We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

22

Neural Networks' Based Inverse Kinematics Solution for Serial Robot Manipulators Passing Through Singularities

Ali T. Hasan[1], Hayder M.A.A. Al-Assadi[2] and Ahmad Azlan Mat Isa[2]
[1]*Department of Mechanical and Manufacturing Engineering*
University Putra Malaysia, 43400UPM,Serdang, Selangor,
[2]*Faculty of Mechanical Engineering, University Technology MARA (UiTM)*
40450 Shah Alam,
Malaysia

1. Introduction

Before moving a robot arm, it is of considerable interest to know whether there are any obstacles present in its path. Computer-based robots are usually served in the joint space, whereas objects to be manipulated are usually expressed in the Cartesian space because it is easier to visualize the correct end-effector position in Cartesian coordinates than in joint coordinates. In order to control the position of the end-effector of the robot, an *inverse kinematics (IK)* solution routine should be called upon to make the necessary conversion (Fu et al., 1987).

Solving the inverse kinematics problem for serial robot manipulators is a difficult task; the complexity in the solution arises from the nonlinear equations (trigonometric equations) occurring during transformation between joint and Cartesian spaces. The common approach for solving the IK problem is to obtain an analytical close-form solution to the inverse transformation, unfortunately, close-form solution can only be found for robots of simple kinematics structure. For robots whose kinematics structures are not solvable in close-form; several numerical techniques have been proposed; however, there still remains several problems in those techniques such as incorrect initial estimation, convergence to the correct solution can not be guarantied, multiple solutions may exist and no solution could be found if the Jacobian matrix was in singular configuration (Kuroe et al., 1994; Bingual et al., 2005).

A velocity singular configuration is a configuration in which a robot manipulator has lost at least one motion degree of freedom DOF. In such configuration, the inverse Jacobian will not exist, and the joint velocities of the manipulator will become unacceptably large that often exceed the physical limits of the joint actuators. Therefore, to analyze the singular conditions of a manipulator and develop effective algorithm to resolve the inverse kinematics problem in the singular configurations is of great importance (Hu et al., 2002).

Many research efforts have been devoted towards solving this problem, one of the first algorithms employed was the Resolved Motion Rate-Control method (Whitney, 1969), which uses the pseudoinverse of the Jacobian matrix to obtain the joint velocities corresponding to a given end-effector velocity, an important drawback of this method was

the singularity problem. To overcome the problem of kinematics singularities, the use of a damped least squares inverse of the Jacobian matrix has been later proposed in lieu of the pseudoinverse (Nakamura & Hanafusa, 1986; Wampler, 1986).
Since in the above algorithmic methods the joint angles are obtained by numerical integration of the joint velocities, these and other related techniques suffer from errors due to both long-term numerical integration drift and incorrect initial joint angles. To alleviate the difficulty, algorithms based on the feedback error correction are introduced (Wampler & Leifer, 1988). However, it is assumed that the exact model of manipulator Jacobian matrix of the mapping from joint coordinate to Cartesian coordinate is exactly known. It is also not sure to what extent the uncertainty could be allowed. Therefore, most research on robot control has assumed that the exact kinematics and Jacobian matrix of the manipulator from joint space to Cartesian space are known. This assumption leads to several open problems in the development of robot control laws (Antonelli et al., 2003).
Intelligent control has been introduced as a new direction making control systems able to attribute more intelligence and high degree of autonomy. Artificial Neural Networks (ANN) have been widely used for their extreme flexibility due to the learning ability and the capability of non linear function approximation, a number of realistic control approaches have been proposed and justified for applications to robotic systems (D'Souza et al.,2001; Ogawa et al., 2005; Köker, 2005; Hasan et al., 2007; Al-Assadi et al., 2007), this fact leads to expect ANN to be an excellent tool for solving the IK problem for serial manipulators overcoming the problems arising.
Studying the IK of a serial manipulator by using ANNs has two problems, one of these is the selection of the appropriate type of network and the other is the generating of suitable training data set (Funahashi, 1998;Hasan et al., 2007).
Different methods for gathering training data have been used by many researchers, while some of them have used the kinematics equations (Karilk & Aydin, 2000; Bingual et al., 2005), others have used the network inversion method (Kuroe et al., 1994; Köker, 2005), another have used the cubic trajectory planning (Köker et al., 2004) and others have used a simulation program for this purpose (Driscoll, 2000). However, there are always kinematics uncertainties presence in the real world such as ill-defined linkage parameters, links flexibility and backlashes in gear train.
A learning method of a neural network has been proposed by (Kuroe et al., 1994), such that the network represents the relations of both the positions and velocities from the Cartesian coordinate to the joint space coordinate. They've driven a learning algorithm for arbitrary connected recurrent networks by introducing adjoin neural networks for the original neural networks (Network inversion method). On-line training has been performed for a 2 DOF robot.
(Graca and Gu, 1993) have developed a Fuzzy Learning Control algorithm. Based on the robotic differential motion procedure, the Jacobian inverse has treated as a fuzzy matrix and has learned through the fuzzy regression process. It was significant that the fuzzy learning control algorithm neither requires an exact kinematics model of a robotic manipulator, nor a fuzzy inference engine as is typically done in conventional fuzzy control. Despite the fact that unlike most learning control algorithms, multiple trials are not necessary for the robot to "learn" the desired trajectory. A major drawback was that it only remembers the most recent data points introduced, the researchers have recommended neural networks so that it would remember the trajectories as it traversed them.

The solution of the Inverse kinematics, which is mainly solved in this chapter, involves the development of two network's configurations to examine the effect of the Jacobian Matrix in the solution.
Although this is very difficult in practice (Hornik, 1991), training data were recorded experimentally from sensors fixed on each joint (as was recommended by (Karilk and Aydin, 2000). Finally the obtained results were verified experimentally.

2. Kinematics of serial robots

For serial robot manipulators, the vector of Cartesian space coordinates x is related to the joint coordinates q by:

$$x = f(q) \tag{1}$$

Where $f(\cdot)$ is a non-linear differential function.
If the Cartesian coordinates x were given, joint coordinates q can be obtained as:

$$q = f^{-1}(x) \tag{2}$$

Using ANN to solve relation (2), for getting joint position q, mapping from the joint space to the Cartesian space is uniquely decided when the end effector's position is calculated using direct kinematics (Köker et al., 2004; Ogawa et al., 2005; Hasan et al., 2006), as shown in Figure 1(a). However, the transformation from the Cartesian to the joint space is not uniquely decided in the inverse kinematics as shown in Figure 1(b).

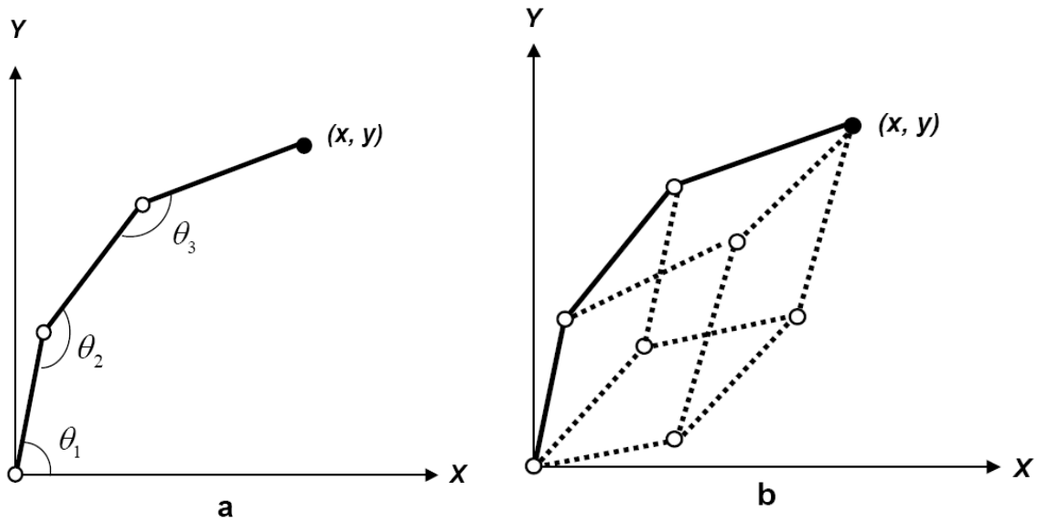


Fig. 1. Three DOF robot arm.
a) Joint angles and end-effector's coordinates (forward kinematics).
b) Combination of all possible joint angles (Inverse Kinematics).

