ELSEVIER

Advances in Engineering Software 37 (2006) 432–438

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator

Ali T. Hasan, A.M.S. Hamouda *, N. Ismail, H.M.A.A. Al-Assadi

Department of Mechanical and Manufacturing Engineering, Faculty of Engineering, University Putra Malaysia, 43400UPM Serdang, Selangor, Malaysia

Received 19 November 2004; received in revised form 4 August 2005; accepted 30 September 2005
Available online 3 February 2006

Abstract

An adoptive learning strategy using an artificial neural network ANN has been proposed here to control the motion of a 6 D.O.F manipulator robot and to overcome the inverse kinematics problem, which are mainly singularities and uncertainties in arm configurations. In this approach a network have been trained to learn a desired set of joint angles positions from a given set of end effector positions, experimental results has shown an excellent mapping over the working area of the robot, to validate the ability of the designed network to make prediction and well generalization for any set of data, a new training using different data set has been performed using the same network, experimental results has shown a good generalization for the new data sets.

The proposed control technique does not require any prior knowledge of the kinematics model of the system being controlled, the basic idea of this concept is the use of the ANN to learn the characteristics of the robot system rather than to specify explicit robot system model. Any modification in the physical set-up of the robot such as the addition of a new tool would only require training for a new path without the need for any major system software modification, which is a significant advantage of using neural network technology.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Neural networks; Inverse kinematics; Back propagation; Robot control

1. Introduction

Trajectory control of robotic manipulators traditionally consists of following a pre-programmed sequence of end effector movements. Robot control usually requires control signals applied at the joints of the robot while the desired trajectory, or sequence of arm end positions, is specified for the end effector. The robot arms operate in a plane. To make the arm move, desired coordinates of the end effector point are fed to the robot controller for generating the joint angles for the motors that move the arms. To perform end effector position control of a robotic manipulator, the inverse kinematics problem (IK), needs to be solved [1].

Finding a solution to the IK problem analytically for serial manipulators is a difficult problem and a focus of much researches [2–12]. One source of difficulty is due to the fact that the inverse is not a true function; the set of solutions is infinite. In addition, the inverse map-ping is non-linear, and the nature of the inverse solution set for particular target locations undergoes a qualitative change as the manipulator moves between different regions of the workspace.

If these solutions provided were reasonably efficient in solving the problem, they have several drawbacks also, Firstly the complexity of the model requires high computational time and if the specific model-structure does not properly reflect all the robot characteristics, it can result in a poor control performance, Secondly the fact that the model is highly system-specific makes it very hard to accommodate physical changes such as the addition of a new tool [13].

The overall complexity of robot control problem and the quest for a truly autonomous robot system has led to considerable interest being devoted to the application of neural network technology to robot control [14–18]. ANNs are widely accepted as a technology offering an alternative way to tackle complex and ill-defined problems. They can learn from examples, are fault tolerant in the sense that they are able to handle noisy and incomplete data, are able to deal with non-linear problems and, once trained, can perform prediction and generalization at high speed, They are particularly useful in system modelling such as implementing complex mappings [19].

To overcome the uncertainties and non-linearity of the robot model, an ANN have been designed to learn the characteristics

* Corresponding author. Tel.: +603 89466330; fax: +603 89422107.
E-mail address: hamouda@eng.upm.edu.my (A.M.S. Hamouda).

0965-9978/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2005.09.010

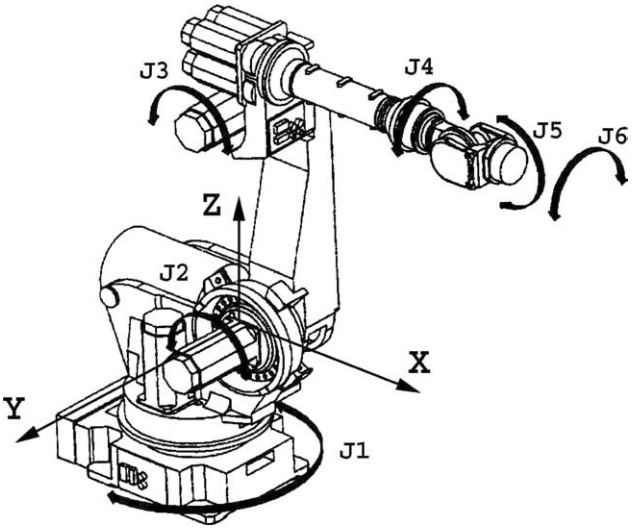

Fig. 1. FANUC M710i Robot, Main axes, wriest axes and global coordinate system.

of a FANUC M-710i robot over particular region of the working space, the FANUC M-710i as can be seen in Fig. 1, 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. J1, J2, and J3 are 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, painting, etc.

