hence the inverse kinematic map is analytically computed. It is clear that the manipulator doesn't have any kinematic redundancy, meaning that reaching a specific point in the task space can't be satisfied through different combinations of the actuator lengths.

B. *Forward Kinematics*

Equations (7)–(8) can also be used for the forward kinematics of the hydraulic shoulder but with the actuator lengths as the input and orientation angles θ_x, θ_y, θ_z as the unknown outputs. In fact, we have four nonlinear equations to solve for three unknowns. Obviously, solving such a system of nonlinear equations for a unique closed-form analytic solution to the forward kinematic problem is very complicated, although three equations of the four could be used. Several inconclusive attempts have been made in this direction which failed on solving the problem. Therefore, we propose using numerical schemes to solve the forward kinematic problem as a basic element in modeling and control of the manipulator. This is studied in detail in the next section.

IV. Forward Kinematics Solution

A. *Multilayer perceptron network*

A simple multilayer perceptron neural network (MLP) with back propagation learning was used in the first step. The input layer has as many nodes as the number of inputs to the map, namely four actuator lengths. Similarly the output layer will have three nodes which represent the orientation of the moving plate ($\theta_x, \theta_y, \theta_z$). The number of neurons in the hidden layer was used as a design parameter. Sigmoid and linear transfer functions were selected for all hidden and output layer nodes respectively. Supervised learning scheme was used in which the manipulator is treated as a black box and the network is taught to learn the map by observing the inputs and outputs. Such a learning scheme will result in offline training. For producing the training data, the target pattern, i.e. the three orientation angles, was randomly generated within the workspace of the robot and the input pattern, i.e. four actuator displacements, was found using the inverse kinematics model. The pair was then used to train the network in a back propagation process. Random initialization was used for the weights. Different configurations of the MLP network were tested by varying the number of neurons in the hidden layer between 5 and 35 and the performance of these networks was compared.

Different performance indices could be used in this case, the best of which could be the sum of square output errors, though other indices such as mean square or mean absolute error may also be used. Networks with best performance as indicated would be selected, from which the network with fewer hidden layer nodes will be better choice since the number of weights and also the training time of the network increase with more neurons in the hidden layer. As another configuration, the same multilayer perceptron network was used with two hidden layers. The activation function of the second hidden layer was also sigmoid. Different networks from each configuration were trained:

- About 30 multilayer feed forward networks with one hidden layer were trained by varying the number of neurons in the hidden layer from 5 to 35.
- About 20 multilayer feed forward networks with two hidden layers were trained by varying the number of neurons from 10 to 25 in the first hidden layer and from 5 to 15 in the second hidden layer.

All these networks were trained over 1000 training epochs with Bayesian regularization training. Each network was evaluated by comparing the predictions to the true outputs, resulting in a prediction error for each orientation angle. The autocorrelation coefficients were also computed for the prediction error in each angle.

TABLE I
Performance of Multilayer Feed Forward Networks

Network Structure	Multilayer Feed Forward One Hidden Layer				
Network Performance	No. of Hidden Layer Neurons	Training Time (sec)	MSE	SSE	MAE
	S=27	7.3e3	2.8e-5	0.644	0.0037
	S=29	8.2e3	2.9e-5	0.66	0.0035
	S=30	8.6e3	1.9e-5	0.428	0.0028
	S=34	1e4	1.1e-5	0.242	0.0022
	S=35	1.1e4	1.1e-5	0.26	0.0022
Network Structure	Multilayer Feed Forward Two Hidden Layers				
Network Performance	No. of Hidden Layer Neurons	Training Time (sec)	MSE	SSE	MAE
	S1=10 S2=15	9.5e3	6.8e-6	0.154	0.0018
	S1=12 S2=15	2.9e4	2.8e-6	0.062	0.0011
	S1=17 S2=15	6.1e4	8.1e-7	0.018	6e-4
	S1=17 S2=9	1e4	5.6e-6	0.12	0.0016
	S1=17 S2=12	2.3e4	1.9e-6	0.044	9e-4

Using the whole stated criteria, five networks with best performance were selected from each configuration. Table (1) summarizes the performance of these networks. It can be seen that networks with two hidden layers have a better performance in general. It should be also noted that the mean square of error is approximately equal to the square of the maximum error, so a mean square error of 1e-5 will correspond to about 0.18 degree of accuracy for the forward kinematics solution. All the trainings and simulations of the neural networks were done on a Pentium4, 2 GHz using MATLAB® R14 software.