Model-based methods for solving the IK problem are inadequate if the structure of the robot is complex, therefore; techniques mainly based on inversion of the mapping established between the joint space and the task space of the manipulator's Jacobian matrix have been proposed for those structures that cannot be solved in closed form.

If a Cartesian linear velocity is denoted by V, the joint velocity vector $\dot{q}$ has the following relation:

$$V = J\dot{q} \tag{3}$$

Where J is the Jacobian matrix.
If V, is a desired Cartesian velocity vector which represents the linear velocity of the desired trajectory to be followed. Then, the joint velocity vector $\dot{q}$ can be resolved by:

$$\dot{q} = J^{-1}V \tag{4}$$

At certain manipulator configurations, the Jacobian matrix may lose its full rank. Hence as the manipulator approaches these configurations (singular configurations), the Jacobian matrix becomes ill conditioned and may not be invertible. Under such a condition, numerical solution for equation (4) results in infinite joint rates.
In differential motion control, the desired trajectory is subdivided into sampling points separated by a time interval Δt between two terminal points of the path. Assuming that at time t_i the joint positions take on the value $q(t_i)$, the required q at time $(t_i + \Delta t)$ is conventionally updated by using:

$$q(t_i + \Delta t) = q(t_i) + \dot{q}\,\Delta t \tag{5}$$

Substituting Eqns. (2) and (4) into (5) yields:

$$q(t_i + \Delta t) = f^{-1}(x)(t_i) + J^{-1}V\Delta t \tag{6}$$

Equation (6) is a kinematics control law used to update the joint position q and is evaluated on each sampling interval. The resulting $q(t_i + \Delta t)$ is then sent to the individual joint motor servo-controllers, each of which will independently drive the motor so that the manipulator can be maneuvered to follow the desired trajectory (Graca & Gu, 1993).

3. Data collection procedure

Trajectory planning was performed to generate the angular position and velocity for each joint, and then these generated data were fed to the robot's controller to generate the corresponding Cartesian position and linear velocity of the end-effector, which were recorded experimentally from sensors fixed on the robot joints.
In details, trajectory planning was performed using cubic trajectory planning method .In trajectory planning of a manipulator, it is interested in getting the robot from an initial position to a target position with free of obstacles path. Cubic trajectory planning method has been used in order to find a function for each joint between the initial position, θ_0, and final position, θ_f of each joint.
It is necessary to have at least four-limit value on the $\theta(t)$ function that belongs to each joint, where $\theta(t)$ denotes the angular position at time t.
Two limit values of the function are the initial and final position of the joint, where:

$$\theta(0) = \theta_0 \tag{7}$$

$$\theta(t_f) = \theta_f \tag{8}$$

Additional two limit values, the angular velocity will be zero at the beginning and the target position of the joint, where:

$$\dot{\theta}(0) = 0 \tag{9}$$

$$\dot{\theta}(t_f) = 0 \tag{10}$$

Based on the constrains of typical joint trajectory listed above, a third order polynomial function can be used to satisfy these four conditions; since a cubic polynomial has four coefficients.
These conditions can determine the cubic path, where a cubic trajectory equation can be written as:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{11}$$

The angular velocity and acceleration can be found by differentiation, as follows:

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{12}$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t \tag{13}$$

Substituting the constrains conditions in the above equations results in four equations with four unknowns:

$$\begin{aligned} \theta_0 &= a_0, \\ \theta_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3, \\ 0 &= a_0, \\ 0 &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{aligned} \tag{14}$$

The coefficients are found by solving the above equations.

$$\begin{aligned} a_0 &= \theta_0, \\ a_1 &= 0, \\ a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0), \\ a_3 &= \frac{-2}{t_f^3}(\theta_f - \theta_0) \end{aligned} \tag{15}$$

Angular position and velocity can be calculated by substituting the coefficients driven in Eqn. (15) into the cubic trajectory Eqns. (11) and (12) respectively (Köker et al., 2004), which yield:

$$\theta_i(t) = \theta_{i0} + \frac{3}{t_f^2}(\theta_{if} - \theta_{i0})t^2 - \frac{2}{t_f^3}(\theta_{if} - \theta_{i0})t^3, \tag{16}$$

$$\dot{\theta}_i(t) = \frac{6}{t_f^2}(\theta_{if} - \theta_{i0})t - \frac{6}{t_f^3}(\theta_{if} - \theta_{i0})t^2 \tag{17}$$

$i = 1, 2, ..., n$ Where n is the joint number.

Joint angles generated ranged from amongst all the possible joint angles that do not exceed the physical limits of each joint. Trajectory used for the training process has meant to be random trajectory rather than a common trajectory performed by the robot in order to cover as most space as possible of the robot's working cell.
The interval of 1 second was used between a trajectory segment and another where the final position for one segment is going to be the initial position for the next segment and so on for every joint of the six joints of the robot.
After generating the joint angles and their corresponding angular velocities, these data are fed to the robot controller, which is provided with a sensor system that can detect the angular position and velocity on one hand and the Cartesian position and the linear velocity of the end-effector on the other hand; which are recorded to be used for the networks' training. As these joints are moving simultaneously with each other to complete the trajectory together.

4. Artificial Neural Networks

Artificial neural networks (ANNs) are collections of small individual interconnected processing units. Information is passed between these units along interconnections. An incoming connection has two values associated with it, an input value and a weight. The output of the unit is a function of the summed value. ANNs while implemented on computers are not programmed to perform specific tasks. Instead, they are trained with respect to data sets until they learn the patterns presented to them. Once they are trained, new patterns may be presented to them for prediction or classification (Kalogirou, 2001).
The elementary nerve cell called a neuron, which is the fundamental building block of the biological neural network. Its schematic diagram is shown in Figure 2.
A typical cell has three major regions: the cell body, which is also called the soma, the axon, and the dendrites. Dendrites form a dendritic tree, which is a very fine bush of thin fibbers around the neuron's body. Dendrites receive information from neurons through axons-Long fibbers that serve as transmission lines. An axon is a long cylindrical connection that carries impulses from the neuron. The end part of an axon splits into a fine arborization. Each branch of it terminates in a small end bulb almost touching the dendrites of neighbouring neurons. The axon-dendrite contact organ is called a *synapse*. The synapse is where the neuron introduces its signal to the neighbouring neuron (Zurada, 1992; Hasan et al., 2006), to stimulate some important aspects of the real biological neuron. An ANN is a group of interconnected artificial neurons usually referred to as "node" interacting with one another in a concerted manner; Figure 3 illustrates how information is processed through a single