This paper is devoted to the develop of a new adaptive ANN controller to track IK control problem of a 6 D.O.F serial robot manipulator, the learning algorithm is based on the adaptive updating of the weights of the network by minimizing the tracking error after each iteration process.

2. Robot arm kinematics configuration

The inverse kinematics problem is the problem of finding a vector of joint variables that produce a desired end effector location. If a unique vector of joint angles exists which attains the desired end effector location, there is a well-defined inverse to the forward kinematics function and the inverse kinematics problem is well posed. Unfortunately, the inverse kinematics problem can be ill-posed because the solution for the forward kinematics is not unique; in many cases solving the inverse kinematics problem may result in infinite number of solutions [20].

The kinematics considerations for the manipulator shown in Fig. 2 are based on the forward kinematics equation. The forward kinematics equation involves mapping of joint angle coordinates (θ_1, θ_2) to the end effector position (x, y). The mapping expressions can be obtained by inspection of

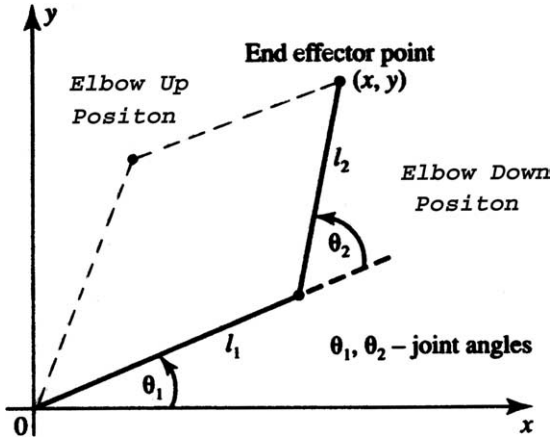

Fig. 2. The two arm configurations positioning at (x, y).

the figure as follows:

$$x = l_1\cos\theta_1 + l_2\cos(\theta_1 + \theta_2) \tag{1}$$

$$y = l_2\sin\theta_1 + l_2\sin(\theta_1 + \theta_2)$$

Where θ_1 and θ_2 are the joint angles of the first and second arm segments, respectively, l_1, and l_2 are respective arm segment lengths. Relation (1) expresses the forward kinematics problem and implements unique mapping from the joint angle space to the Cartesian space.

The inverse kinematics problem is described as follows:

$$\theta_2 = \cos^{-1}\left[\frac{\left(x^2 + y^2 - l_1^2 - l_2^2\right)}{(2l_1l_2)}\right] \tag{2a}$$

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left[\frac{l_2\sin\theta_2}{(l_1 + l_2\cos\theta_2)}\right] \tag{2b}$$

Since ($\cos^{-1}$) is not a single-valued function in the range of angles of interest, two possible orientations typically result from relation (2) for the robot arm joint angles. The arm can be positioned with the elbow up or down, with the end effector still at the required (x, y) point. The inverse kinematics transformation (2) implementing mapping from Cartesian space to joint space is thus not unique [21].

3. The neural network strategy

The possibility of developing a machine that would 'think' has intrigued human beings since ancient times, machinery can outperform humans physically. Similarly, computers can outperform mental functions in limited areas, notably in the speed of mathematical calculations. For example, the fastest computers developed are able to perform roughly 10 billion calculations per second. But making more powerful computers will probably not be the way to create a machine capable of thinking. Computer programs operate according to set procedures, or logic steps, called algorithms. In addition, most computers do serial processing such as operations of recognition and computations are performed one at a time. The brain works in a manner called parallel processing,