B. Radial Basis Function neural network

Radial basis function (RBF) neural network architecture was tested as another choice for computing the forward kinematics of the hydraulic shoulder. In general, RBF networks require more neurons but much less training time than MLP networks. Input and output patterns were generated in a same procedure as in the multilayer feed forward network. Supervised learning method was used in a way to reduce the estimated error of the network. Other specifications such as weight initialization, network evaluation and performance indices were just the same as the multilayer feed forward network. About ten different configurations with different spread parameters were trained and compared; from which two networks with best performance were selected. The performance of these networks is shown in Table (2). From the comparison of the selected structures in table (1) and (2) which are highlighted in gray, the multilayer feed forward with two hidden layers provides better approximation, with a training mean square error of 2.8e-6, and mean absolute error of 0.0011.

TABLE II
PERFORMANCE OF RBF NETWORKS

Network Performance	Training Time (sec)	MSE	SSE	MAE
RBF1	750	1.3e-5	0.1	0.0019
RBF2	680	9.9e-6	0.074	0.0017

C. Polynomial Neural Network Estimation

The Group Method of Data Handling (GMDH) has been known as one of the first approaches in design of nonlinear relationships. It was developed in the late 60s by Ivakhnenko [14] as a tool for identifying nonlinear maps by generating an optimal structure of the model through successive generations of partial descriptions (PDs) of data being regarded as quadratic regression polynomials with two input variables. This method, having a limited generic structure (quadratic polynomial with two variables) tends to result in very complex models for highly nonlinear systems as in our case. Polynomial Neural Networks (PNN) has been introduced in literature based on the paradigm of GMDH algorithm and has shown to be a useful data analysis technique for the identification of nonlinear complex systems [15]. This is a multilayered network with a self-organizing structure in contrast to classical networks with a fixed structure. In this network each node (processing element forming a PD) can have a different number of input variables or a different order of the polynomial (linear, quadratic, cubic, etc.) which results in a high level of flexibility. Its final topology is synthesized during the learning phase where contributing nodes are retained based on their performance, so the network becomes fully optimized (both structurally and parametrically) during learning process. Figure (3) shows a general structure of such a network used to identify the forward kinematics of the hydraulic shoulder.

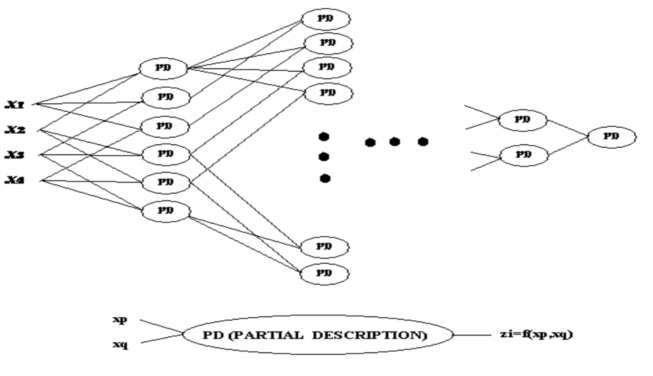

Fig. 3. A general structure for a polynomial neural network

As stated, the PNN algorithm uses a class of polynomials with different orders. The network is initiated with just one layer. Additional layers are generated until the best performance of the extended model is reached, which results in an optimal structure. The output is estimated by constructing a PD for each pair of input variables in the first layer.The parameters of each of the PDs are determined by the least square method using given training data and comparing their output to the desired network output. If none of the PDs reaches the performance criterion (which is generally the case for highly nonlinear functions) a new layer is added to the network. For this layer, new PDs are constructed using intermediate variables which are the outputs of the PDs in the previous layer.

