International Journal of Emerging Research in Management &Technology ISSN: 2278-9359 (Volume-7, Issue-1)

Research Article | **January 2018**

Back Propagation Method of Artificial Neural Networks for Finding the Position Control of Stanford Manipulator and Direct Kinematic Analysis of Elbow Manipulator

M. Sailaja, R. D. V. Prasad

Assistant Professor, Department of Mechanical Engineering, ANITS, Visakhapatnam, Andhra Pradesh, India

Email- sailaja.me@anits.edu.in, rdvprasad.me@anits.edu.in

Abstract—

Nowadays the robot technology is advancing rapidly and the use of robots in industries has been increasing. In designing a robot manipulator, kinematicsplays a vital role. The kinematic problem of manipulator control is divided into two types, direct kinematics and inverse kinematics. Robot inverse kinematics, which is important in robot path planning, is a fundamental problem in robotic control. Past solutions for this problem have been through the use of various algebraic or algorithmic procedures, which may be less accurate and time consuming. Artificial neural networks have the ability to approximate highly non-linear functions applied in robot control. The neural network approach deserves examination because of the fundamental properties of computation speed, and they can generalize untrained solutions. In the present work an attempt has been made to evaluate the problemof robot inverse kinematics of Stanford manipulator using artificial neural network approach. Finally two programs are written using C language to solve inverse kinematic problem of Stanford manipulator using Back propagation method of artificial neural network. In this network, the input layer has six nodes, the hidden layer has three nodes, and the output layer has two nodes. And also Elbow manipulator was modelled and its direct kinematics was analysed.

Keywords —Stanford manipulator, Direct kinematics, Inverse kinematics, Artificial neural networks, Back propagation method, Elbow manipulator

I. INTRODUCTION

A. Robot Definition

The definition used by Robotics Institute of America is "A Robot is a reprogrammable multifunctional manipulator designed to move materials, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks." Webster's New World College Dictionary defines a robot as "Any anthropomorphic mechanical being built to do routine manual work for human being."

B. Stanford Manipulator

The Stanford manipulator is a six-dof industrial manipulator and is characterized by a three degree of freedom arm and three degree of freedom wrist. The first three joints, two revolute and one prismatic, constitute the arm of the spherical (RRP) configuration. The last three revolute joints constitute a wrist of RRR configuration. The first three links are larger in size and are used to position the wrist and the last three links for wrist are small in size and are used to orient the end effector as shown in Figure 1.

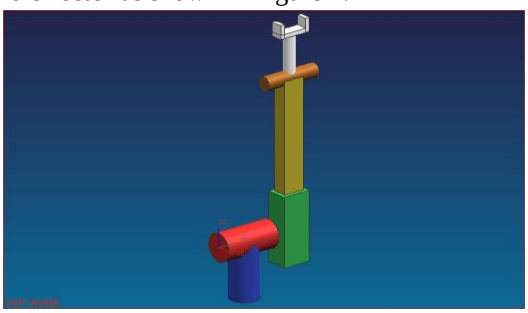


Fig. 1 Stanford Manipulator

C. Definition of Artificial Neural Network

An artificial neural network, also called a simulated neural network (SNN) or just a neural network (NN), is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. ANN involves a network of relatively simple processing elements, where the global behavior is determined by the connections between the processing elements and element parameters. So ANN is considered as a highly simplified model of the structure of the biological neural network. In a neural network model, simple nodes (called variously "neurons", "neurodes", "PEs" ("processing elements") or "units") are connected together to form a network of nodes — hence the term "neural network". Artificial neural networks are organized in layers of interconnected nodes in a predetermined manner to accomplish a desired pattern recognition task. ANN presents a new information processing paradigm for decision support that integrates knowledge and learning. ANN has the capability of performing massive parallel processing. ANNs can perform the task of associative memory. The arrangement of processing units, connections and pattern input/output is referred to as "topology". The simplified view of artificial neural network is shown in Figure2