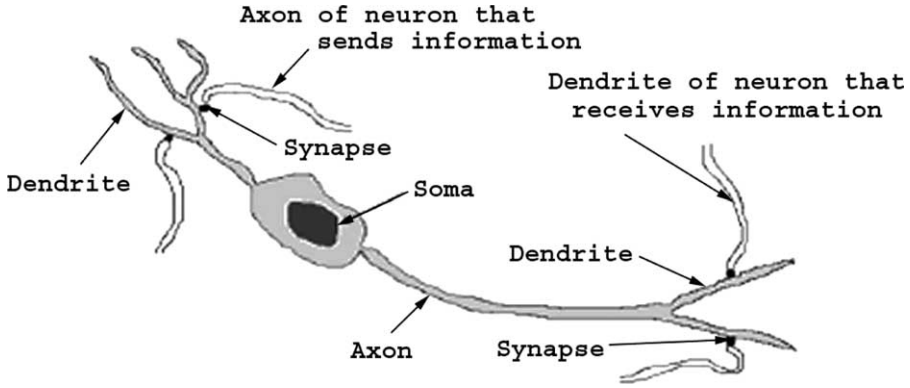

Fig. 3. Schematic diagram for the biological neuron.

performing a number of operations simultaneously to achieve simulated parallel processing.

Artificial neural networks ANNs are collections of small individual interconnected processing units. Information is passed between these units along interconnections; the 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.

The elementary nerve cell called a neuron, which is the fundamental building block of the biological neural network. Its schematic diagram is shown in Fig. 3. 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 [1].

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, Fig. 4 illustrates how information is processed through a single 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 [19]; 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, 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, there are typically tow 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. Following some suitable operation, it results in a desired output [13]. Knowledge is usually stored as a set of connecting weights (presumably corresponding to synapse efficiency in biological neural system).

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 [22].

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 [19].

Two different types of learning can be distinguished: supervised and unsupervised learning, In supervised learning we assume that at each instant of time when the input is applied, the desired response d of the system is provided by

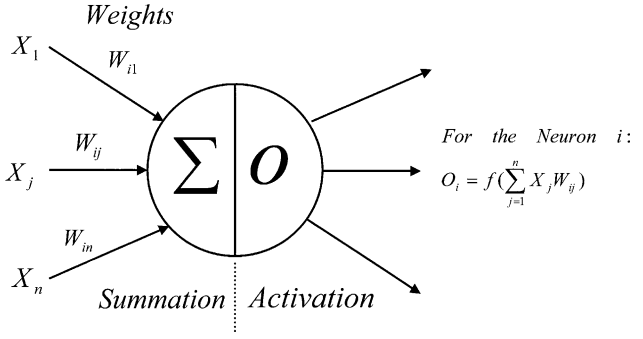

Fig. 4. Information processing in the neural unit.

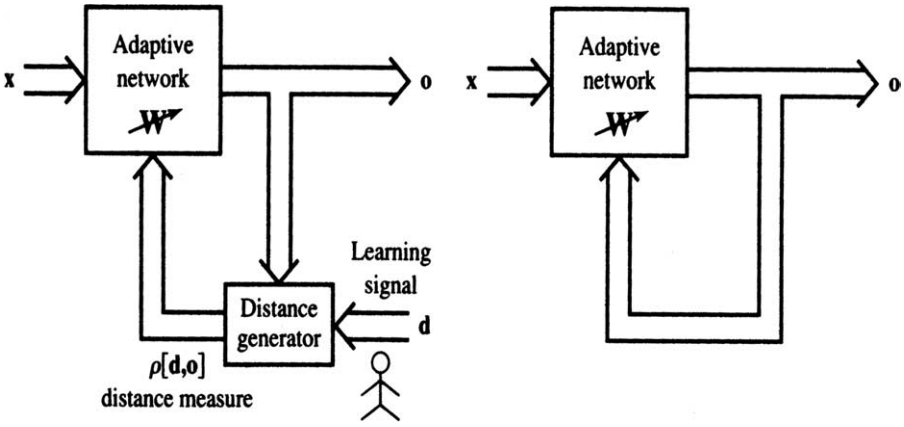

Fig. 5. Block diagram for explanation of basic learning modes.

the teacher. This is illustrated in Fig. 5(a). The distance $\rho[d, O]$ between the actual and the desired response serves as an error measure and is used to correct network parameters externally. Since we assume adjustable weights, the teacher may implement a reward-and-punishment scheme to adapt 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.

Fig. 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 [1].