The optimal coefficients of each of the PDs in this new layer are computed by least squares method. The operation is repeated until the stopping criterion has been satisfied, that is, the output of one of the PDs in the last layer reaches the desired performance. Since the number of PDs can increase exponentially, in each layer only PDs with better performance are retained to construct the next layer and other PDs are removed. Once the final layer has been constructed, the node with the best performance is selected as the output node and all remaining nodes in that layer are ignored. Furthermore, all the nodes of previous layers that do not have influence on the estimated output are also removed by tracing the data flow path of each layer. The design procedure can be summarized in the following steps:

1. Determine system’s input variables.
2. Form train and test data.
3. Choose the structure of the PNN by selecting the order of the polynomial forming a PD of data.
4. Estimate the PD coefficients.
5. Select PD with the best predictive performance.
6. Check the stopping criterion. Stop if it is satisfied.
7. Determine new inputs for the next layer.
8. Repeat steps 4-8.

This design procedure is applied to identify the forward kinematic map of the hydraulic shoulder. The input variables are the four actuator lengths of the manipulator as before. Three separate networks are used for each output angle which causes different network structure for each output. This ends to smaller structure for each output since their solutions are not bound to each other. A generic type of polynomial neural network [15] as in figure (3) was used with the number of layers and the number of remaining PDs in each layer as design variables. About 15000 inputs were randomly generated in the workspace. The input data was divided into train and test data set. Different PNN structures were selected based on the number of the inputs and different orders of PDs in each layer. The number of layers (*numl*) and the remaining PDs in each layer (*rempd*) were increased up to 25 and 15 respectively. Different possible network structures were trained and the trained network performances were compared using different criteria for the prediction errors along each orientation angle. Table (3) summarizes the performance of four selected PNNs with best performance, where the types of the polynomials are defined as:

Bilinear=$c_0 + c_1x_1 + c_2x_2$

Biquadratic=Bilinear+$c_3x_1^2 + c_4x_2^2 + c_5x_1x_2$

Bicubic= Biquadratic+$c_6x_1^3 + c_7x_2^3 + c_8x_1^2x_2 + c_9x_1x_2^2$

Also, “Biquadratic to Bicubic” stands for the network in which the PDs in the first layer are Biquadratic and for the former layers are Bicubic.

TABLE III
PERFORMANCE OF POLYNOMIAL NETWORKS

Network Structure	Polynomial Neural Network				
Network Performance with *numl*=10 and *rempd*=14	Type of PD Polynomial	Training Time (sec)	Train MSE		
			θ_x	θ_y	θ_z
	BiLinear	32.828	2.7e-5	6.1e-4	1.8e-4
	Biquadratic	124.172	6.3e-8	8e-7	2.3e-7
	Bicubic	226.6	5.1e-9	2e-7	1.1e-7
	Biquadratic to Bicubic	227	2.9e-7	7.8e-7	7.3e-7

The selected structure is again highlighted with the mean square error in the order of 10^{-7} which shows better training errors compared to classical neural networks.

D. *Adaptive-network-based fuzzy inference system*

Adaptive-network-based fuzzy inference system (ANFIS) is a feedforward adaptive neural network which implies a fuzzy inference system through its structure and neurons [19]. In adaptive network each node performs a particular function. The links in adaptive networks only indicate the flow directions of the signals between nodes, that is, no weights are associated with the links. The structure of the network and the function of each node vary in each layer and node; depending on the overall function which the network is to carry out. Feedforward adaptive network is a superset of all kinds of feedforward neural networks like MLP and RBF. Its nodes are divided to nodes which their parameters are adapted and those which are fixed. The adaptation rule is basically back propagation learning rule, although LMS is also used specially in the last layer.

In this study, ANFIS is used to realize a Takagi-Sugeno (TS) type fuzzy inference system. If-then rules in TS fuzzy system are in the form:

If x_1 is A_1 and … x_n is A_n then $y=a_1x_1+…+a_nx_n+a_{n+1}$.

Fig. 4 shows the structure of an ANFIS for a TS fuzzy system with 2 input and 2 membership functions (MF) for each input. FIS consists of four possible rules. The first layer computes the membership degree of each input in its MFs. MF parameters in this layer may be trained using back propagation learning rule.