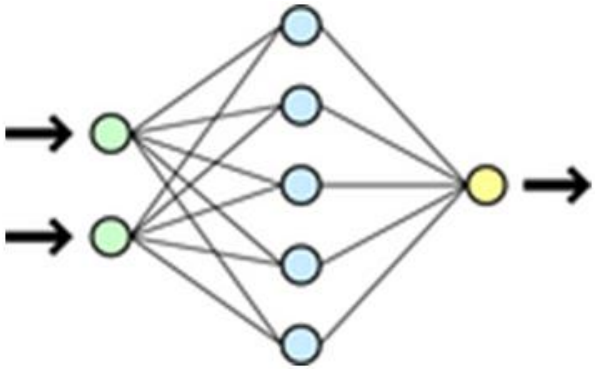


Fig 2 Simplified view of an artificial neural network

II. NEURAL NETWORK APPROACH IN ROBOTICS

A. Kinematic robot learning by neural networks

It is well known in robotics that control is applied at the level of the robot's joints, while the desired trajectory is specified through the movement of the end effector. Hence a control algorithm requires the solution of the inverse kinematic problem for a complex nonlinear system (connection between internal and external coordinates) in real time. However, in general, the path in Cartesian space is often very complex and location of the arm end effector cannot be efficiently determined before the movement is actually made. Also the solution of the inverse kinematic problem is not unique because of the fact that in the case of redundant robots there may be an infinite number of solutions.

In this case, conventional methods of solution consist of closed form methods and iterative methods. These are either limited only to a class of simple non redundant robots or are time-consuming and the solution may diverge due to a bad initial guess. This method is referred to as position based inverse kinematic control. In this area of position-based inverse kinematic control, various methods have been proposed to solve this problem. The basic idea common to all these algorithms is the use of the same topology of neural network (multilayer perceptron) and the same learning rule: back propagation algorithm.

B. Training a Neural Network for Inverse Kinematics

Neural Network solutions have the benefit of having faster processing times since information is processed in parallel. Neural systems can generalize to approximate solutions from small training sets. The network will not fail if a few neurons are damaged, and the solutions may still retain accuracy. When implementing a neural network approach, complex computers are not essential and robot controllers need not be specific to any one manipulator.

The nature of neural networks require that a set of training points be chosen which represent the nature of the inputs (x, y, z, dx, dy, dz) and the corresponding outputs ($\theta 1$, $\theta 2$ and $\theta 3$). If the training points chosen tend to be clustered, then the network will be very accurate when dealing with points near the cluster. In this case, the robot should be familiar with points throughout the entire workspace; thus, points should be evenly distributed.

The order in which points are presented to the network also affects the speed and quality of convergence. If the points are not presented in a random order, each training update will tend to train the network for the current section of space which the points are from.

The neural network architecture chosen for this problem is a simple network with one hidden layer. The network has six input neurons for the desired position, and three output neurons for the estimated joint angles. The nodes in the hidden layer are taken as greater than the square root of the sum of the nodes in the input layer and output layer. Instead of gathering experimental information from a robot, the Inverse Kinematic equations were used. The use of simulated data provided a reliable training and comparison for the neural network.

In order for the neural network to generalize to a one-to-one mapping, the training set consisted of a single solution set. It is assumed that the length of the robot links is equal to 0.5 and workspace is a semicircle with one unit radius. The decision to train over a semi of the work sphere was based upon the premise that by limiting the scatter of the training points, the most accurate inverse kinematics mapping would be acquired. If the training set had covered the entire workspace, then the solution may have been poorer with longer convergence times, because the network would have been forced to generalize over a volume, which is two times larger. The process of reaching each intermediate location along a path by the back propagation method, its schematic diagram, the flow chart and the procedure of back propagation method is shown below

III. BACK PROPAGATION METHOD

The back propagation algorithm is a popular and useful feed-forward supervised learning technique for tuning network parameters to fit a training set of input-output pairs.

1. Feed-forward neural network: The neural network which has no loops and the signals propagate only in one direction from an input stage through intermediate neurons to the output stage is feed-forward neural network.
2. Supervised learning: The learning technique which requires input and output data during the training phase is supervised learning.

A. Introduction

A back propagation network consists of at least three layers of units: an input layer, at least one intermediate hidden layer, and an output layer. In contrast to Hopfield networks, connection weights in a back propagation network are one-way. Typically, units are connected in a feed-forward fashion with input units fully connected to units in the hidden layer and hidden units fully connected to units in the output layer. When a back prop network is cycled, an input pattern is propagated forward to the output through the inventing input-to-hidden and hidden-to-output weights.With back prop network, learning occurs during a training phase in which each input pattern in a training set is applied to the input units and then propagated forward. The pattern of activation arriving at the output layer is then compared with the correct (associated) output pattern to calculate an error signal. The error signal for each such target output pattern is then back propagated from the outputs to the inputs in order to appropriately adjust the weights in each layer of the network. After a back prop network has learned the correct classification for a set of inputs, it can be tested on a second set of inputs to see how well it classifies untrained patterns. Thus, an important consideration in applying back prop learning is how well the network generalizes. This method is easy to understand, and can be easily implemented as a software simulation.