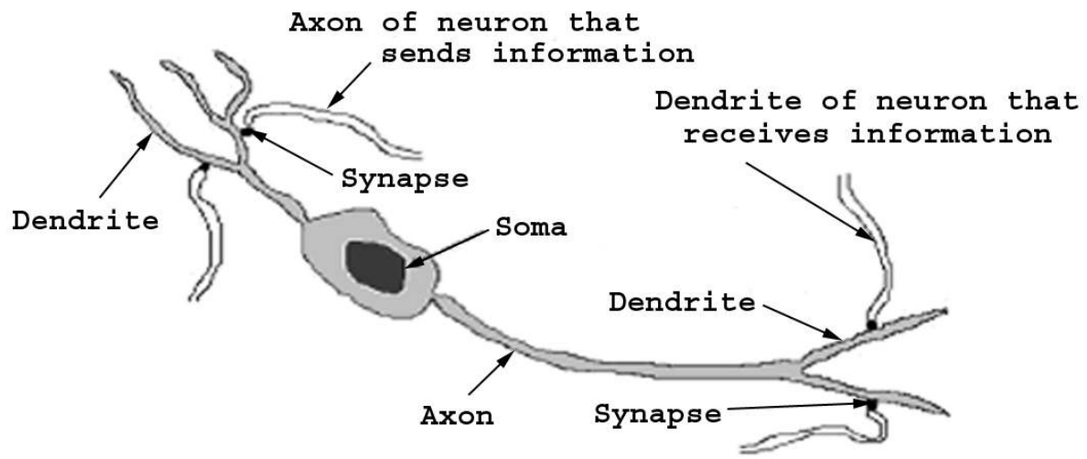


Fig. 2. Schematic diagram for the biological neuron

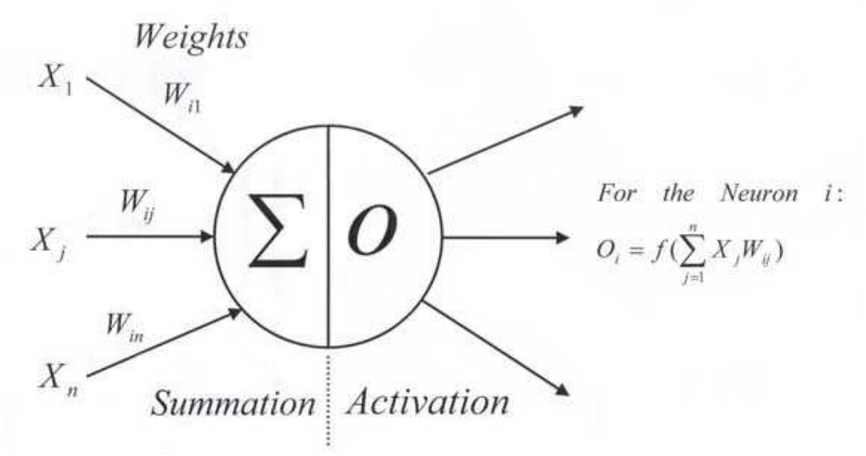


Fig. 3. Information processing in the neural unit

node. The node receives weighted activation of other nodes through its incoming connections. First, these are added up (summation). The result is then passed through an activation function and the outcome is the activation of the node. The activation function can be a threshold function that passes information only if the combined activity level reaches a certain value, or it could be a continues function of the combined input, the most common to use is a sigmoid function for this purpose. For each of the outgoing connections, this activation value is multiplied by the specific weight and transferred to the next node (Kalogirou, 2001; Hasan et al., 2006).

An artificial neural network consists of many nods joined together usually organized in groups called 'layers', a typical network consists of a sequence of layers with full or random connections between successive layers as Figure 4 shows. There are typically two layers with connection to the outside world; an input buffer where data is presented to the network, and an output buffer which holds the response of the network to a given input pattern, layers distinct from the input and output buffers called 'hidden layer', in principle there could be more than one hidden layer, In such a system, excitation is applied to the input layer of the network.

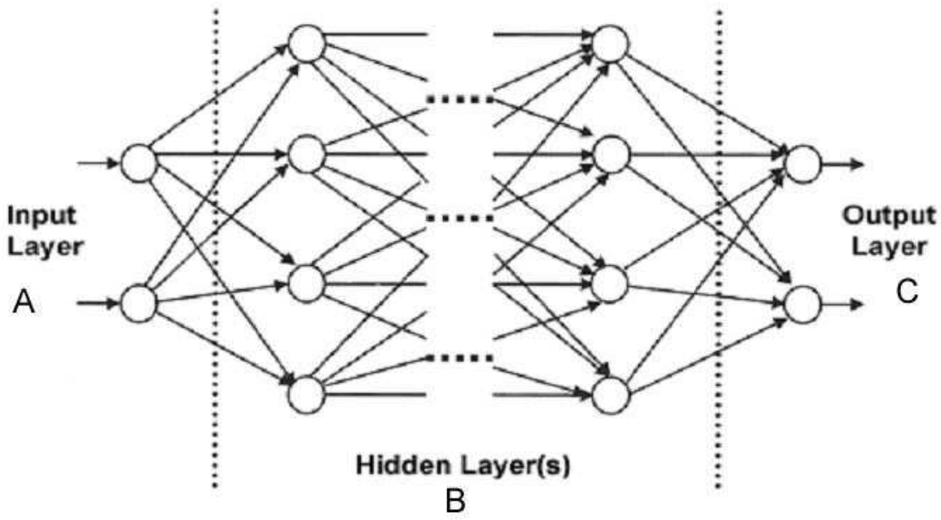


Fig. 4. Schematic diagram of a multilayer feedforward neural network

Following some suitable operation, it results in a desired output. Knowledge is usually stored as a set of connecting weights (presumably corresponding to synapse efficiency in biological neural system) (Santosh et al., 1993). A neural network is a massively parallel-distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the human brain in two respects; the knowledge is acquired by the network through a learning process, and interneuron connection strengths known as synaptic weights are used to store the knowledge (Haykin, 1994).

Training is the process of modifying the connection weights in some orderly fashion using a suitable learning method. The network uses a learning mode, in which an input is presented to the network along with the desired output and the weights are adjusted so that the network attempts to produce the desired output. Weights after training contain meaningful information whereas before training they are random and have no meaning (Kalogirou, 2001).

Two different types of learning can be distinguished: supervised and unsupervised learning, in supervised learning it is assumed that at each instant of time when the input is applied, the desired response d of the system is provided by the teacher. This is illustrated in Figure 5-a. The distance ρ [d,o] between the actual and the desired response serves as an error measure and is used to correct network parameters externally. Since adjustable weights are assumed, the teacher may implement a reward-and-punishment scheme to adopt the network's weight. For instance, in learning classifications of input patterns or situations with known responses, the error can be used to modify weights so that the error decreases. This mode of learning is very pervasive.

Also, it is used in many situations of learning. A set of input and output patterns called a training set is required for this learning mode. Figure 5-b shows the block diagram of unsupervised learning. In unsupervised learning, the desired response is not known; thus, explicit error information cannot be used to improve network's behaviour. Since no information is available as to correctness or incorrectness of responses, learning must somehow be accomplished based on observations of responses to inputs that we have marginal or no knowledge about (Zurada, 1992).

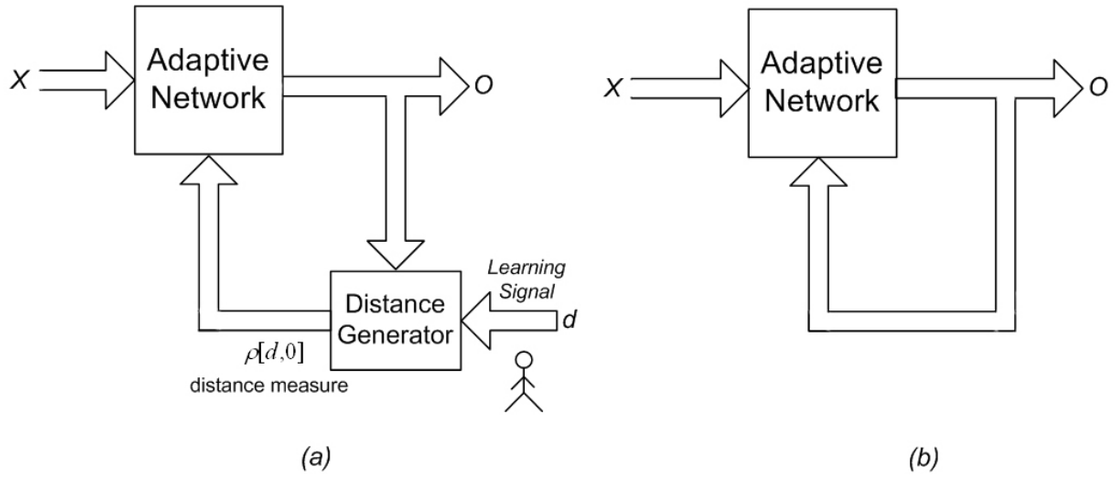


Fig. 5. Basic learning modes

The fundamental idea underlying the design of a network is that the information entering the input layer is mapped as an internal representation in the units of the hidden layer(s) and the outputs are generated by this internal representation rather than by the input vector. Given that there are enough hidden neurons, input vectors can always be encoded in a form so that the appropriate output vector can be generated from any input vector (Santosh et al., 1993).

As it can be seen in figure 4, the output of the units in layer ***A*** (Input Layer) are multiplied by appropriate weights W_{ij} and these are fed as inputs to the hidden layer. Hence if O_i are the output of units in layer ***A,*** then the total input to the hidden layer, i.e., layer ***B*** is:

$$Sum_B = \sum_i O_i W_{ij} \tag{18}$$

And the output O_j of a unit in layer ***B*** is:

$$O_j = f(sum_B) \tag{19}$$

Where *f* is the non-linear activation function, it is a common practice to choose the sigmoid function given by:

$$f(O_j) = \frac{1}{1+e^{-O_j}} \tag{20}$$

As the nonlinear activation function.