Fig. 6 illustrates the topology of the designed network which consists of three layers one of them is a hidden layer, presented to store the internal representation, the fundamental idea underlying the design of the 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 [13].

As it can be seen in Fig. 6 the output of the units in layer A 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:

$$\mathrm{Sum_B} = \sum_i O_i W_{ij} \tag{3}$$

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

$$O_j = f(\mathrm{sum_B}) \tag{4}$$

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

$$f(O_j) = \frac{1}{1 + \mathrm{e}^{-O_j}} \tag{5}$$

as the non-linear 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

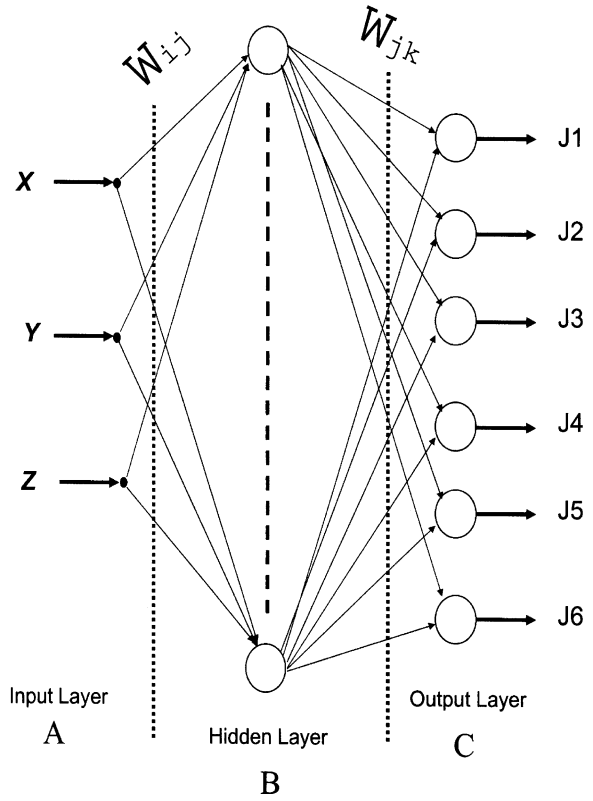

Fig. 6. The topology of the designed network.

calculated by:

$$e_K = d_K - O_K \tag{6}$$

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:

$$E = \frac{1}{2}\sum_K (d_K - O_K)^2 \tag{7}$$

Learning comprises changing weights so as to minimize the error function. 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. Eqs. (3) and (4) 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 Eq. (5), the so-called delta-rule for iterative convergence towards a solution may be stated in general as:

$$\Delta W_{JK} = \eta \delta_K O_J \tag{8}$$

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

$$\delta_K = O_K(1 - O_K)(d_K - O_K) \tag{9}$$

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

$$\delta_J = O_J(1 - O_J)\sum_K \delta_K W_{JK} \tag{10}$$

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 tow step procedure as it is shown in Fig. 7, 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:

$$W_{IJ} = W_{IJ} + \eta \delta_J O_I \tag{11}$$

$$W_{JK} = W_{JK} + \eta \delta_K O_J \tag{12}$$

Until for each input vector the output vector produced by the network is the same as (or sufficiently close to) the desired output vector [13].

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 [19].

4. Simulation results

In order to overcome the uncertainties and non-linearity which results from the problem of the IK and to control the robot by making sure that for a certain path the position of each joint angle will be the same as desired when planning the trajectory for the robot, a supervised feed forward ANN has been designed using C++ programming language.

The network consists of input, output and one hidden layer as can be seen in Fig. 6, every neuron in the network 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 with a constant learning rate of 0.99. 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 six joints.

In backpropagation networks, number of hidden neurons determines how well a problem can be learned. If too many are used, the network will tend to try to memories the problem and thus not generalize well later, if too few are used the network will generalize well but may not have enough power to learn the patterns well. Getting the right number of hidden neurons is a matter of trial and error, since there is no science to it [19]; number of hidden neurons was set to be 43 experimentally.