B. Description of the Back Propagation Algorithm

Finding the right set of weights to accomplish a given task is the central goal in connectionist research .Since the real uniqueness or 'intelligence' of the network exists in the values of the weights between neurons, we need a method of adjusting the weights to solve a particular problem. For this type of network, the most common learning algorithm is called Back Propagation (BP). A BP network learns by example, that is, we must provide a learning set that consists of some input examples and the known-correct output for each case. So, we use these input-output examples to show the network what type of behaviour is expected, and the BP algorithm allows the network to adapt. The BP learning process works in small iterative steps: one of the example cases is applied to the network, and the network produces some output based on the current state of its synaptic weights (initially, the output will be random). This output is compared to the known-good output, and a mean-squared error signal

is calculated. The error value is then propagated backwards through the network, and small changes are made to the weights in each layer. The weight changes are calculated to reduce the error signal for the case in question. The whole process is repeated for each of the example cases, then back to the first case again, and so on. The cycle is repeated until the overall error value drops below some pre-determined threshold.

C. When to Update Weights

Updating the weights in a Back Propagation network can be done either after the presentation of each pattern (pattern learning), or after all of the patterns in the training set have been presented (epoch learning). Weights updates in Brainwave are by pattern. If the learning rate is small, there are little differences between the two procedures. However, substantial differences can be observed when the learning rate is large, as the derivation of the back propagation algorithm assumes that the error derivatives are summed over all of the patterns.

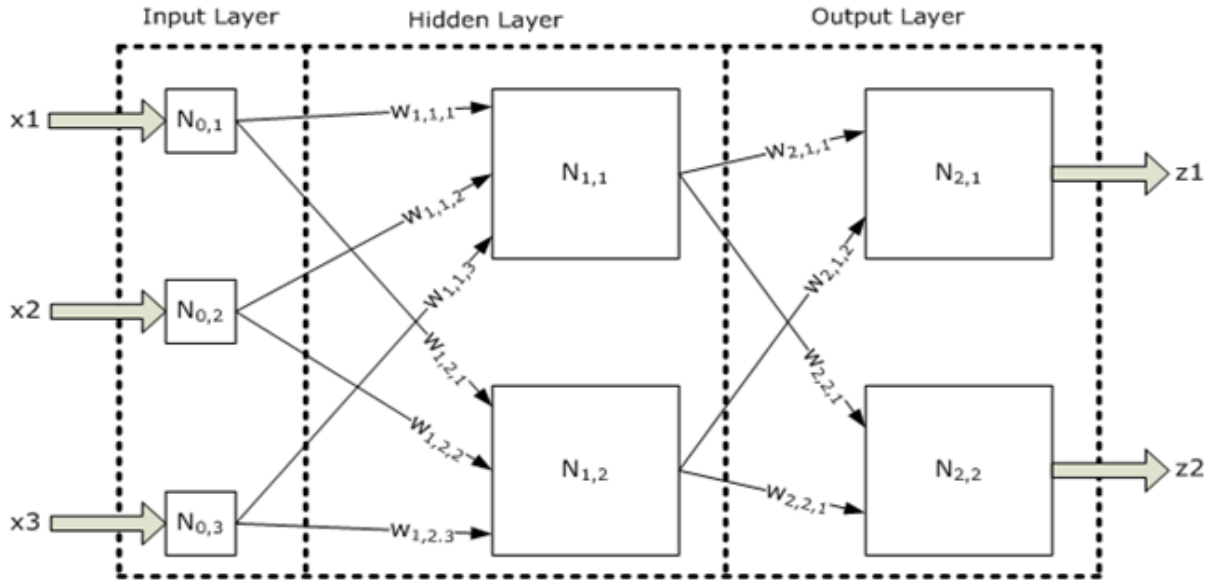


Fig3: Back propagation neural network

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, and could change per network configuration and type of data. In general the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance. You could start a network configuration using a single hidden layer, and add more hidden layers if you notice that the network is not learning as well as you like.

For example, suppose we have a bank credit application with ten questions, which based on their answers, will determine the credit amount and the interest rate. To use a Back propagation NN, the network will have ten neurons in the input layer and two neurons in the output layer.

The Back propagation NN works in two modes, a supervised training mode and a production mode. The training can be summarized as follows:

Start by initializing the input weights for all neurons to some random numbers between 0 and 1, then:

1. Apply input to the network.
2. Calculate the output.
3. Compare the resulting output with the desired output for the given input. This is called the error.
4. Modify the weights and threshold for all neurons using the error.
5. Repeat the process until error reaches an acceptable value (e.g. error< 1%), which means that the NN was trained successfully, or if we reach a maximum count of iterations, which means that the NN training was not successful.