However, any input-output function that possesses a bounded derivative can be used in place of the sigmoid function. If there is a fixed, finite set of input-output pairs, the total error in the performance of the network with a particular set of weights can be computed by comparing the actual and the desired output vectors for each presentation of an input vector. The error at any output unit e_K in the layer **C** can be calculated by: -

$$e_K = d_K - O_K \tag{21}$$

Where d_K is the desired output for that unit in layer **C** and O_K is the actual output produced by the network .the total error ***E*** at the output can be calculated by:

$$E = \frac{1}{2}\sum_{K}(d_K - O_K)^2 \tag{22}$$

Learning comprises changing weights so as to minimize the error function and to minimize ***E*** by the gradient descent method. It is necessary to compute the partial derivative of ***E*** with respect to each weight in the network. Equations (19) and (19) describe the forward pass through the network where units in each layer have there states determined by the inputs they received from units of lower layer. The backward pass through the network that involves "back propagation " of weight error derivatives from the output layer back to the input layer is more complicated. For the sigmoid activation function given in Equation (20), the so-called delta-rule for iterative convergence towards a solution maybe stated in general as:

$$\Delta W_{JK} = \eta \delta_K O_J \tag{23}$$

Where η is the learning rate parameter, and the error δ_K at an output layer unit K is given by:

$$\delta_K = O_K(1 - O_K)(d_K - O_K) \tag{24}$$

And the error δ_J at a hidden layer unit is given by:

$$\delta_J = O_J(1 - O_J)\sum_{K}\delta_K W_{JK} \tag{25}$$

Using the generalize delta rule to adjust weights leading to the hidden units is back propagating the error-adjustment, which allows for adjustment of weights leading to the hidden layer neurons in addition to the usual adjustments to the weights leading to the output layer neurons. A back propagation network trains with two step procedures as it is

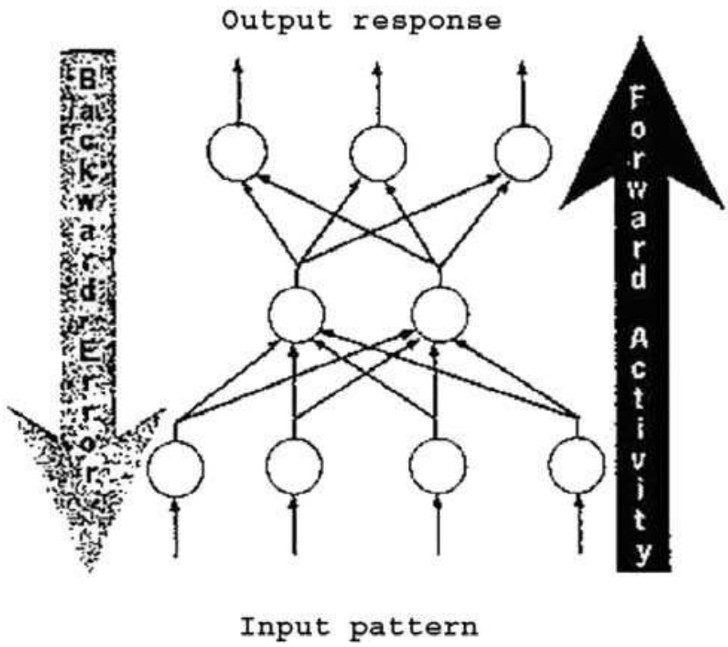


Fig. 6. Information flow through a backpropagation network

shown in figure 6, the activity from the input pattern flows forward through the network and the error signal flows backwards to adjust the weights using the following equations:

$$W_{IJ} = W_{IJ} + \eta \delta_J O_I \tag{26}$$

$$W_{JK} = W_{JK} + \eta \delta_K O_J \tag{27}$$

Until for each input vector the output vector produced by the network is the same as (or sufficiently close to) the desired output vector (Santosh et al., 1993).
ANNs while implemented on computers are not programmed to perform specific tasks. Instead, they are trained with respect to data sets until they learn the patterns presented to them. Once they are trained, new patterns may be presented to them for prediction or classification (Kalogirou, 2001).

5. ANN implementation

Two supervised feedforward ANN have been designed using C programming language to overcome the singularities and uncertainties in the arm configurations. Both networks consist of input, output and one hidden layer, every neuron in each of the networks is fully connected with each other, sigmoid transfer function was chosen to be the activation function, generalized backpropagation delta learning rule (GDR) algorithm was used in the training process.
Off-line training was implemented; Trajectory planning was performed for 600 data set for every 1-second interval from amongst all the possible joint angles in the robot's workspace, then data sets were recorded experimentally from sensors fixed on the robot joints as was recommended by (Karilk and Aydin, 2000), 400 set were used for the training while the other 200 sets were used for the testing the network.
All input and output values are usually scaled individually such that overall variance in data set is maximized, this is necessary as it leads to faster learning, all the vectors were scaled to reflect continuous values ranges from –1 to 1.
FANUC M-710i robot was used in this study, which is a serial robot manipulator consisting of axes and arms driven by servomotors. The place at which arm is connected is a joint, or an axis. This type of robot has three main axes; the basic configuration of the robot depends on whether each main axis functions as a linear axis or rotation axis. The wrist axes are used to move an end effecter (tool) mounted on the wrist flange. The wrist itself can be wagged about one wrist axis and the end effecter rotated about the other wrist axis, this highly non-linear structure makes this robot very useful in typical industrial applications such as the material handling, assembly of parts and painting.
The networks' implementation carried out on two phases, the first was the training phase where the performance of the two networks were compared, and then the network that has shown better response has been chosen to apply the testing data during the testing phase, which has been implemented through two stages. The first stage was the simulation then the results were verified experimentally.

5.1 Training phase

To examine the effect of considering the Jacobian Matrix for the Inverse Kinematics solution two networks have been designed and compared. ANN technique has been utilized where

learning is only based on observation of the input output relationship unlike other schemes that require an explicit system model.

5.1.1 The first configuration (3 - 6 configuration)

As in our previous research (Hasan et al., 2006; Hasan et al., 2007), the input vector for the network consists of the position of the end effector of the robot along the X, Y and Z coordinates of the global coordinate system, while the output vector was the angular position of each of the 6 joints as can be seen in Figure 7.

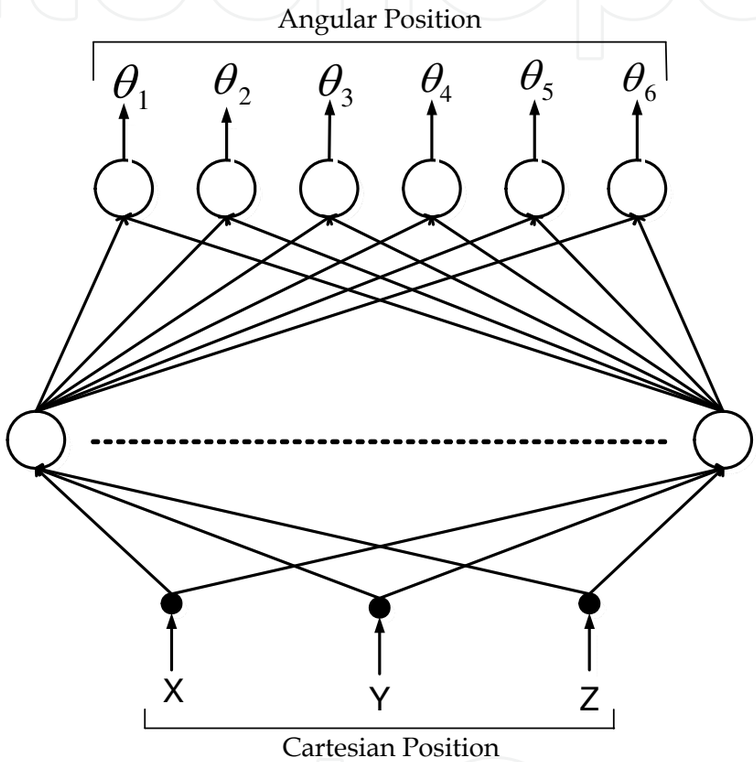


Fig. 7. The topology of the first configuration network

Although the number of training patterns was doubled, number of neurons in the hidden layer still the same as previous which is 43 only a little difference in the learning factor was experienced which was 0.5 in this case by trial and error.

Figure 8 shows the building knowledge curve for this configuration while Table 1 shows the percentage of error of each of the 6 joints after the training was finished after 1.5 million iterations.

Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
13.193%	10.855%	3.478%	14.748%	12.243%	8.348%

Table 1. Error percentages obtained after training (first configuration)

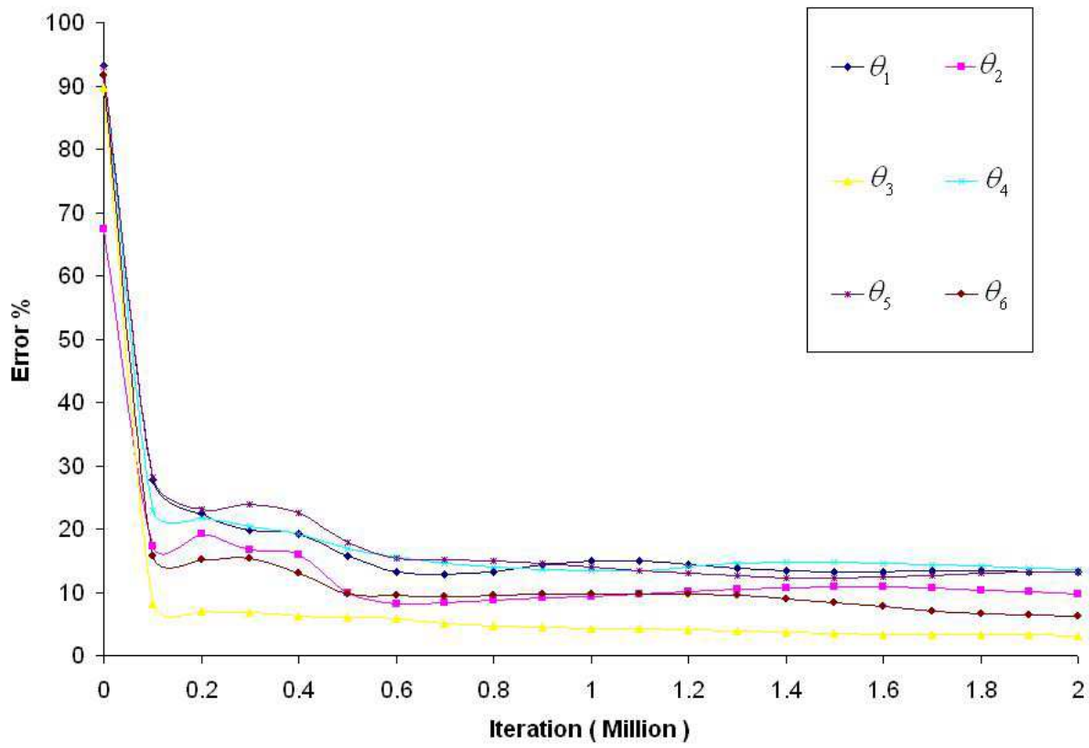


Fig. 8. The learning curve for the first configuration

5.1.2 The second configuration (4 – 12 configuration)

To examine the effect of considering the Jacobian matrix to the IK solution, another network has been designed, as in Figure 9, the new network consists of the Cartesian Velocity added to the input buffer and the angular velocity of each of the 6 joints added to the output buffer of the previous network.

Number of the neurons in the hidden layer was set to be 77 with constant learning factor of 0.9 by trial and error.

Figure 10 shows the building knowledge curve while table 2 shows the percentage of error of each of the 6 joints after the training was finished after 1.5 million iterations.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Angular Position	2.817 %	1.645%	0.88%	3.163%	3.125%	2.09%
Angular Velocity	2.65%	3.018%	1.683%	3.208%	2.525%	1.6%

Table 2. Error percentages obtained after training (second configuration)

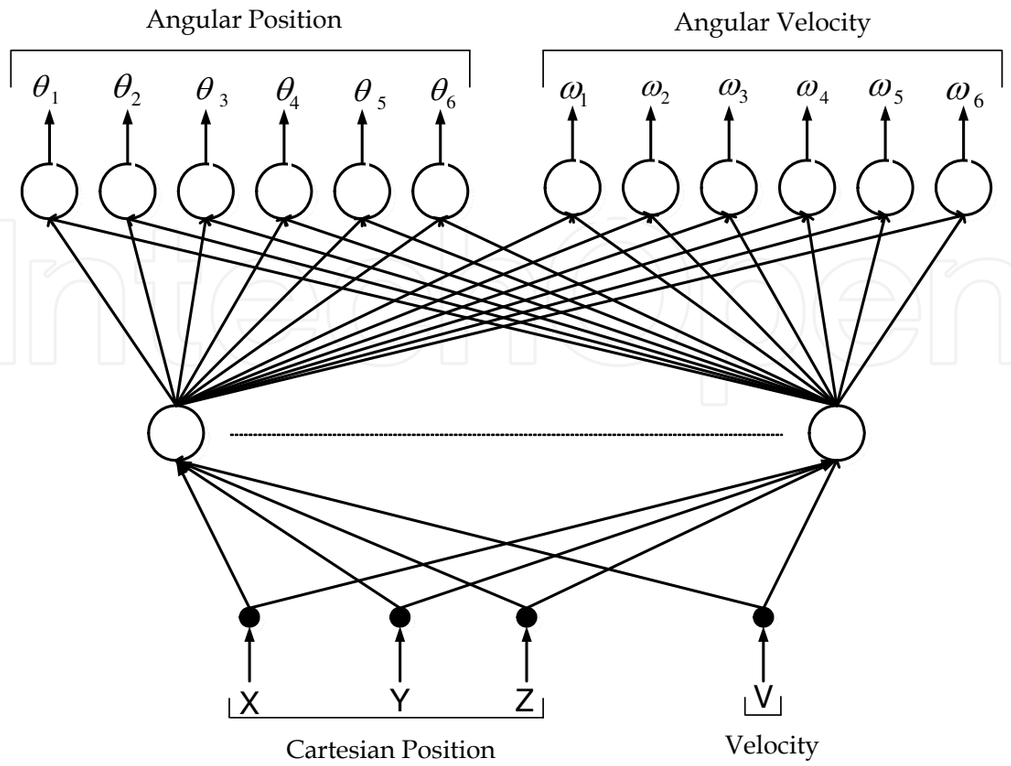


Fig. 9. The topology of the second configuration network

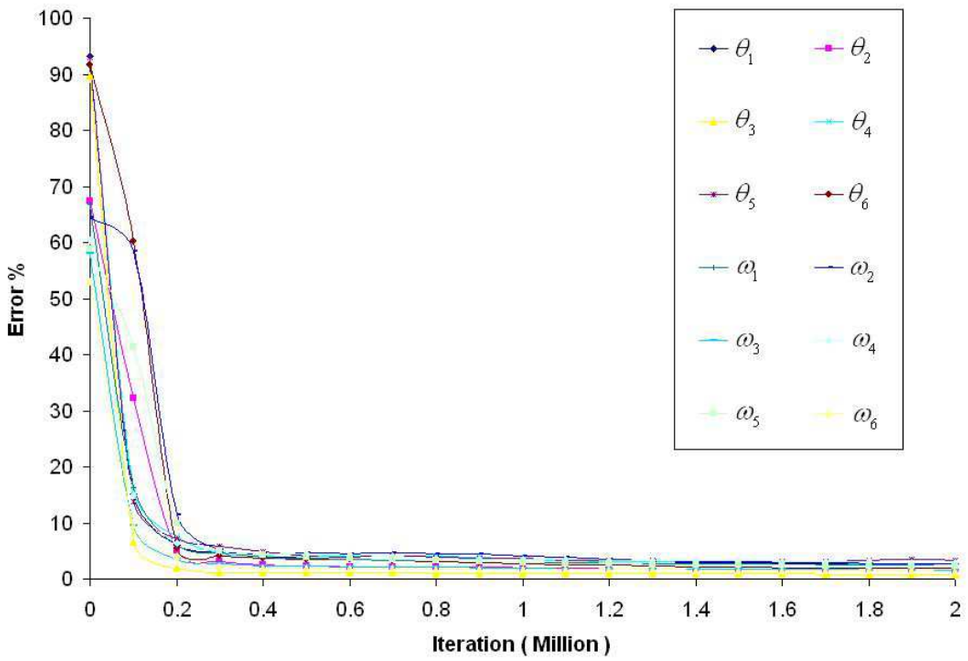


Fig. 10. The learning curve for the second configuration

5.1.3 Networks' performance

The performance of the two networks was measured as the difference between desired and actual system output.

To drive the robot to follow a desired trajectory, it will be necessary to divide the path into small portions, and to move the robot through all intermediate points. To accomplish this task, at each intermediate location, the robot's IK equations are solved, a set of joint variables is calculated, and the controller is directed to drive the robot to the next segment, when all segments are completed, the robot would be at the end point as desired.