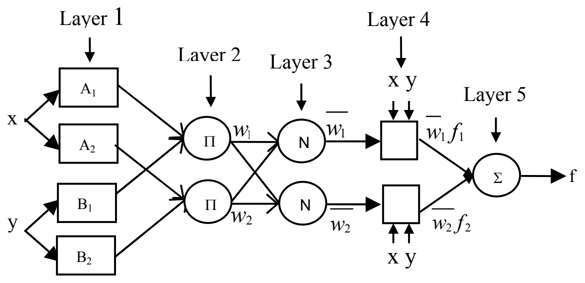

Fig.4. ANFIS structure for a TS fuzzy system

Neurons of the second layer combine their inputs by the *t*-norm and represent the firing strength of the rules. In the third layer the normalized firing strengths are computed using the following relation:

$$\overline{w}_i = \frac{w_i}{\sum_j w_j} \tag{9}$$

where w_i is firing strength of rule *i*. The normalized output of each rule is computed in the forth layer using parameters $a_j^i, i = 1,...m,\ j = 1,...,n$ where *m* is the number of rules. a_j^i 's are also design parameters. Using these normalized outputs of each rule, inferred output of TS fuzzy system is the sum of them which is obtained in the last layer.

Three separate ANFIS’s with similar structures are utilized to approximate the forward kinematics of the hydraulic shoulder. Each network has 4 inputs for the length of the legs. For each input, 3 bell shaped MFs are considered which are defined as:

$$\mu_{A_i^j}(x_i) = \frac{1}{1 + \left(\frac{x_i - c_i^j}{a_i^j}\right)^{2} b_i^j}, \quad i = 1,...,n,\ \ j = 1,...3 \tag{10}$$

where $\mu_{A_i^j}$ is the *j*th MF of the *i*th input and *a*, *b*, *c* are its parameters. Multiplication *t*-norm is used in the second layer and the parameters of the last layer are trained using least squares estimation. Each network is trained by 1500 training data which are generated randomly in the space [-π/6, +π/6] radian of each outputs. The networks are trained for just 2 epochs. Table (4) summarizes the simulation results of the training.

TABLE IV
PERFORMANCE OF ANFIS NETWORKS

	Training Time (sec)	Train MSE		
		θ_x	θ_y	θ_z
ANFIS	195	3.3e-6	5.1e-6	2.8e-6

V. COMPARATIVE STUDIES

A. Sample Trajectory Generation

We consider a smooth motion specified in terms of a desired pose of the moving platform of the hydraulic shoulder. The sample trajectory is easily defined given the initial and final points and the time to reach the final point. Fig. 5 shows the sample trajectory for each orientation angle in the task space of the hydraulic shoulder.

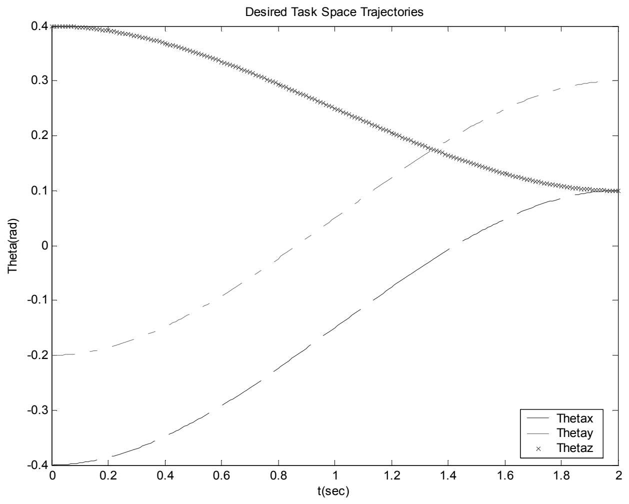

Fig. 5. Sample Trajectory for Orientation Angles

B. Simulations

Figs. 6-7 show the simulation results using the trained neural networks of different structures. Best representatives from each structure of MLP and RBF selected from tables (1) and (2) were tested with the sample trajectory along each orientation angle. Figure 8 shows the simulation results for the best representative of polynomial neural network applied to follow the sample trajectory. Simulation results show that ANFIS represents the best approximation, Fig. 9.

Table (5) summarizes the statistics of approximation errors, and the accuracies obtained by each method for the considered trajectory. As it is observed through this comparative study for the typical trajectory, the maximum approximation error reached by the suitable MLP and RBF structures are limited to 0.03 radians (1.7 degrees) and 0.1 radians (5.7 degrees) error respectively. PNN performs better approximation, especially along x and y directions, by a maximum error of 0.014 radians (0.8 degrees). As it can be seen the performance of the ANFIS is clearly better than that of other neural networks, with a maximum error of 0.0025 radians (0.15 degrees). This is quite adequate for a moderate precision robot, although it may be yet behind required accuracy in a very precise robotic application.