The challenge is to find a good algorithm for updating the weights and thresholds in each iteration (step 4) to minimize the error.

Artificial neural network of the back-propagation type are being used increasingly for modeling environment systems. The connection weights are taken initially as random numbers less than unity. Use of both types of sigmoid functions is allowed, depending on the range of input and target vectors. A generalized computer program for training and subsequent testing operations is to be developed which requires a training file train.dat, where a set of input-target patterns is to be presented in a row wise manner. Upon training, the network stores connection weights;this will be retrieved when the new inputs are presented for obtaining outputs in forward direction.

Robot inverse kinematics is a fundamental problem in robotic control. In robotic control applications, neural networks can provide tools for system identification, including forward and inverse kinematic identification.Past solutions for this problem have been through the use of various algebraic or algorithmic procedures, which may be time consuming. Factor that affect the accuracy of most neural networks are the presence of mapping errors and local minima. Applying a network that calculates incremental rather than absolute joint parameters improves the performance by an order of magnitude. Networks in robotics application are trained with incremental data. In this thesis work, a neural network approach called back propagation method to the problem of robot inverse kinematics is evaluated.

D. Description Of The Back Propagation Method

The back propagation network consists of one input layer, one hidden layer, and one output layer. The input layer consists of six nodes, and the hidden layer consists of three nodes and the output layer consists of two nodes. It is assumed that the link lengths d_2 and d_3 are taken as 0.5 and 0.5 and the workspace is a semicircle with one unit radius. The network architecture of Stanford manipulator is shown in the Figure4

The forward kinematic equations of Stanford manipulator relating to the translation are given as

$$x = d_3 s_2 c_1 - s_1 d_2 , y = d_3 s_2 s_1 + c_1 d_2 , z = d_3 c_2$$

In these equations there are two angles θ_1 and θ_2 and the inverse kinematic equations are given as

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left(\frac{d_2}{+/-\sqrt{r^2 - d_2^2}}\right)$$

$$r = \sqrt{x^2 + y^2} \quad , \quad \theta_2 = \tan^{-1}\left(\frac{c_1 x + s_1 y}{z}\right)$$

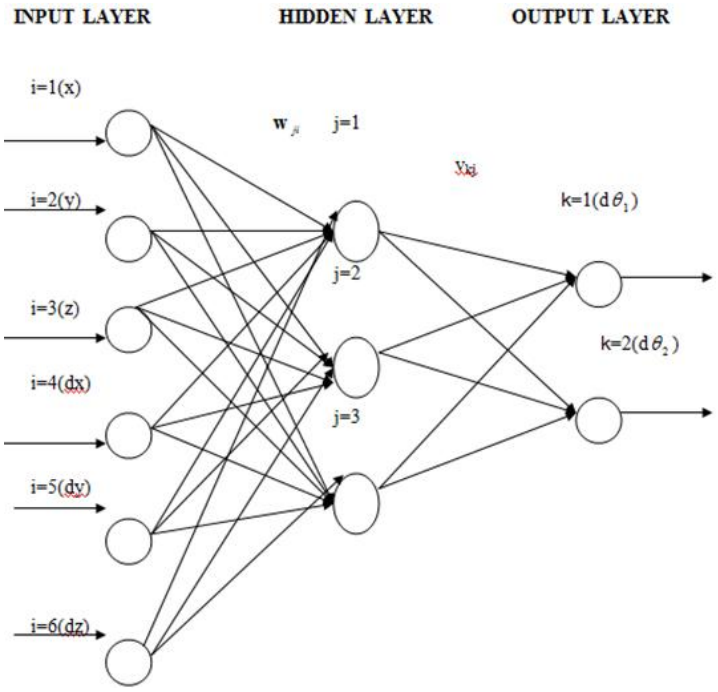


Fig 4: Network architecture of Stanford manipulator

First substituting in the forward kinematic equations x, y, z make a set of training patterns by taking different combinations of θ_1 and θ_2. Each pattern consists of (x, y, z, dx, dy, dz, dθ_1, dθ_2). The pattern (0 10 0.0868 0.5 0.4924) is taken the base where θ_1=0 and θ_2 =10 x = 0.0868 y =0.5 z =0.4924.From the base the incremental positions (dx, dy, dz) are calculated for the patterns shown below.The patterns require a training file train.dat. Patterns are stored as x [ι](x, y, z, dx, dy, dz), which is called, input vector and as t [k] (dθ_1, dθ_2), which is called output vector. The patterns are stored in this file.Each input pattern in a training set is applied to the input units and then propagated forward.