Figures 11 to 13 show the experimental trajectory tracking for the robot over the X, Y and Z Coordinates of the global coordinates system for both of the networks compared to each other verses the desired trajectory.

As can be seen through these Figures, the performance of the first network has improved when considering the Jacobian Matrix in the second network, in terms of precision and iteration

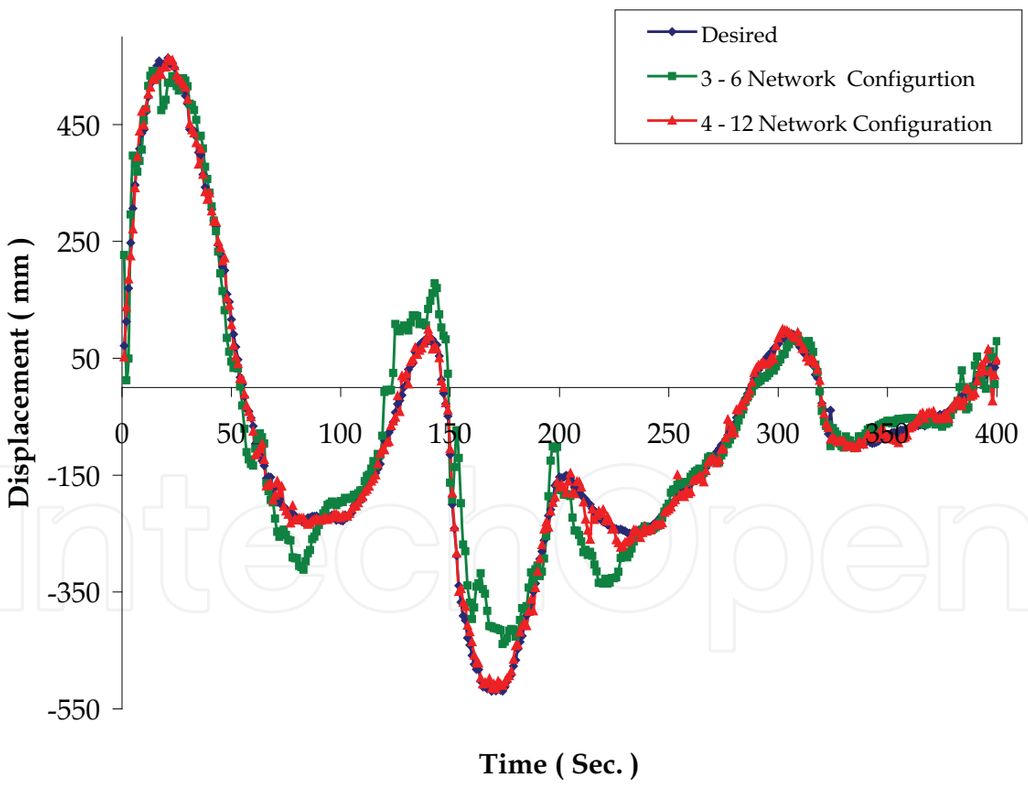


Fig. 11. Trajectory tracking for both configurations compared to each other after the training was finished for the X coordinate

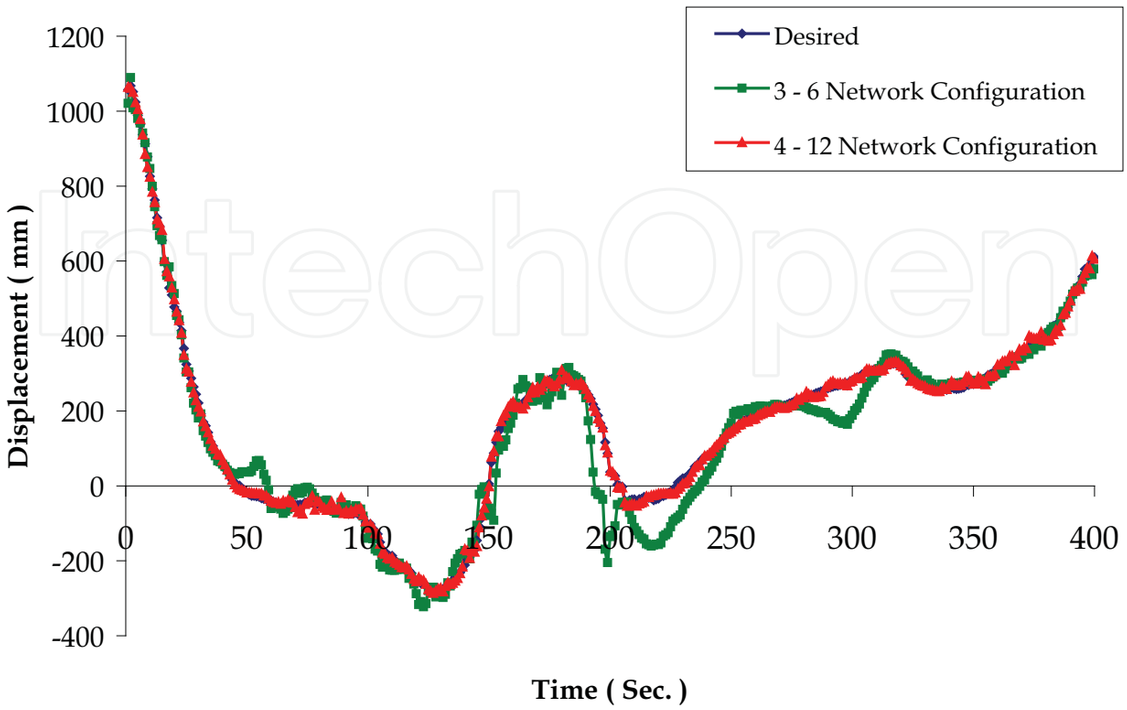


Fig. 12. Trajectory tracking for both configurations compared to each other after the training was finished for the Y coordinate

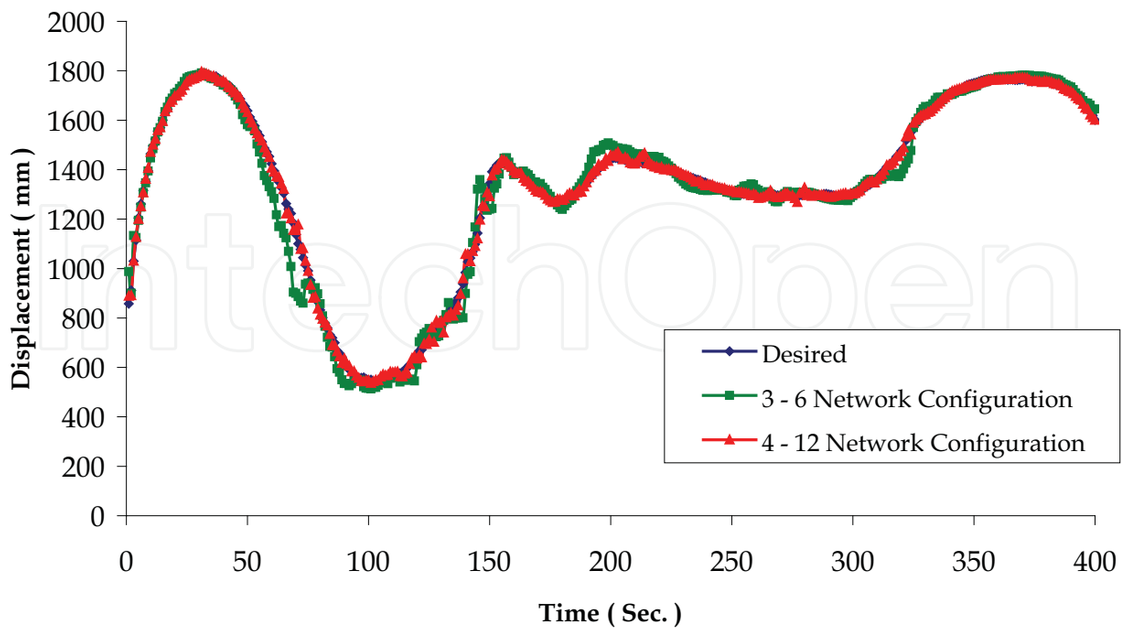


Fig. 12. Trajectory tracking for both configurations compared to each other after the training was finished for the Z coordinate

5.2 Testing phase

New data that has never been introduced to the network before have been fed to the second configuration network to test its ability to make prediction and generalization to any set of data later (as it has shown better response than the first configuration network).