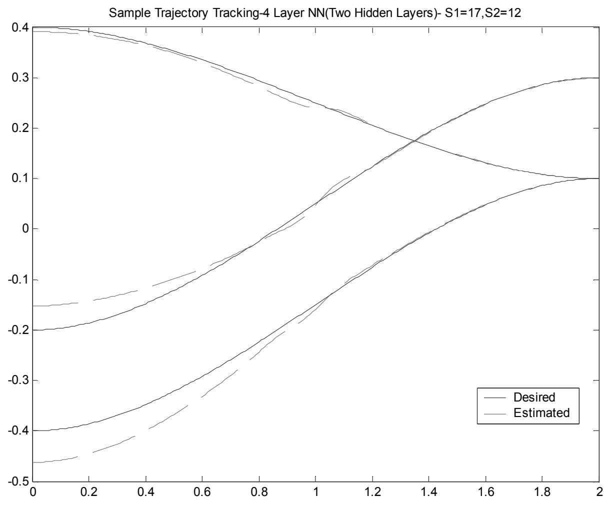

Fig. 6. Tracking Performance for selected MLP

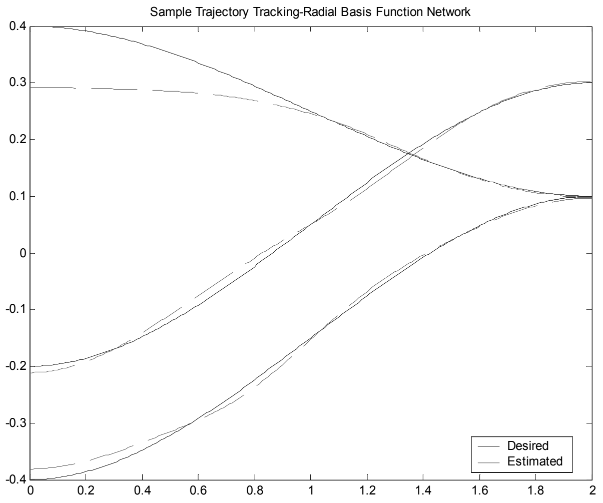

Fig. 7. Tracking Performance for selected RBF

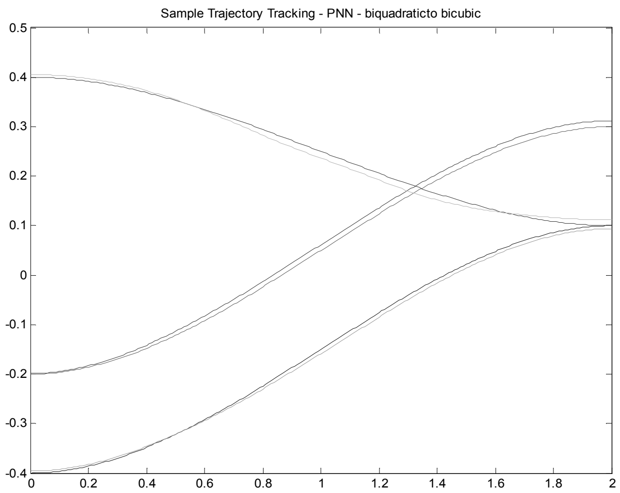

Fig. 8. Tracking Performance for selected PNN

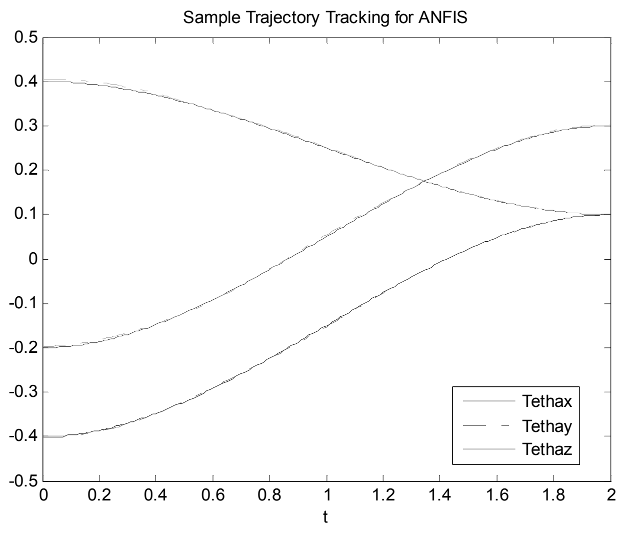

Fig. 9. Tracking Performance for ANFIS

TABLE IV
MEASURES OF TRACKING ERRORS (SI UNITS)

Performance Index / Solution Method		Emax	SSE	MSE	MAE
3 Layer Feed-forward Neural Net (s=34)	θ_x	0.054	0.124	6.1e-4	0.017
	θ_y	0.045	0.056	2.8e-4	0.011
	θ_z	0.03	.025	1.3e-4	0.007
4 layer FF Neural Net $s_1 = 12, s_2 = 15$	θ_x	0.028	0.032	1.6e-4	0.009
	θ_y	0.03	0.069	3.4e-4	0.014
	θ_z	0.032	0.054	2.7e-4	0.012
RBF Neural Network	θ_x	0.018	0.019	9.9e-5	0.008
	θ_y	0.017	0.016	8.3e-5	0.007
	θ_z	0.1	0.53	0.002	0.033
PNN Biquadratic/Bicubic *numl*=10 *rempd*=14	θ_x	0.009	0.008	4e-5	0.006
	θ_y	0.001	0.021	1e-4	0.01
	θ_z	0.014	0.017	8.6e-5	0.008
ANFIS (3 MF for each input)	θ_x	1.9e-3	2.7e-4	1.3e-6	9.3e-4
	θ_y	2.5e-3	5.7e-4	2.9e-6	1.6e-3
	θ_z	1.4e-3	6.1e-4	3.0e-6	1.3e-3

VI. CONCLUSION

In this paper, 4 different neural networks were studied to solve the forward kinematics problem in a three DOF actuator redundant hydraulic parallel manipulator. First, Two classical neural networks of different structures (MLP and RBF) were introduced to solve the problem. Simulation results showed that multilayer perceptron neural networks with two hidden layers had a better performance compared to those with one hidden layer in this application. The training time for RBF networks was shown to be much less than MLP networks. Their tracking performance and estimation errors were also acceptable, but the weak point of such networks could be the big size leading to large number of neurons and weights. The main drawback of these classical neural networks would be the long training times and the big size of the networks resulting in much more number of weights. Alternatively, polynomial neural network, which were introduced based on the paradigm of Group Method of Data Handling, is applied to solve the forward kinematic problem of this spatial parallel manipulator. It is observed that the polynomial network has better performance with acceptable prediction errors for general robotic applications with much less training time required compared to the above classical structures of neural networks. The best results are obtained by using ANFIS. ANFIS, which combines the specifications of fuzzy systems and neural networks, clearly outperforms the above neural networks. Its accuracy (around 0.15 degree) is in the range which is suitable for moderate precision robots. An interesting observation is that the speed of learning is with just 2 epochs for the above accuracy.

REFERENCES

[1] J.P. Merlet, Still a long way to go on the road for parallel mechanisms, ASME 2002 DETC Conference, Montreal, Canada, 2002. Available: http://www-sop.inria.fr.

[2] J.P. Merlet, Parallel Robots: Open problems, In 9th Int'l. Symp. of Robotics Research, Snowbird, 9-12 October 1999. Available: http://www-sop.inria.fr.

[3] O.Didrit, M.Petitot and E.Walter, Guaranteed solution of direct kinematic problems for general configurations of parallel manipulators, IEEE Trans. On Robotics & Automation, April 1998, 259-266.

[4] B. Dasgupta, T.S. Mruthyunjaya, The Stewart platform manipulator: a review, Elsevier Science, Mechanism & Machine theory,2000,15-40.

[5] Hayward, V.: "Design of a hydraulic robot shoulder based on a combinatorial mechanism" Experimental Robotics III: The 3rd Int'l Symposium, Japan Oct. 1994. Lecture Notes in Control & Information Sciences, Springer-Verlag, 297-310.

[6] Hayward, V.: "Borrowing some design ideas from biological manipulators to design an artificial one" in Robots and Biological System, NATO Series, Springer-Verlag, 1993, 135-148.