Input patterns (x, y, z, dx, dy, dz, dθ_1, dθ_2) for the angles θ_1=45, 90,135,180, θ_2 =30, 80,140,180taken are

-0.1768 0.5303 0.4330 -0.2636 0.0303 -0.0594 0.25 0.1176
-0.5 0.4924 0.0868 -0.5868 -0.0076 -0.4056 0.5 0.4117
-0.5808 -0.1263 -0.3830 -0.6676 -0.6263 -0.8754 0.75 0.7647
0 -0.5 -0.5 -0.0868 -1 -0.9924 1 1

The connection weights w_{ji} and v_{kj} are taken initially as random numbers less than unity. w_{ji} is the weight connecting neuron ι in the input layer to j in the hidden layer. v_{kj} is the weight connecting neuron j in the hidden layer to k in the output layer. the internal potential u_j of the j^{th} unit of the hidden layer is calculated by taking the summation of the linear combination of input vector x [ι] and the corresponding j^{th} row vector w_{ji} of the weight matrix w. the output h_j is obtained by non-linear transfer function from the input potential u_j. Similarly the internal potential s_k of the k^{th} unit of the output layer is calculated by taking the summation of the linear combination of the input vector h_j and v_{kj}. The output o_k is calculated by the non-linear transfer function of the input potential s_k.

The error is calculated by taking the difference between desired and actual output (t_k - o_k). This error signal for each such target output pattern is then back propagated from the outputs to the inputs in order to appropriately adjust the weights in each layer of the network. The connection weights w_{ji} and v_{kj} are adjusted to minimize the difference between the de-sired and actual output as shown in the flow chart fig. Like this the iterations are continued so as to minimize the cycle average error.

In this project four patterns and ten cycles are taken so that the cycle average error is reduced and became stable. Two C-programswere writtenon this back propagation method to find the updated weights where the cycle average error is under the acceptable limits. The process for reaching each intermediate location along a path by the back propagation method for the case of six degrees of freedom Stanford manipulator is described by five stages: from the first program after getting the weights updated last cycle updated weights are taken.

1. Assuming that the arm moves from the point (x, y, z) to (x_f, y_f, z_f). Incremental values (dx, dy, dz) are calculated and used to predict the corresponding incremental joint angles (dθ_1, dθ_2) using back propagation neural network method as described above, which are o_1 and o_2. Using inverse kinematics, angles θ_1, θ_2 are calculated using (x, y, z). New joint angles θ_1', θ_2' are calculated by adding the increments to the previous angles as follows: $\theta_1' = \theta_1 + d\theta_1$ $\theta_2' = \theta_2' + d\theta_2$
2. Forward kinematics model calculates the new Cartesian positions of the end effector (x', y', z') that corresponds to these joint angles.
3. If the distance between (x_c, y_c, z_c) and (x', y', z') is greater than the acceptable error, then the new incremental distances$dx' = x_c - x'$ $dy' = y_c - y'$ $dz' = z_c - z'$

are calculated and fed to the network as inputs to compute additional joint increments. The iterative procedure is repeated until the error is under acceptable limits. A new point is selected and stages 1 to 3 are to be repeated until the required end point is reached. For example to trace a straight line from (-0.3, 0.9, 0.4) and (0.6, 0.2, 0.3) equally spaced Cartesian points along the test line are regarded as the intermediate positions that the end effector of three-link arm actually moves to. The iterative process can be stopped at appropriately low value of Cartesian root mean square error. A schematic block diagram of back propagation neural network of Stanford manipulator is shown in Figure 5

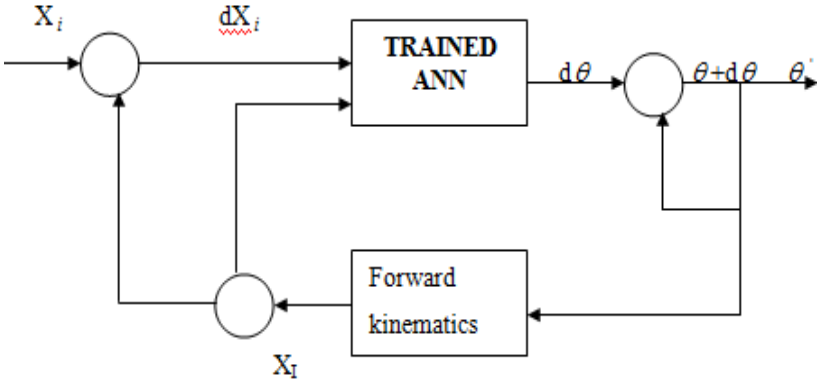


Fig5: Schematic block diagram of back propagation method of Stanford manipulator

IV. RESULTS

A. Weights obtained after taking 4 patterns and 10 cycles

The weights W*ji* (connection weights between hidden and input layers) and V*kj*(connection weights between output and hidden layers) obtained after the execution of the C program.

W00=496.218992 W10=-4.431115 W20=556.969740
W01=190.561914 W11=1.209471 W21=206.150620
W02=482.755379 W12=-1.135833 W22=534.799036
W03=630.073987 W13=-5.671727 W23=706.367955
W04=963.540177 W14=-2.481522 W24=1071.004398
W05=1243.658596 W15=-5.283731 W25=1384.489837
V00=39.191430 V01=-34.224281 V02=41.478174
V10=38.079149 V11=-32.746045 V12=40.243054

B. Intermediate Positions alonga Path

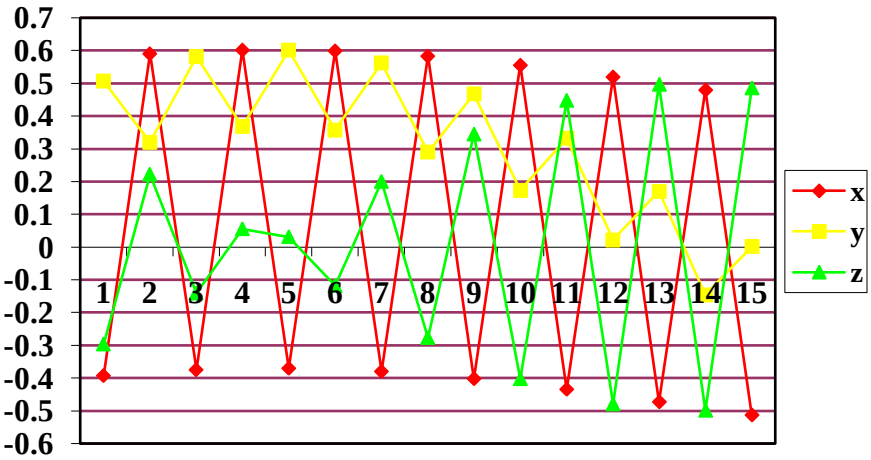
Table 1: Intermediate positions along a path

S. NO	x	y	Z	dx	Dy	dz
1	-0.39336	0.50723	-0.29662	0.99336	-0.30723	0.59662
2	0.59003	0.32013	0.22222	0.00997	-0.12013	0.07778
3	-0.37552	0.58231	-0.14107	0.97552	-0.38231	0.44107
4	0.60136	0.36780	0.05563	-0.00136	-0.16780	0.24437
5	-0.37105	0.60111	0.03150	0.97105	-0.40111	0.26850
6	0.59883	0.35716	-0.11767	0.00117	-0.15716	0.41767
7	-0.38050	0.56135	0.20027	0.98050	-0.36135	0.09973
8	0.58274	0.28949	-0.27678	0.01726	-0.08949	0.57678
9	-0.40273	0.46783	0.34489	1.00273	-0.26783	-0.04489
10	0.55504	0.17296	-0.40251	0.04496	0.02704	0.70251
11	-0.43506	0.33183	0.44790	1.03506	-0.13183	-0.14790
12	0.51907	0.02162	-0.47969	0.08093	0.17838	0.77969
13	-0.47358	0.16975	0.49690	1.07358	0.03025	-0.19690
14	0.47916	-0.14627	-0.49901	0.12084	0.34627	0.79901
15	-0.513	0.001	0.485	1.11	0.198	-0.185

C. Incremental Angles for Each Loop obtained from C-Program

dt1=-180.00000 dt2=-170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000
dt1=180.00000 dt2=170.00000

The cycle average error of four patterns after ten cycles is under acceptable limit. The relationship between the intermediate positions along a path for fifteen loops is shown in the below graph 1.



Graph : Relation between intermediate position along a path

V. ELBOW MANIPULATOR

The Elbow manipulators are anthropomorphic. The first three axes form the positioning arm and the last three axes form the orienting wrist. This arrangement maximizes the reachable workspace of the robot for a given total link length. It is a six degrees of freedom robot and all joints are revolute. The figure of Elbow manipulator is shown in the Fig 5.5

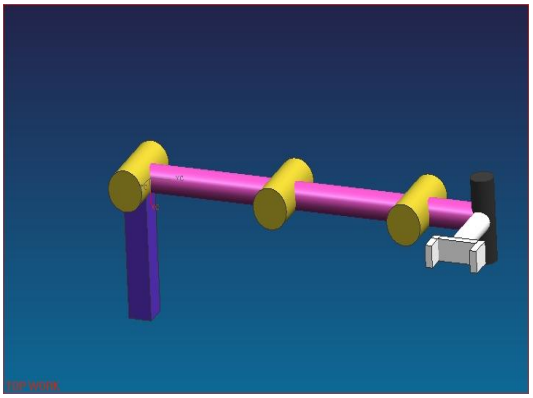


Fig.6 Elbow Manipulator

Table 2.Joint link parameters table of Elbow Manipulator

Joint	θ	α	a	D
1	θ_1	90	0	0
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0
4	θ_4	-90	a_4	0
5	θ_5	90	0	0
6	θ_6	0	0	0

Direct kinematics equations of Elbow manipulator

$u_x = c_1[c_5 c_6 c_{234} - s_6 s_{234}] - s_1 s_s c_6$

$u_y = s_1[c_5 c_6 c_{234} - s_6 s_{234}] + c_1 s_s c_6$

$u_z = c_5 c_6 s_{234} + c_{234} s_6$

$v_x = c_1[-c_5 s_6 c_{234} - c_6 s_{234}] + s_1 s_s s_6$

$v_y = s_1[-c_5 s_6 c_{234} - c_6 s_{234}] - c_1 s_s s_6$

$v_z = -c_5 s_6 s_{234} + c_{234} c_6$

$w_x = s_5 c_1 c_{234} + s_1 c_5$

$w_y = s_5 s_1 c_{234} - c_1 c_5$

$w_z = s_5 s_{234}$

$p_x = c_1[a_2 c_2 + a_3 c_{23} + a_4 c_{234}]$

$p_y = s_1[a_2 c_2 + a_3 c_{23} + a_4 c_{234}]$

$p_z = [a_2 s_2 + a_3 s_{23} + a_4 s_{234}]$

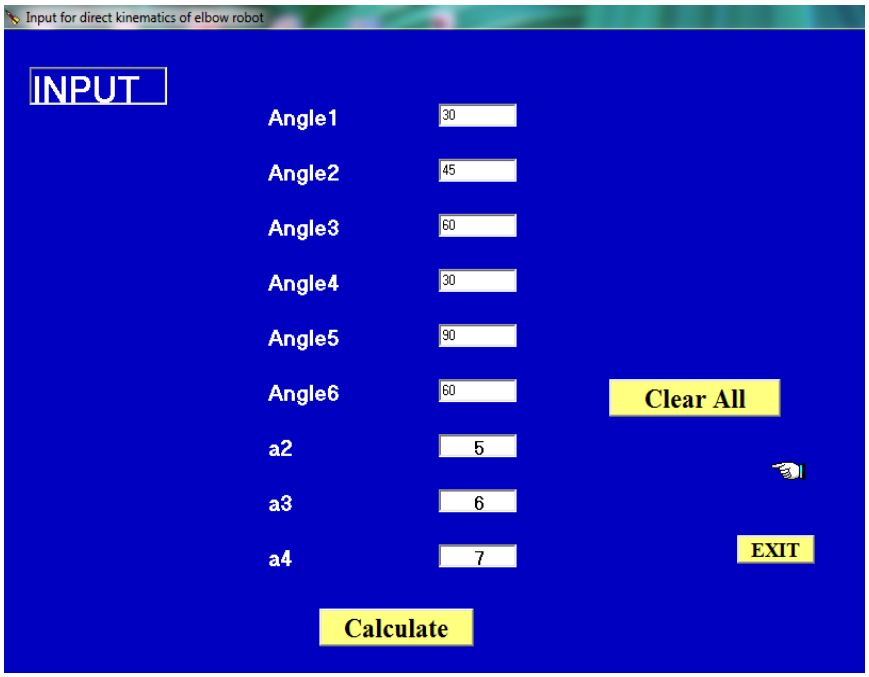


Fig 7.Input data for Direct kinematics of Elbow manipulator

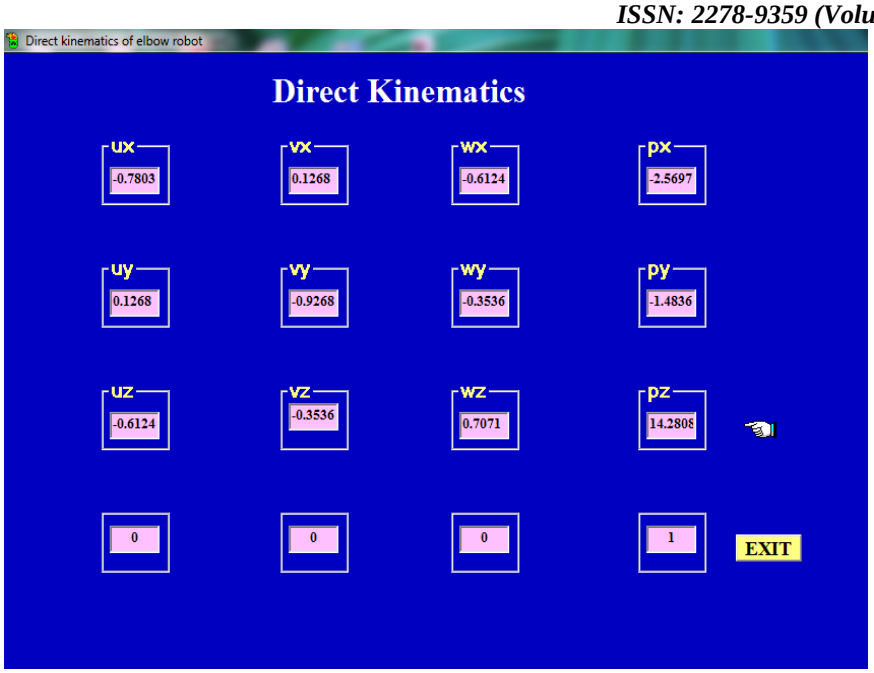


Fig8. Direct kinematics of Elbow manipulator

VI. CONCLUSIONS

An iterative strategy has been suggested to solve the inverse kinematic problem of robot manipulators. Within the iterative strategy, an ANN whose network architecture is different from the conventional ANN was used for predicting incremental joint angles when given absolute and incremental Cartesian positions. Besides the ANN an analytical forward kinematic model was used in the iterative process for transforming the joint angles into new Cartesian position, which was then used to calculate the real error that is iteratively reduced. An error that exists in the ANN of the iterative strategy has been found. The error, which has different values for each goal position, can be used to compensate for the error of the ANN. The tests showed that the iteration strategy with back propagation could be successfully applied in solving the inverse kinematics problem of the six degrees of robot manipulator. Comparing the errors with those from the conventional ANN method, the proposed back propagation method improves the accuracy of the solution, reducing the average error.Here in this paper modeling and direct kinematics of elbow manipulator is also presented.

REFERENCES

[1] Karlik.B.,Serkan.Aydin. 2000. An improved approach to the solution of inverse kinematics problems for robot manipulators.*Engineering applications of artificial intelligence*, Vol.13, Issue-2, 1159-164.

[2] Martin, P., Jose Del R. Millan.2000. Robot arm reaching through neural inversions and reinforcement learning. *Robotics and autonomous systems.* Vol.31, Issue-4. 227-246.

[3] RasitKoker., Cemil Oz., Tarik,Cakar., Huseyi Ekiz.2004.A study of neural network based inverse kinematics solution for a three joint robot. *Robotics and autonomous systems.*Vol.49, Issues 3-4, 227-234.

[4] Pramod,Gupta., and Naresh ,K.,2004.Intelligent control of robotic manipulators: experiment study using neural networks", *Mechatronics*, Vol.10, Issues 1-2, 289-305.

[5] Rasit, K.,2005. Reliability based approach to the inverse kinematics solution of robots using Elman's network. *Engineering applications of artificial intelligence*, Vol.18, Issues 6, 685-693.

[6] Abdelhameed,M.M.,1999.Adaptive neural network based controller for robots.*Mechatronics.*Vol.9,Issue-2,1,147-162.

[7] Medhat.A.M.,1998.An experiment in approximating an end effector positional error of a 6 dof manipulator using neural networks. Computers and industrial engineering.Vol.35,Issue3-4.

[8] Ozkan.M,Inoue.k,Negishi.k,Yamanaka.T.,2000.Defining a Neural network controller structure for a rubbertuatorrobot.*Neural networks*,Vol 13,Issue4-5.

[9] Ding Han,ChanSai Piu.,1997.Collision free motion planning for redundant robots using neural network processing.*Engineering applications of artificial intelligence.*Vol.10.Issue 2.179-188.

[10] Lou,Y.F.,Brunn.A hybrid artificial neural network inverse kinematic solution for accurate robot path control.*ProcInstnMechEngrs.*Vol 213,23-33