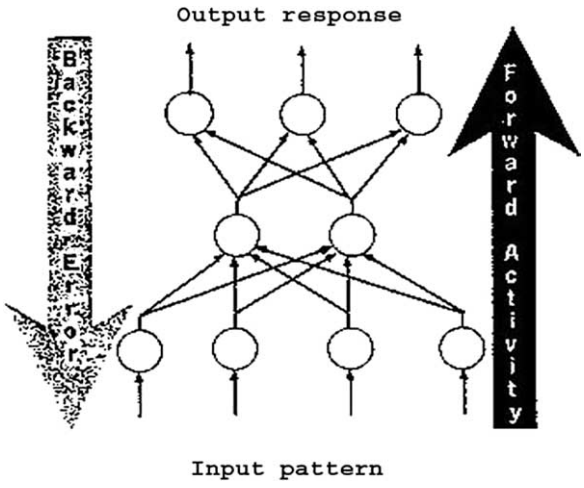

Fig. 7. Information flow through a backpropagation network.

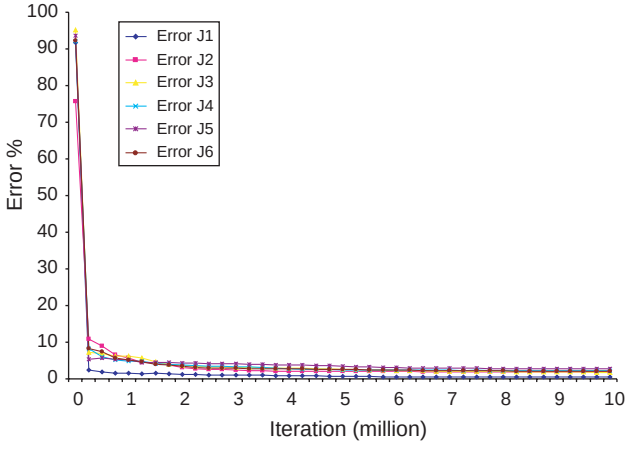

Fig. 8. The learning curve for the desired path.

Table 1
The absolute error percentages for the desired path after 8×10^6 iterations

Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
0.47%	1.96%	1.985%	2.3%	2.8%	2.185%

Table 2
The absolute error percentages for the new path after 8×10^6 iterations

Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
3.635%	3.66%	5.31%	1.73%	3.435%	6.1%

All input and output values are usually scaled individually such that overall variance in the 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, training sets used were taken by driving the robot manually to follow a desired path.

A learning system my adapt its internal structure to achieve a better response, a performance measure could be the difference or error between desired and actual system output. In the GDR the system is modified following each iteration which leads to the learning curve shown in Fig. 8, this error is reduced in subsequent trials, as this curve detects; convergence has been achieved after 8×10^6 iterations, and error is still approximately constant afterwards.

To drive the robot to follow a desired path, it will be necessary to divide this path into small portions, and to move the robot through all intermediate points. To accomplish this task, at each intermediate location, the robot's inverse kinematics equations are solved, a set of joint variable is calculated, and the controller is directed to drive the robot to the next segment. When all segments are completed, the robot will be at the end point as desired, 200 points represent a desired path for the robot were used in the simulation.

A very low percentage of absolute error has been found experimentally, Table 1 shows the absolute error percentage for each of the six joints. Through this table an excellent path tracking can be seen.

To improve the ability of the designed network to make prediction and well generalization later, a new path was fed to the network, experimental results has shown the ability of the designed network to make good prediction for a new path which have not been trained for previously, after 8×10^6 iterations, absolute error is shown in Table 2, the learning curve shown in Fig. 9, shows the building knowledge procedure for the new path which gives an indication for the success of the proposed algorithm.

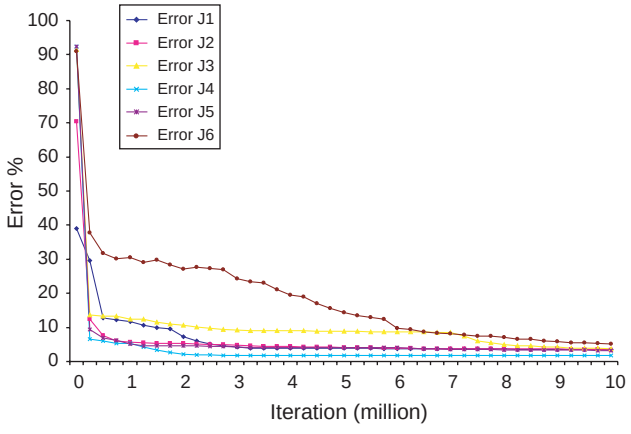

Fig. 9. The learning curve for the new path.