[7] Hayward, V. and Kurtz, R.: Modeling of a parallel wrist mechanism with actuator redundancy, Int'l. J. Laboratory Robotics and Automation, VCH Publishers, Vol. 4, No. 2.1992, 69-76.

[8] Z.Geng and L.Haynes, Neural network solution for the forward kinematics problem of a Stewart platform, Proc. Of the 1991 IEEE Int'l Conf. on Robotics & Automation, California, April 1991, 2650-2655.

[9] C.S.Yee and Kah-bin Lim, Forward kinematics solution of Stewart platform using neural networks, Elsevier Science, Neurocomputing 16, 1997, 333-349.

[10] Nguyen, L., Patel, R.V. and Khorasani, K.: Neural Network Architectures for the forward kinematics problem in robotics. In Proc. of the Joint IEEE International Conference on Neural Networks, San Diego, 1990, 393-399.

[11] D.Wang and A.Zilouchian, Solutions of kinematics of robot manipulators using a kohonen self organizing neural network, Proc. Of the 1997 IEEE Int'l Symp. on intelligent control, Turkey, July 1997, 251-255.

[12] L.H.Sang and M.C.Han, The Estimation for forward kinematic solution of Stewart platform using the neural network, Proc. Of the 1999 IEEE/RSJ Int'l Conf. on Intelligent Robots & Systems, 1999, 501-506.

[13] Lee, S. and Kil, R.M.: Robot kinematic control based on bidirectional mapping neural network. ," in Proc. IJCNN, San Diego, CA, Vol. 3, 1990, 327–335.

[14] Ivakhnenko AG. Polynomial theory of complex systems. IEEE Trans. Systems, Man, Cybernetics, 1971, SMC-1, 364-378.

[15] S.K. Oh, W. Pedrycz, B.J. Park, Polynomial neural networks architecture: analysis & design, Information Sciences 141, 2002, 237-258.

[16] C.Y. Tsai, An iterative feature reduction algorithm for probabilistic neural networks, the International Journal of Management Science, Omega 28, 2000, 513-524.

[17] C.L. Philip chen and A.D. Mc Aulary, Robot kinematics Learning computatons using polynomial neural networks, proceeding of the 1991 IEEE International conference on Robotics and Automation, 1991, 2638-2643.

[18] R. Boudreau, S. Darenfed, On the computation of the Direct kinematics of parallel Maniputators using polynomial networks, IEEE transactions on systems, man and cybernetics, Vol. 28, No. 2, March 1998, 213-220.

[19] J.R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," IEEE Transaction on systems, man and cybernetics, Vol. 23, No. 3, May/June 1993, 665-685.

Hooman Sadjadian received his B.Sc. degree in electrical engineering from Tehran University, Tehran, Iran, in 1995, his M.Eng in electrical engineering in 1999, from Iran University of Science and Technology, Tehran, Iran. He is currently a Ph.D. student with the Electrical Engineering Department, at K.N. Toosi University of Technology, Tehran, Iran. His research interest are Dynamics and Control of Parallel manipulators.

Hamid D. Taghirad received his B.Sc. degree in mechanical engineering from Sharif University of Technology, Tehran, Iran, in 1989, his M.Eng in mechanical engineering in 1993, and his Ph.D. in electrical engineering in 1997, both from McGill University, Montreal, Canada. He is currently an Assistant Professor with the Electrical Engineering Department, and the Director of the control group and the Advanced Robotics and Automated System, ARAS research center at K.N. Toosi University of Technology, Tehran, Iran. He becomes a member of IEEE in 1995. His publications include two books, and more than 70 papers in international Journals and conference proceeding, and his research interest are robust and nonlinear control applied on the robotic systems.

Alireza Fatehi received the B.S. degree in Electronics Engineering from the Isfahan University of Technology, in 1990, the M.S. degree in Control Engineering from Tehran University, Tehran, Iran, in 1995 and Ph.D. degree in Control Engineering from Tohoku University, Sendai, Japan, in 2001. He is currently an assistant professor of control engineering in Control Department of K.N. Toosi University of Technology and the head of Advance Automation and Digital Control Lab. His research interests include process control, intelligent control systems, industrial automation and multiple control systems.