Testing data were meant to pass through singular configurations (fourth and fifth joints); these configurations have been determined by setting the determinant of the Jacobian matrix to zero.

Table 3 shows the percentages of error for the testing data set for each joint during simulation stage.

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Angular Position	1.8%	0.245%	0.2%	5.32%	9.885%	0.08%
Angular Velocity	3.82%	2.875%	1.47%	2.64%	3.28%	1.41%

Table 3. Error percentages obtained for testing data through simulation stage

In order to verify the testing results during simulation stage, experiment has been performed to make sure that the output is the same or sufficiently close to the desired trajectory, and to show the combined effect of error, Figures 13, 14 and 15 show the predicted trajectory tracking of the X, Y, and Z coordinates respectively. Locus of which robot is passing through singular configurations are also shown.

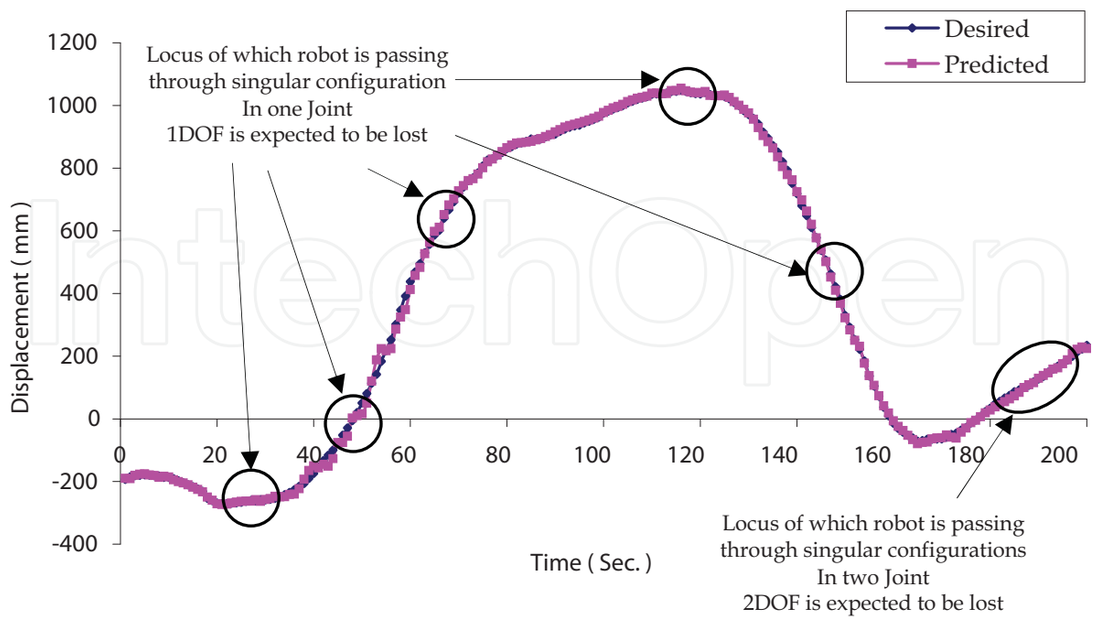


Fig. 13. Predicted trajectory for the X coordinate

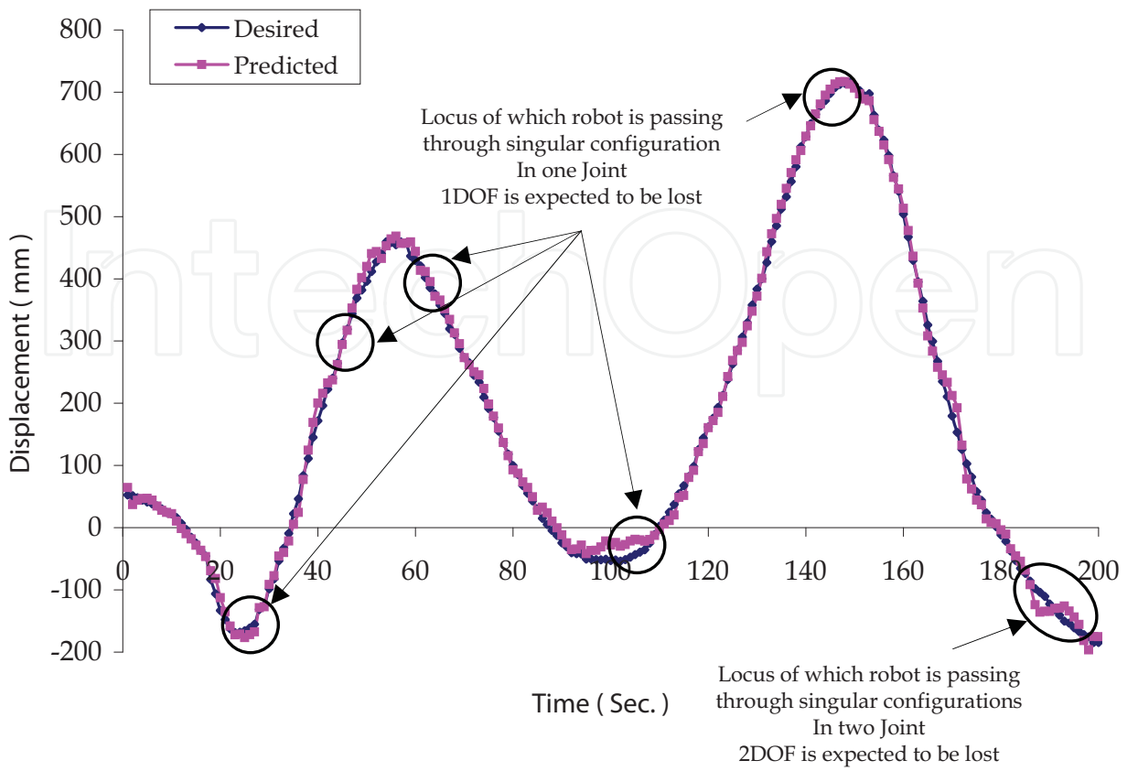


Fig. 14. Predicted trajectory for the Y coordinate

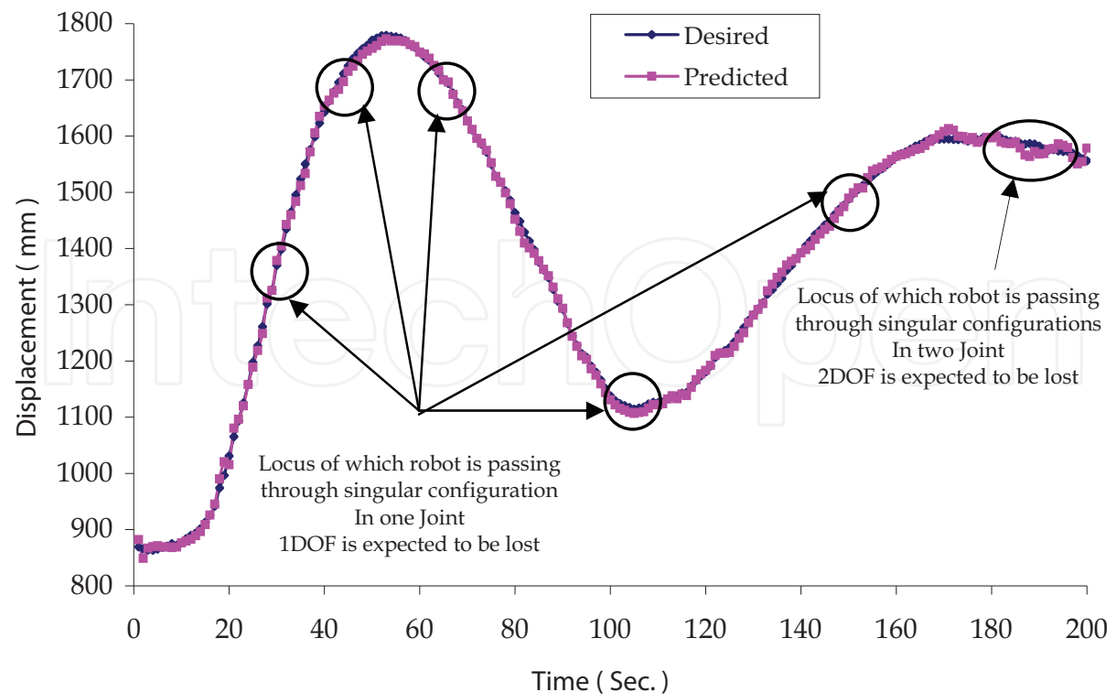


Fig. 15. Predicted trajectory for the Z coordinate

The error percentages in the experimental data are shown in table 4.