5. Conclusion

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

The aim of using neural network is to find a set of weights that ensure that for each input vector the output vector produced by the network is the same as or sufficiently close to the desired output vector, so, system model does not have to be known at the time of the controller design, also, any change in the physical setup of the system such as the addition of a new tool would only involve training and will not require any major system software modifications.

An ANN has been devoted to address the problem of adaptive control in unknown environments; the ANN controller was trained to overcome the IK problem in serial manipulators over particular region of the working space, a good path tracking could be seen even when the network was provided with new data which was not trained for previously which improves that the use of the ANN technology is a reasonable method to overcome the IK problem in serial manipulators of general geometry.

This control scheme would work well in a typical industrial set-up where the controller of a robot could be taught the handful of trajectories depending on the task assigned to that robot. Since the neural controller is system independent, the same controller could be used for different robots, and each one could be trained according to the specifics of the task the robot would be required to perform.

References

[1] Jacek M Zurada. Introduction to artificial neural systems. Singapore: West Publishing Company; 1992.
[2] Denavit J, Hertenberg RS. Kinematics notation for lower pair mechanism based on matrices. Appl mech 1955;77:215–21.
[3] Chase MA. Vector analysis of linkages. Eng Ind 1963;85:289–97.
[4] Yang AT. Displacement analysis of spatial five-link mechanism using (3×3) matrices with dual-number element. Eng Ind 1969;9(1):152–7.
[5] Albala H, Angeles J. Neumerical solution to the input–output displacement equation of the general 7R spatial mechanism. Fifth world congress on theory of machines and mechanisms 1979 p.1008–11.
[6] Duffy J, Rooney J. A foundation for a unified theory of analysis of spatial mechanism. Eng Ind 1975;97(4):1159–64.
[7] Duffy J. Analysis of mechanism and robot manipulators. New York: Wiley; 1980.
[8] Pennock GR, Yang AT. Application of Dual-Number matrices to the inverse kinematics problem of robot manipulators. Mech Transm Autom Des 1985;107(2):201–8.

[9] Tsai LW, Morgan A. Solving the Kinematics of the most general six and five degree of freedom manipulators by continuation methods. Mech Transm Autom Des 1985;107:189–200.
[10] Lee HY, Liang CG. A new vector theory for the analysis of spatial mechanism. Mech Mach Theory 1988;23(3):209–17.
[11] Lee HY, Liang CG. Displacement analysis of the general spatial 7-link 7R mechanism. Mech Mach Theory 1988;23(3):219–26.
[12] Daniel M, Raul G. Hierarchical Kinematics analysis of robots. Mech Mach Theory 2003;33:497–518.
[13] Santosh A, Devendra P Garg. Training back propagation and CMAC neural networks for control of a SCARA robot. Eng Appl Artif Intell 1993;6(2):105–15.
[14] Ananthraman S, Nagchaudhuri A, Garg DP. Control of a robotic manipulator using a neural controller. Proceedings of 22nd Modeling and Simulation Conference. Pittsburgh, PA; 1991. p. 1846–1853.
[15] Atkeson CG, Reinkensmeyer DJ. Using content address-able memories to control robots. Proceedings of 27th conference on decision and control. Austin, TX; 1988. p. 792–797.
[16] Cruse H, Bruwer M. A simple network controlling the movement of a three joint planar manipulator. Parallel Process Neural Syst Comput 1990; 409–12.
[17] Kuperstein M, Wang J. Neural controller for adaptive movements with unforeseen payloads. IEEE Trans Neural Netw 1990;1: 137–42.
[18] Miller III WT, Hewes RP, Glanz FH, Kraft III LG. Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. IEEE Trans Robot Autom 1990;6:1–9.
[19] Soteris AKalogirou. Artificial neural networks in renewable energy systems applications: a review. Renew Sust Energy Rev 2001;5: 373–401.
[20] Omid O, Patrick VDS. Neural systems for robotics. London: Academic Press Limited; 1997.
[21] Craig JJ. Inttrduction to robotics Mechanics and control. Reading, MA: Adison-Wesley Publishing Co; 1986.
[22] Haykin S. Neural networks A comprehensive foundation. New York: Macmillan; 1994.