X	Y	Z
6.444%	16.355%	0.462%

Table 4. Error percentages obtained for testing data through experimental stage

6. Conclusions and recommendations

In order to overcome the drawbacks of some control schemes which depends on modeling the system being controlled, ANN technique has been utilized where learning is done iteratively based only on observation of input-output relationship unlike most other control schemes, which is a significant advantage of using ANN technology.

In the first network, although the number of hidden neurons was the same with the previous research despite the fact that the number of training patterns was doubled, an important remark is that the error percentage was higher than it was in the previous research which leads to a conclusion that this network configuration does not have the ability to learn huge number of patterns and its use will be limited to small number of data patterns.

As this research has shown, the consideration of the Jacobian Matrix in the solution of the Inverse Kinematics problem using neural networks gives a better response. As compared to the Fuzzy Learning Control algorithm results, the trained network was able to remember not only the training data but also was able to predict unknown trajectories as well as can be seen through the testing phase which is a significant advantage of using this approach.

Backpropagation algorithm has been used as a learning algorithm with sigmoid transfer function as an activation function in all neurons, we would like to recommend that a different learning algorithm, different activation function and/or different number of hidden layers to be used in order to achieve, if possible, a better response in terms of precision and iteration.

7. References

Al-Assadi, H.M.A.A.; Hamouda, A.M.S.; Ismail, N. & Aris, I. (2007). An adaptive learning algorithm for controlling a two-degree-of-freedom serial ball-and-socket actuator. *Proceedings of the IMechE Part I Journal of Systems & Control Engineering*, Vol.221, No. 7,pp.1001-1006.

Antonelli, G.; Chiaverini, S. & Fusco, G. (2003). A new on-line algorithm for inverse kinematics of robot manipulators ensuring path-tracking capability under joint limits. *IEEE Transaction on Robotics and Automation*, Vol.19, No.1, pp. 162-167.

Bingual, Z.; Ertunc, H.M. & Oysu, C. (2005). Comparison of Inverse Kinematics Solutions Using Neural Network for 6R Robot Manipulator with Offset. *ICSC congress on Computational Intelligence*.

Driscoll, J.A. (2000). Comparison of neural network architectures for the modeling of robot inverse kinematics. *Proceedings of the IEEE*, south astcon, pp. 44-51.

D'Souza, A.; Vijayakumar, S. & Schaal, S. (2001). Learning Inverse Kinematics. *Proceedings of the 2001 IEEE/ RSJ International Conference on Intelligent Robots and Systems*, pp.298-303, Maui, Haw- USA.

Fu, K.S.; Gonzalez, R.C. & Lee, C.S.G. (1987). *Robotics control, Sensing, Vision and intelligence*, McGraw-Hill Book Co, New York.

Funahashi, K.I., 1998. On the approximate realization of continuous mapping by neural networks. *Journal of Neural Networks*, Vol.2, No.3, pp.183-192.

Graca, R.A. & Gu, Y. (1993). A Fuzzy Learning Algorithm for Kinematic Control of a Robotic System. *Proceeding of the 32nd Conference on Decision and Control*, pp.1274-1279, San Antonio, Texas, USA.

Hasan, A.T.; Hamouda, A.M.S.; Ismail, N. & Al-Assadi, H.M.A.A. (2007). A new adaptive learning algorithm for robot manipulator control. *Proceeding of the IMechE, Part I: Journal of System and Control Engineering*, Vol.221, No.4, pp. 663-672.

Hasan, A.T.; Hamouda, A.M.S.; Ismail, N. & Al-Assadi, H.M.A.A.(2006). An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator. *Journal of Advances in Engineering Software*, Vol.37, pp. 432-438.

Haykin S. (1994). *Neural Networks*. A Comprehensive Foundation. New York: Macmillan.

Hornik, K. (1991). Approximation capabilities of multi-layer feed forward networks. *IEEE Trans. Neural Networks*, Vol.4, No.2, pp. 251-257.

Hu, Z.; FU, Z. & Fang, H. (2002). Study of singularity robust inverse of Jacobian matrix for manipulator, *Proceedings of the First International Conference on Machine Learning and Cybernetics*, pp. 406-410,China, Beijing.

Kalogirou, S.A. (2001) Artificial Neural Networks In Renewable Energy Systems Applications: a review. *Renewable and Sustainable Energy Reviews*. Vol. 5,pp.373-401.

Karilk, B. & Aydin, S. (2000). An improved approach to the solution of inverse kinematics problems for robot manipulators. *Journal of Engineering applications of artificial intelligence*, Vol.13, pp.159-164.

Köker, R. (2005). Reliability-based approach to the inverse kinematics solution of robots using Elman's networks. *Engineering Applications of Artificial Intelligence*, Vol.18, pp. 685-693.

Köker, R.; Öz, C.; Çakar.T. & Ekiz, H. (2004). A study of neural network based inverse kinematics solution for a three-joint robot. *Journal of Robotics and Autonomous Systems*, Vol.49, pp. 227–234.

Kuroe, Y.; Nakai, Y. & Mori, T. (1994). A new Neural Network Learning on Inverse Kinematics of Robot Manipulators, *International Conference on Neural Networks, IEEE world congress on computational Intelligence*. Vol.5, pp. 2819-2824.

Nakamura, Y. & Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control, *Journal of Dynamic Systems Measurements Control*, Vol. 108,pp. 163–171.

Ogawa, T.; Matsuura, H. & Kanada, H. (2005). A Solution of Inverse Kinematics of Robot Arm Using Network Inversion. *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation*.

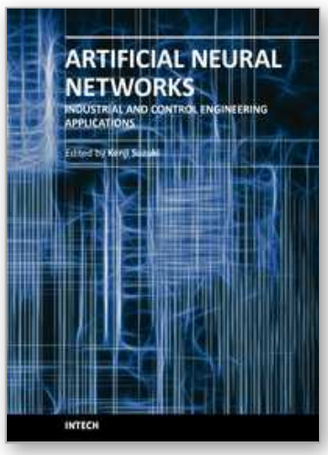
Santosh, A. ;Devendra P. Garg. (1993). Training back propagation and CMAC neural networks for control of a SCARA robot. *Journal of Engineering Applications of Artificial Intelligence*. Vol.6.No.2. pp.105-115.

Wampler, C. W. & Leifer, L. J. (1988). Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators. *Journal of Dynamic Systems Measurements Control*, Vol. 110,pp. 31–38.

Wampler, C. W. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods, *IEEE Transaction Syst., Man, Cybernetics*. Vol. 16,pp. 93–101.

Whitney. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transaction Man–Mach. Systems*, Vol. MMS–10,pp.47–53.

Zurda, M. J. (1992). *Introduction to Artificial Neural System Network*. West Publishing Companies, ISBN 0-314-93397-3, St. Paul, MN, USA.



Artificial Neural Networks - Industrial and Control Engineering Applications

Edited by Prof. Kenji Suzuki

ISBN 978-953-307-220-3
Hard cover, 478 pages
Publisher InTech
Published online 04, April, 2011
Published in print edition April, 2011

Artificial neural networks may probably be the single most successful technology in the last two decades which has been widely used in a large variety of applications. The purpose of this book is to provide recent advances of artificial neural networks in industrial and control engineering applications. The book begins with a review of applications of artificial neural networks in textile industries. Particular applications in textile industries follow. Parts continue with applications in materials science and industry such as material identification, and estimation of material property and state, food industry such as meat, electric and power industry such as batteries and power systems, mechanical engineering such as engines and machines, and control and robotic engineering such as system control and identification, fault diagnosis systems, and robot manipulation. Thus, this book will be a fundamental source of recent advances and applications of artificial neural networks in industrial and control engineering areas. The target audience includes professors and students in engineering schools, and researchers and engineers in industries.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ali T. Hasan, Hayder M.A.A. Al-Assadi and Ahmad Azlan Mat Isa (2011). Neural Networks' Based Inverse Kinematics Solution for Serial Robot Manipulators Passing Through Singularities, Artificial Neural Networks - Industrial and Control Engineering Applications, Prof. Kenji Suzuki (Ed.), ISBN: 978-953-307-220-3, InTech, Available from: http://www.intechopen.com/books/artificial-neural-networks-industrial-and-control-engineering-applications/neural-networks-based-inverse-kinematics-solution-for-serial-robot-manipulators-passing-through-sing

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen