Scientific Research and Essays Vol. 6 (13), pp. 2784-2794, 4 July, 2011
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE11.211
ISSN 1992-2248 ©2011 Academic Journals

Full Length Research Paper

A neuro-genetic approach to the inverse kinematics solution of robotic manipulators

Raşit KÖKER

[1]Department of Computer Engineering, Engineering Faculty, Sakarya University, 54187 Sakarya – Turkey.
[2]Computer Engineering Department, Faculty of Engineering and Natural Sciences, International University of Sarajevo, Hrasnicka cesta 15, 71000 Sarajevo, Bosnia and Herzegovina. E-mail: rkoker@sakarya.edu.tr

Accepted 23 May, 2011

In this paper, a neuro-genetic approach is proposed for the inverse kinematics problem solution of robotic manipulators. The proposed solution method is based on using neural networks and genetic algorithms in a hybrid system. Neural networks have been used by many researchers in the inverse kinematics solution. Since the neural networks work with an acceptable error, the error at the end of learning has to be minimized for sensitive applications. This study is based on using genetic algorithms to minimize this error. A case study is presented for a 6 degree of freedom robot. In the neural network part, three Elman networks are separately trained and then used in parallel since one Elman network may give better result than the other two ones. These three results are placed in the initial population of the genetic algorithm. The end effector position error is defined as the fitness function and genetic algorithm is implemented. Thus, the error is reduced in micrometer levels.

Key words: Elman neural networks, error minimization, six-degree-of-freedom robot, genetic algorithms, robotics.

INTRODUCTION

The fundamental problem of robot kinematics deals with mapping between joint space and Cartesian space. The mapping from joint space to Cartesian space is known as direct kinematics and the mapping from Cartesian space to joint space is known as inverse kinematics. In many robotic applications, the inverse kinematics problem solution is more significant and interesting, because of the fact that the robot tasks are specified in Cartesian task space whereas lower level joint controllers needs joint space coordinate that requires the solution of the inverse kinematics problem (Kuroe et al., 1993). There are three traditional methods used to solve inverse kinematics problem: geometric (Featherstone, 1983; Lee, 1982), algebraic (Duffy, 1980; Manocha and Canny, 1994; Paul et al., 1981; Fu et al., 1987) and iterative (Korein and Balder, 1982) methods. Each method has some disadvantages. The algebraic methods do not guarantee closed form solutions. In case of using geometric methods, closed form solutions for the first three joints of the manipulator must exist geometrically. The iterative methods converge to only a single solution and this solution depends on the starting point. For simple manipulator geometry, the problem is solved using trigonometry approaches involving tedious mathematical steps. If the joint structure of the manipulator is more complex, the inverse kinematics solution by using these traditional methods is a time consuming study (Koker et al., 2004). In the recent years, new methods were presented to solve inverse kinematics problem such as artificial neural networks and the optimal algorithm. The computation of inverse kinematics using artificial neural network is particularly useful where shorter calculation times are required, such as in real-time adaptive robot control. In other words, for a more generalized m-degrees of freedom manipulator, traditional methods will become prohibitive due to the high complexity of mathematical structure of the formulation. To compound the problem further, robots have to work in the real world that cannot be modeled concisely using mathematical expressions (Koker et al., 2004). However, as it is a well known fact that the neural networks are working with an acceptable solutions and cannot give a precise solution, but it approaches to the solution. That is why neural network based inverse kinematics problem solution needs error minimization at the end effector.

Uicker et al. (1964) presented a study about using iterative solutions for finding inverse kinematics solution of robotic manipulators. Nearchou (1998) proposed an evolutionary approach to obtain a unique solution for the inverse kinematics problem of nonredundant and redundant robotic manipulators based on a modified binary-coded genetic algorithm (GA). The multiplicity resolution issue of a PUMA robot was solved through the minimization of total joint displacement and the closest solution was also evaluated in the joint space relative to the current configuration. In his study, the superiority of the evolutionary approach over the pseudo-inverse method and the simple binary-coded genetic algorithm has been established. Furthermore, the computation of the Jacobian matrix is not required in the evolutionary approach; therefore any problem related to the inversion of this matrix like singularities has been overcome. However, the proposed approach has some certain limitations. Khawaja et al. (1998) presented a study about the inverse kinematics problem solution of an arbitrary robotic manipulator based on a binary-coded genetic algorithm. Their approach was used to compute the motion of an n-R robotic manipulator following a specified end-effector path. The genetic algorithms may come up with multiple satisfying solutions because they search the whole solution space in parallel. The sum of squares of the discrete joint velocities is computed and described as an additional fitness function to guide the evolutionary process to a single solution. The multiple configurations of a robot existing because of the multimodal nature of the inverse kinematics problem are therefore not available at the end of the search. Bingul and Ertunc (2005) proposed a neural network approach using back propagation algorithm to solve the inverse kinematics problem of a robotic manipulator not having an analytical inverse kinematics solution. Their approach has large errors in the joint angles as a disadvantage and inability of the approach is providing multiple solutions of the inverse kinematics problem. Wang and Lienhardt formulated and solved a kinematic model as an optimization problem for ABB robotic manipulators (Kesheng and Jonathan, 2005). The aim was to analyze and evaluate the performance of the genetic algorithms in the optimization of the robot kinematic accuracy. In their algorithm, small changes in the kinematic parameter values represent the parent and offspring population and the end-effector error represents the fitness function.

A numerical example was given to show the convergence and effectiveness of the given solution model. The multiple solutions of the inverse kinematics problem were not provided by this solution approach. In this paper, a neuro-genetic hybrid approach is applied to the inverse kinematics problem solution of PUMA 560 robotic manipulator with six joints by using simulation software. The main point of this study is the minimization of end effector position error by using the random search feature of the genetic algorithms. The proposed solution schema includes a neural network block, which has three Elman networks working in parallel, and a genetic algorithm. Due to the fact that a neural network based solution system works with an acceptable error, this error needs to be minimized for some sensitive tasks. That is the reason why the genetic algorithm is used in this study to minimize the error at the end of inverse kinematics learning of the Elman networks. Elman network has feedback loops, which has a profound impact on the learning capability of the network, and on network's performance. Because of this advantage Elman network is used in the neural network part. Ten digits decimal parts of the obtained solutions from the neural networks were placed in the initial population of the genetic algorithm to find the best ten digits for the decimal part. The aim of using three Elman networks in parallel is that one of these networks can give better result than other ones. The end effector error was defined as a fitness function in the genetic algorithm. Elman neural networks and the genetic algorithm were used online. In this study, the genetic algorithm was used to find the best decimal part of the neural network based solution and the end effector error was reduced to level of micrometers. An illustrative example for the genetic algorithm implementation is also presented.

THE DENAVIT-HARTENBERG METHOD FOR THE KINEMATIC ANALYSIS

The Denavit-Hartenberg (D-H) method deals with the allocation of coordinate frames to each link by using a set of rules to locate the frame origin and the orientation axes. The poses of subsequent links are then described by the homogeneous transformation matrix that transforms the frame attached to the link i-1 into a frame fixed to link i. This transformation is obtained from simpler transformations representing the three basic translations along, and three rotations about the frames' x-, y- and z- axes (Paul et al., 1981). This paper takes the PUMA 560 robotic manipulator as an example. As shown in Figure 1, the PUMA robot has six joints. In order to analyze the inverse kinematics problem, Denavit-Hartenberg frames are given in Figure 1 (Fu et al., 1987). D-H parameters of PUMA robotic manipulator is given in Table 1, where θ_i is the joint angle, d_i is the joint offset, a_i is the link length, and α_i is the link twist. According to the D-H parameters θ_i, d_i, a_i and α_i, homogeneous transformation matrix A_i I = 1, 2,...,6 can be obtained. If the θ_i is given, the cartesian position of the end effector can be computed by using Equation 1. The matrix T_6 describes the position and also the orientation of the manipulator. The orientation of the hand is described according to the roll-pitch-yaw (RPY) rotations.

$$T_{[illegible]-[illegible]} = T_6 = A_1 . A_2 . A_3 . A_4 . A_5 . A_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

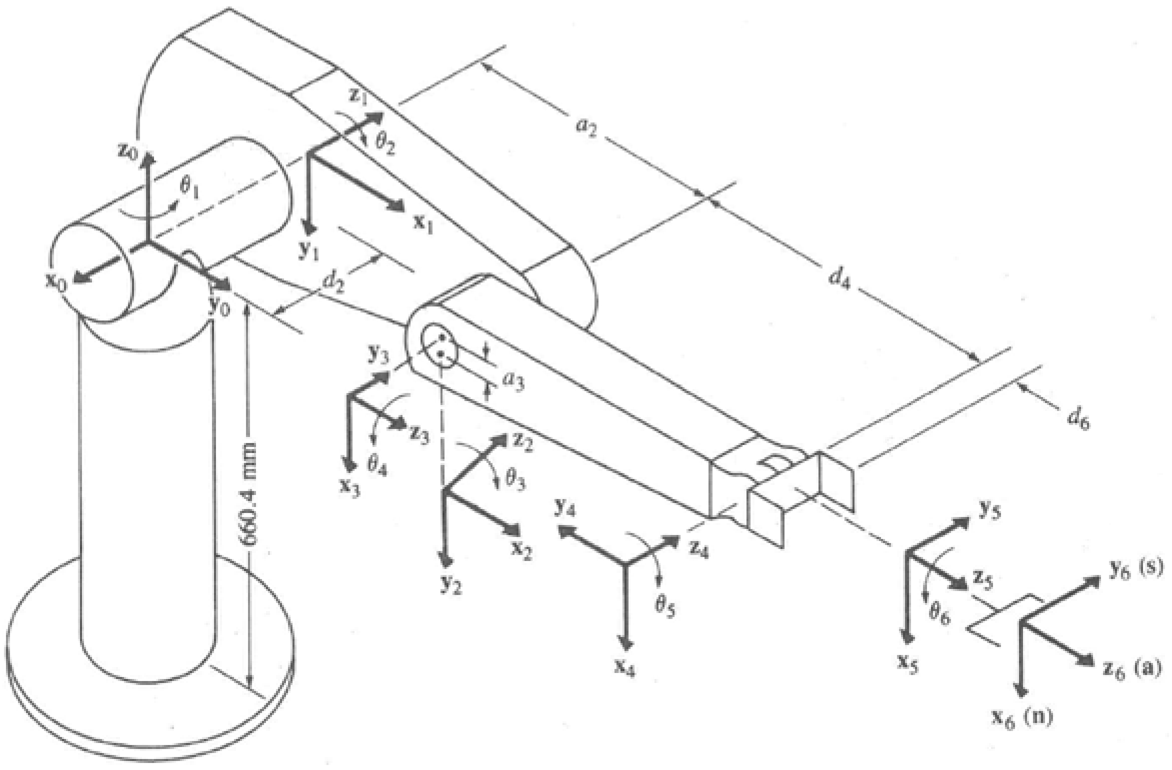


Figure 1. PUMA 560 robotic manipulator.

Table 1. D-H parameters of PUMA 560 robotic manipulator.

Joint i	θ_i (°)	α_i (°)	a_i (mm)	d_i (mm)	Joint range (°)
1	90	-90	0	0	-160 to +160
2	0	0	431.8	149.09	-225 to +45
3	90	90	-20.32	0	-45 to +225
4	0	-90	0	433.07	-110 to +170
5	0	90	0	0	-100 to +100
6	0	0	0	56.25	-266 to +266

The matrix T_6 describes the position and also the orientation of the manipulator. The orientation of the hand is described according to the RPY rotations (Kozakiewicz et al., 1991).

$$RPY(\phi_z, \phi_y, \phi_x) = Rot(z, \phi_z)\, Rot(y, \phi_y)\, Rot(x, \phi_x) \quad (2)$$

If T_6 matrix is solved, then:

$$\phi_z = a\tan 2(n_y, n_x), \quad (3)$$

$$\phi_y = a\tan 2(-n_z, n_x \cos\phi_z + n_y \sin\phi_z), \quad (4)$$

$$\phi_x = a\tan 2(a_x \sin\phi_z - a_y \cos\phi_z, o_y \cos\phi_z - o_x \sin\phi_z) \quad (5)$$

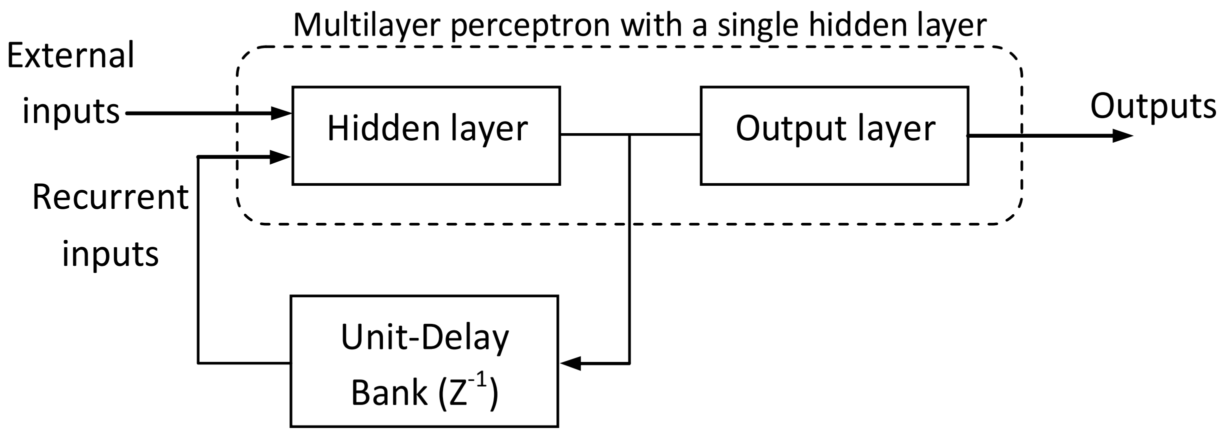
These obtained equations give information about the position and orientation of the robot according to the real world coordinate frame. The coordinate frames for each joint are used to describe the position and orientation of robot. Equation 6 shows the inverse kinematics solution as a function.

$$f_{inverse\ kinematics}(x, y, z, \phi_z, \phi_y, \phi_x) = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \quad (6)$$

It can be clearly seen that the forward kinematics problem can be easily solved for the given joint angles (Karlik and Aydin, 2000; Köker, 2005). These Equations 1 to 6 have been used to prepare training, verification and test data set of the neural network.

RECURRENT NEURAL NETWORKS

A recurrent neural network is distinguished from a feed forward neural network because of having at least one feedback loop. The presence of these feedback loops has a profound impact on the learning capability of the network, and on the performance of the



Multilayer perceptron with a single hidden layer
External inputs
Recurrent inputs
Hidden layer
Output layer
Outputs
Unit-Delay Bank (Z^{-1})

Figure 2. The block diagram of a simple recurrent neural network.

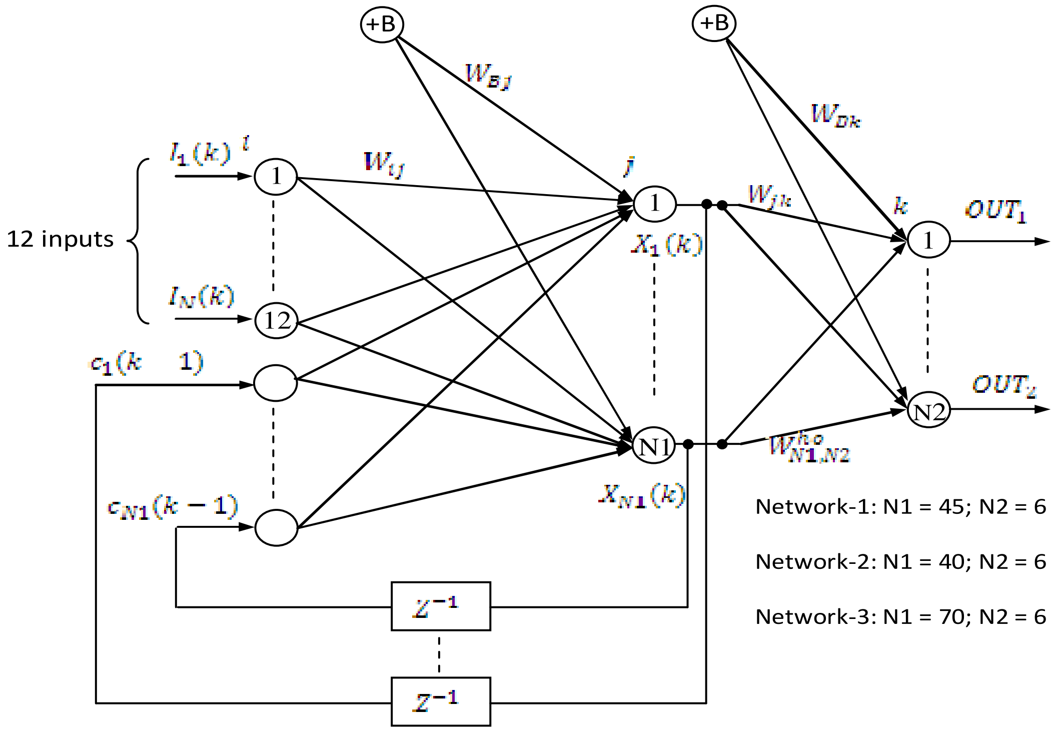


Figure 3. The structure of the Elman network used in the inverse kinematics learning.

network. Additionally, these feedback loops involve the use of particular branches composed of unit-delay elements shown as Z^{-1} that result in a nonlinear nature of the neurons. Nonlinear dynamics has a key role in the storage function of a recurrent network. A block scheme of the recurrent neural network is given in Figure 2. Some important properties of recurrent neural network can be listed as given as follows: firstly, the recurrent neural networks are universal predictor of nonlinear dynamic systems, provided that they are designed with an adequate number of hidden layers. Secondly, they are locally controllable and observable, provided that their linearized versions satisfy certain conditions. Thirdly, given any finite-state machine, a recurrent neural network regarded as a black-box machine can be built, will behave like that finite-state machine. And lastly, a recurrent neural network shows a meta-learning capability. In other words, its learning ability can be explained as learning to learning. Actually, using recurrent neural networks in computing, control and signal processing applications really give precise benefits. Especially, in the robot control using recurrent neural networks will be beneficial because of dynamical structure of robotic manipulators (Haykin, 2009). The expressions for the outputs of each hidden layer neurons are given in Equations 7 and 8 using the parameters shown in Figure 3:

$$[illegible]_{[illegible]}([illegible]) = \frac{1}{\left(1 + [illegible]\left([illegible]^{h}_{[illegible]}([illegible])\right) + \sum_{[illegible]=1}^{[illegible]} [illegible]^{h}_{[illegible]}([illegible])\,[illegible]_{[illegible]}([illegible]) + \sum_{[illegible]=[illegible]+[illegible]1}^{[illegible]+[illegible]1} [illegible]^{h}_{[illegible]}([illegible])\,[illegible]_{[illegible]}([illegible]-1)\right)} \qquad (7)$$

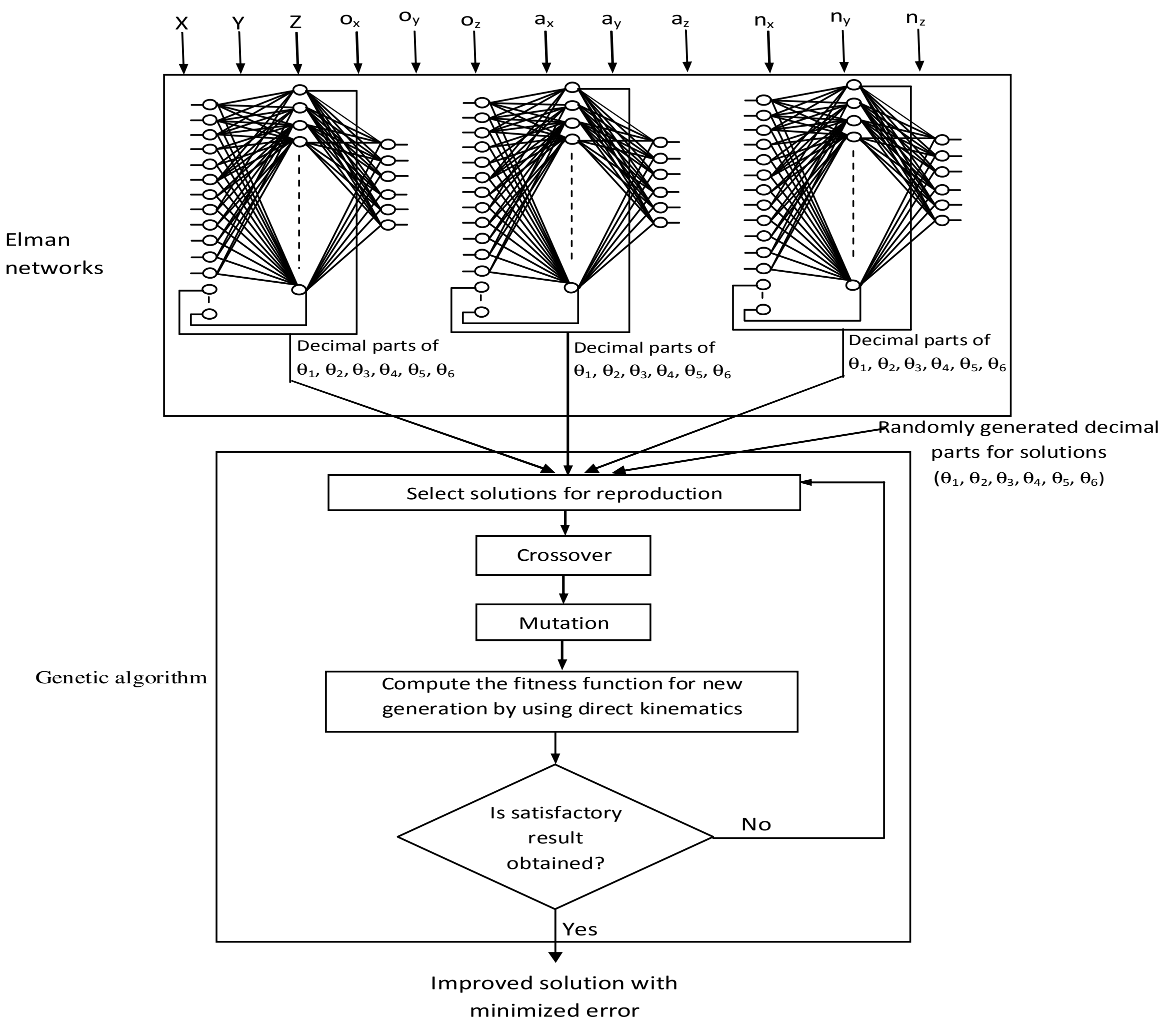

X
Y
Z
o_x
o_y
o_z
a_x
a_y
a_z
n_x
n_y
n_z
Elman networks
Decimal parts of $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$
Decimal parts of $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$
Decimal parts of $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$
Randomly generated decimal parts for solutions ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$)
Genetic algorithm
Select solutions for reproduction
Crossover
Mutation
Compute the fitness function for new generation by using direct kinematics
Is satisfactory result obtained?
No
Yes
Improved solution with minimized error

Figure 4. The schematic diagram of the proposed inverse kinematics solution schema.

Elman network output can be expressed as:

$$output_n(k) = \frac{1}{\left[1 + \exp\left[x_i(k) + \sum_{j=1}^{m} w_{ji}^{h}(k)\, output_j(k)\right]\right]} \quad (8)$$

Where $x_i(k)$ is the ith input to the network (external inputs), $y_j(k-1)$ is recurrent input to the network, $w_{1,0}^{h}$ is the weight value from the first neuron of input layer to first neuron of hidden layer, $y_j(k)$ is the ith recurrent input, $w_{1,0}^{h}$ is the weight value from the first neuron of input layer to the first neuron of hidden layer, $output_n(k)$ is the nth output (Temurtas et al., 2004).

NEURO-GENETIC APPROACH TO INVERSE KINEMATICS SOLUTION

Here, proposed solution scheme is explained. The system is composed of the neural network block and a genetic algorithm as denoted in Figure 4. The given block schema is online implemented for the inverse kinematics solution of Puma 560 robotic manipulator with six joints by using designed neural networks and genetic algorithm. Here, it is mainly aimed to reduce the prediction error of the neural network by using a genetic algorithm.

Training of the Elman networks

In this study, neural networks and genetic algorithms were used together in the inverse kinematics problem solution of a six-joint Puma 560 robotic manipulator to minimize the error at the end effector. Firstly, three Elman neural networks were trained for the inverse kinematic solution. The following step is genetic algorithm implementation. Genetic algorithm was used online to improve the decimal part of the Elman neural network result up to ten digits. The neural networks for inverse kinematics learning are working with an acceptable error. That is the reason why in the neural network part, three Elman networks have been trained by using separately prepared data sets. The aim of using more than a unique neural network is trying to get possibly better results. For instance, for a desired solution for any joint is “35.0019726393”, these three neural networks may give the solution for this joint as “35.003619870”,

Table 2. Training parameters and the error values of Elman networks.

	Elman network-1	Elman network-2	Elman network-3
Learning rate	0.40	0.35	0.40
Momentum coefficient	0.70	0.80	0.75
Activation function	Sigmoid	Sigmoid	Sigmoid
Number of neurons in the hidden layer	45	40	70
Iteration number	800,000	1,100,000	1,000,000
Sample size in training set	5000	5000	5000
Sample size in test set	2000	2000	2000
MSE values for training set	1,45622	2,39754	1,87345
MSE values for validation set	1,39012	1,98007	1,78809
MSE values for test data set	1,12003	1,54099	1,44199

"34.991358752" and "35.012980025", respectively. In this case, the solution of the second network is not a good one to be improved. Because in the genetic algorithm we only improve the decimal part of the solutions and then the improved solution was joined with the integer part. The decimal parts of these three Elman network solutions exist in the initial population of the genetic algorithm, and good ones will survive. As it is clearly seen, the decimal part of the second network of "34.991358752" is not good since the integer part of the solution is not used in the genetic algorithm. These three feed forward Elman neural networks with sigmoid activation function were used to solve inverse kinematics problem. Each network was trained with the least error as much as possible. A case study about using neural networks in parallel was previously presented by Koker (2005). The block diagram of the proposed solution was given in Figure 4. Training of a neural network is the process of setting the best weights on the inputs of each of the units. The aim is to use the training set to produce weights where the output of the network is as much closer to the desired output values for many of the examples in the training set (Shanthi et al., 2009). The training set is a part of the input dataset used for neural network training, that is for adjustment of network weights.

In this study, for the training, firstly, three different 5000 data, which consists of the (θ_1, θ_2, θ_3, θ_4, θ_5 and θ_6) joint angles according to the different (X, Y, Z, o_x, o_y, o_z, n_x, n_y, n_z, a_x, a_y and a_z) cartesian coordinate parameters, were generated separately using Equations 1 to 6 in the work volume of robotic manipulator. It has been tried to obtain well-structured learning sets to make the learning process successful and easy. These values were recorded in the files to form the learning sets of the networks. Each 5000 of these data was used in the training of Elman networks. As a validation set, 2000 data was prepared. The validation set is a part of the data used to tune network topology or network parameters other than weights. For example, it is used to define the number of units to detect the moment when neural network performance started hidden to deteriorate. To choose the best network (that is by changing the number of units in the hidden layer) the validation set is used. As it is well known too much training can cause over fitting, the validation set also have to be used in an early stopping of the training process. For the test set, 2000 data were prepared and used in the test of each neural network to see their success for the same data set. The test set is a part of the input data set used to test how well the neural network will perform on new data. The training process was completed until the error was possibly minimum for each network. Mean square error (MSE) has been used in the error computation and the training parameters and test results have been given in Table 2. The neural networks have been trained using the neural networks toolbox of the Matlab. Conventional back propagation algorithm that uses a threshold with a sigmoid activation function and gradient-descent learning algorithm has been used. The momentum coefficient, learning rate and number of neurons in the hidden layers were determined, experimentally.

The genetic algorithm

The genetic algorithms are dealing with directed random search emulating the process of genetic evolution found in nature to perform artificial evolution. They were developed by Holland (1975) in the early 1970s and since then have been applied successfully to many different complex search problems. Naturally, organisms have certain characteristics, which affect their ability to survive or reproduce. These characteristic features are found in their genes. Natural selection guarantees that genes from a strong individual are presented in greater numbers in the next generation than those from a weak individual. Over a number of generations, the fittest individuals have the highest probability of survival and tend to increase in numbers, while the less fit individuals tend to die out. The main important point here is survival of the fittest and this point constitutes the basic idea behind genetic algorithms (Nearchou, 1998; Cakar et al., 2005). In this study, a genetic algorithm is used to search the best decimal parts up to 10 digits by using obtained solution from neural network block. The first thing in the performance of a genetic algorithm is coding. After coding process, the genetic algorithm operators are applied on the chromosomes. Crossover, mutation and reproduction processes continue until an optimal solution is obtained. In this study, the evaluations of the offsprings are done by using fitness function, which is defined as the end effector error, namely the distance between the target point and end effector. The genetic algorithm tries to find the best ten digits for the decimal part by minimizing this end effector error. An illustrative example is given as follows:

Coding

The decimal parts of the obtained neural network solutions are used in the genetic algorithm implementation. The genetic algorithm is designed to search the best ten digits decimal part for the solution. Binary coding is used to represent these ten digits. A sample representation of the coding process for a neural network solution is given in Table 3.

Initial population

In this study, the initial population is not produced totally randomly. The decimal parts of the obtained solutions from neural network are

Table 3. Representation of the coding process for a neural network solution.

Joint angle	Neural network result	Decimal part	Binary representation
θ_1	30,2768940110	2768940110	0010100101000010101010110001001110
Θ_2	26,0073200971	0073200971	0000000100010111001111010101001011
Θ_3	35,8862341501	8862341501	1000010000001111001001100101111101
Θ_4	10,7635920333	7635920333	0111000111001000101110010111001101
Θ_5	20,9100783441	9100783441	1000011110011100101110111101010001
Θ_6	35,1107296387	1107296387	0001000010000000000000000010000011

Table 4. Randomly generated chromosome.

Randomly generated chromosome	Actual number
1000101111011000111101010001110110	9385006198
0011011111010111100000011101010011	3747481427
1111010111000010101111000010101010	16492720298
1010011110101000010100010111111101	11251303933
0000011110010001110101011110010101	507991957
1101111111110010111001010001110001	15028950129

Table 5. A sample representation of chromosome operation.

	The matched chromosome pairs			After the crossover operation
1st pieces	0010100101000010101010110001	001110	X	0010100101000010101010110001110110
	1000101111011000111101010001	110110		1000101111011000111101010001001110
2nd pieces	0000000100010111001111010101	001011	X	0000000100010111001111010101010011
	0011011111010111100000011101	010011		0011011111010111100000011101001011
3rd pieces	1000010000001111001001100101	111101	X	1000010000001111001001100101101010
	1111010111000010101111000010	101010		1111010111000010101111000010111101
4rd pieces	0111000111001000101110010111	001101	X	0111000111001000101110010111111101
	1010011110101000010100010111	111101		1010011110101000010100010111001101
5th pieces	1000011110011100101110111101	010001	X	1000011110011100101110111101010101
	0000011110010001110101011110	010101		0000011110010001110101011110010001
6th pieces	0001000010000000000000000010	000011	X	0001000010000000000000000010110001
	1101111111110010111001010001	110001		1101111111110010111001010001000011

also put in the initial population. Other chromosomes have been generated randomly. A sample of randomly generated chromosome is given in Table 4.

Crossover process

The crossover process is done by crossing obliquely from the randomly determined two chromosomes. At the end of crossover operation, two new chromosomes will be obtained. Let us show this crossover operation by using two chromosomes that were given earlier. As it is explained, one of these chromosomes was obtained from neural network solution and other one was randomly generated (Table 5). Here “X” refers to crossover operation. The randomly selected crossover points are shown by a dotted line. After the crossover operation, obtained new offsprings were given in Table 6.

Mutation process

In the mutation process, a gene is randomly selected inside the

Table 6. New offsprings obtained from crossover operations.

New offspring #1	0010100101000010101010110001110110	2768940150
	0000000100010111001111010101010011	73200979
	1000010000001111001001100101101010	8862341482
	0111000111001000101110010111111101	7635920381
	1000011110011100101110111101010101	9100783445
	0001000010000000000000000010110001	1107296433
New offspring #2	1000101111011000111101010001001110	9385006158
	0011011111010111100000011101001011	3747481419
	1111010111000010101111000010111101	16492720317
	1010011110101000010100010111001101	11251303885
	0000011110010001110101011110010001	507991953
	1101111111110010111001010001000011	15028950083

Table 7. The obtained results after the mutation operation.

New result #1	New result #2
30,11358874742	30,9385006158
26,73200979	26,3747481419
35,8862341482	35,16492720317
10,7635920381	10,11251303885
20,9100783445	20, 507991953
35,1107296433	35,15028950083

chromosomes in the population according to defined mutation rate (Cakar et al., 2008). Because of this, binary coding is used; during the mutation selected gene will be just inverted. For example, if the selected gene is "1" it will be "0". To apply mutation process to the given offspring aforementioned, let us assume the first bit of the first piece of the offspring 1 is randomly selected for mutation. It will be converted from "0" to "1". The results are given in Table 7 after the mutation operation. The decimal parts are added to show the new result. The fitness function is defined as end effector error in this study. This end effector error can be computed easily in the three dimensional space by using Euclidian distance equation. The main aim of this paper is trying to minimize the error by using the random search ability of the genetic algorithm. In genetic algorithm implementation part, experimentally determined parameters were given as follows:

Population size: 100, crossover rate: 100%.
Mutation rate: 1%, max generation: 100.

Reproduction

In the population, the reproduction operator makes a copy of each gene and it is added to the candidate genes list. Basically, this guarantees that each chromosome in the current population remains a candidate to be selected for the next population. In this paper, it is aimed to find the solution, which minimizes the given fitness function. As it is mentioned previously, the fitness function is the position error as a distance between robot end effector and the target. The Cartesian coordinate information can be computed by using direct kinematics equations for any obtained new offspring. Then, this position error can be obtained easily in metric form by using three-dimensional (3-D) distance equation between two points in 3-D space as shown in Figure 5. Euclidian distance equation can be used in the distance computation and given in Equation 9. The genetic algorithm may get better chances to survive chromosomes with quite higher fitness. The living good chromosomes stay in the population. This process will be going on until an optimal solution is obtained in each population.

$$[illegible] = ((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{1/2} \quad (9)$$

Where, (x_1, y_1, z_1) refers to end effector's Cartesian position and (x_2, y_2, z_2) the Cartesian position for the obtained solution from the genetic algorithm by the way of using direct kinematic equations.

RESULTS

The graphical representation of the fitness function according to the number of generation is given in Figures 6 and 7. Population size is selected as 100 experimentally. In Figure 6, the genetic algorithm is trying to minimize the defined fitness function as it is evidently seen. When the optimal results are approximating to the zero, it is not clearly seen on the graphic after 57th generation. That is the reason why to expose the region near zero on the graphic, it is also sketched by using a logarithmic scale in Figure 7. In this graphic, the change of the fitness function can also be seen clearly near zero. When the genetic algorithm is stopped the obtained optimal solution is 0.00041 mm as an end effector error of the robot. On the other hand, in Figure 7, there is no change on the fitness function after seventieth generation. In Table 8, the sample outputs of the Elman neural networks and genetic algorithm are given. As it is clearly seen that the second Elman neural network gave a result for θ_3 with different integer part from the desired value. Therefore, since the floating-parts of the values were used in the genetic algorithm, the output of second

Table 8. Sample results of the inverse kinematics for puma 560 robotic manipulator.

	θ_1(°)	θ_2(°)	θ_3(°)	θ_4(°)	θ_5(°)	θ_6(°)	Error (mm.)
Desired value	30.088100123	26.4619968742	35. 0019726393	15.2706900181	18.151077294	37.7592149001	0.00029
Elman Network 1	30.0866014425	26.479100985	35. 003619870	15.2779871599	18.158000258	37.763980010	3.19358
Elman network 2	30.087569120	26.478071130	34. 991358752	15.285015411	18.15188750	37.754112980	3.896197
Elman network 3	30.075666001	26.492377582	35. 012980025	15.276088501	18.16222980	37,766598001	3.658421
The result after the genetic algorithm	30.088100118	26.4619968745	35. 0019726389	15.2706900180	18.151077291	37.7592149014	0.00041

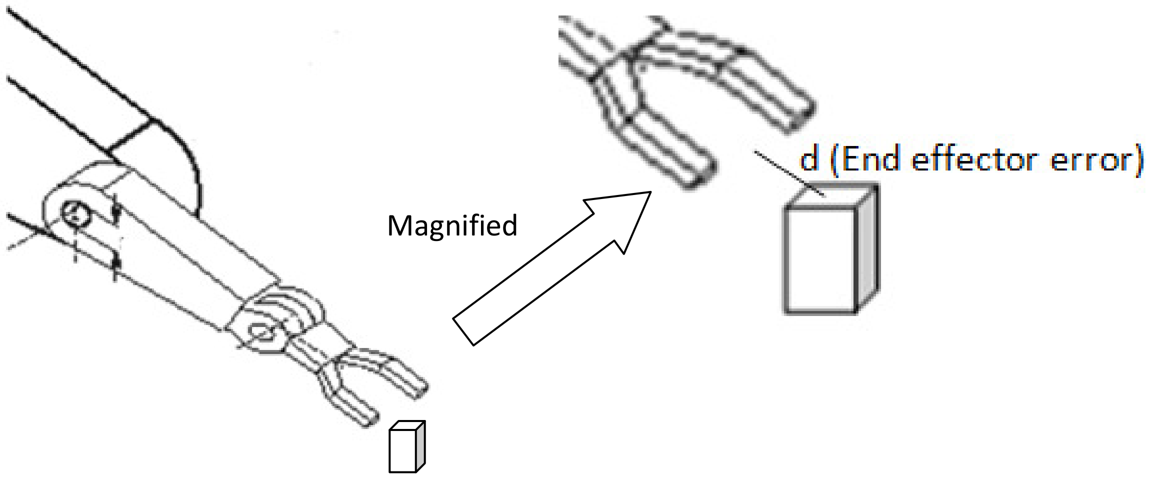


Figure 5. The symbolic end effector error representation (fitness function in the genetic algorithm).

Elman network for this joint is not good. However, since the floating parts of each Elman network are given to the genetic algorithm, the better chromosomes will survive during the generations. Since the integer part was not given to the genetic algorithm, the chromosome obtained result for θ_3 from the second Elman network will not survive. To show the performance of the genetic algorithm in this study, all solutions in test set of Elman networks are given to the genetic algorithm to be improved. After this improvement, the MSE value was computed as 0.0000008099 for the test data set.

DISCUSSION

Obtained result from the neural network based inverse kinematics solution part was improved by using genetic algorithm. The three Elman networks were used to use possibility of having the best result in the initial population of the genetic algorithm. In the initial population of the genetic algorithm, the best solution either obtained from neural network or from randomly generated solutions will survive, others will die. The genetic algorithm was stopped when there is no change on the end effector error. In this case, the obtained angle values were substituted in the direct kinematics equation of the robotic manipulator to obtain the Cartesian position information, then by using Euclidian distance equation given in Equation 9, the error for the end

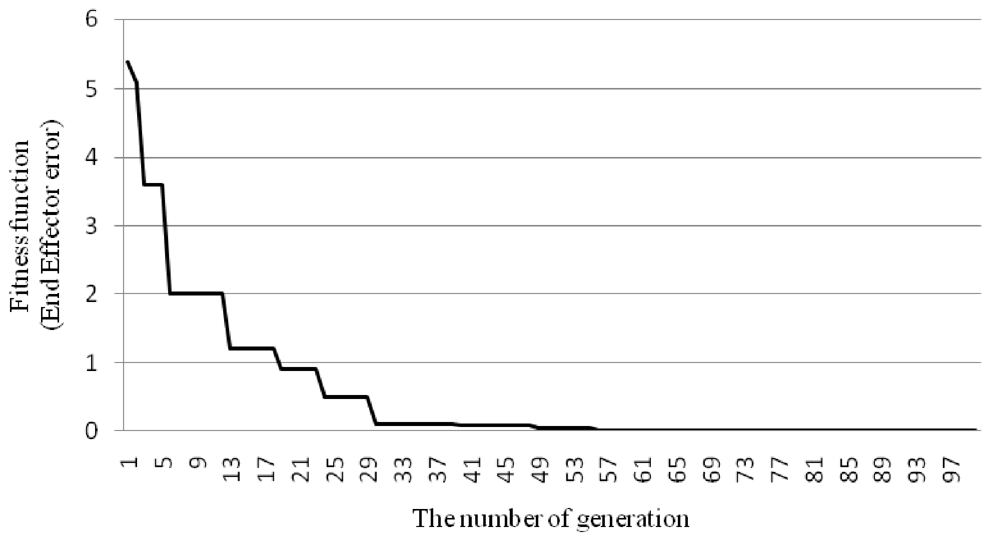


Figure 6. The fitness function change according to the number of generation.

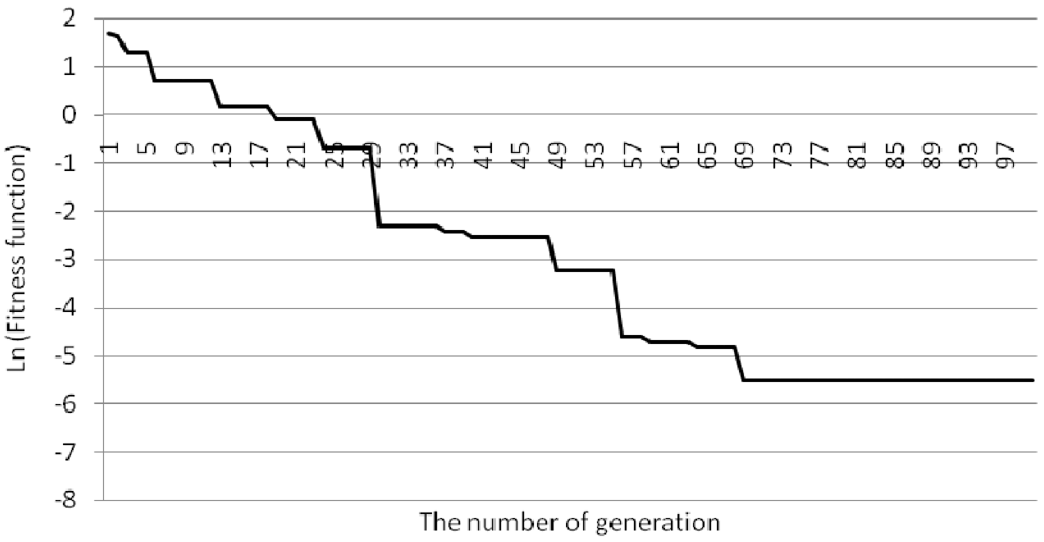


Figure 7. The fitness function change according to the number of generation using a logarithmic scale.

effector was obtained as 0.00029 mm. According to the application, the genetic algorithm can be stopped earlier if the satisfactory result is obtained. The satisfactory results were obtained as shown in Figures 6, 7 and Table 8. On the other hand, considering the MSE computations for each Elman network and whole solution system with genetic algorithm, the error for the test set was also reduced significantly.

CONCLUSIONS

In this study, a hybrid approach using neural networks and genetic algorithms for the inverse kinematics problem solution of the robotic manipulators based on end effector error minimization was presented. Since the neural networks work with acceptable error, the error has to be minimized after neural network based inverse kinematics solution in the sensitive applications. Puma 560 robotic manipulator was used as a case study to show the performance of the proposed solution method. The proposed approach combines the characteristics of genetic algorithms as an evolutionary algorithm and the neural networks. Three Elman neural networks were used in parallel to have the possibility of better decimal parts in the initial population of the genetic algorithm. The decimal part of the neural network result was improved up to ten digits by using a genetic algorithm. The genetic algorithm was used online in the solution. The result of each neural network was placed in the initial population of the genetic algorithm with randomly generated solutions. The genetic algorithm tries to find the best decimal part of the solution based on using fitness function defined as end effector error. The usage of the genetic algorithm reduced the end effector errors to micrometer levels, significantly. That is why the improved solution can be used for redundant robotic manipulators especially for critical applications.

REFERENCES

Bingul Z, Ertunc HM (2005). Applying neural network to inverse kinematic problem for 6R robot manipulator with offset wrist. in: Proc. of the 7th International Conference on Adaptive and Natural Computing Algorithms. Portugal.

Cakar T, Yıldırım MB, Barut M (2005). A neuro-genetic approach to design and planning of a manufacturing cell. J. Intell. Man., 16: 453-462.

Cakar T, Koker R, Demir HI (2008). Parallel robot scheduling to minimize mean tardiness with presedence constraints using a genetic algorithm. Adv. Eng. Software, 39: 47-54.

Duffy J (1980). Analysis of Mechanism and Robot Manipulators. Wiley. New York.

Featherstone R (1983). Position and velocity transformation between robot end-effector coordinate and joint angle. Int. J. Robotic. Res., 2(2): 35-45.

Fu KS, Gonzalez RC, Lee CSG (1987). Robotics-Control, Sensing, Vision and Intelligence. McGrow-Hill, pp. 36-39.

Haykin S (2009). Neural networks and Learning Machines. Third Ed., Pearson. New Jersey, pp. 818-834.

Holland JH (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press. Ann Arbor. MI.

Karlik B, Aydın S (2000). An improved approach to the solution inverse kinematics problems for robot manipulators. Eng. Appl. Artif. Intell., 13: 159-164.

Kesheng W, Jonathan L (2005). Robot kinematic calibration using genetic algorithms. Intell. Prod. Machines Syst., pp. 213-218.

Khawaja AA, Rahman MO, Wagner MG (1998). Inverse kinematics of arbitrary robotic manipulators using genetic algorithms. Advances in Robot Kinematics: Analysis and Control. Kluwer Academic Publishers, pp. 375-382.

Korein JU, Balder NI (1982). Techniques for generating the goal-directed motion of articulated structures. IEEE Comput. Graphics Appl., 2(9): 71-81.

Köker R (2005). Reliability-based approach to the inverse kinematics solution of robots using the Elman's network. Eng. Appl. Artif. Intell., 18: 685-693.

Kozakiewicz K, Ogiso T, Miyake N (1991). Partitioned neural network architecture for inverse kinematics calculation of a 6 DOF robot manipulator. Proceedings of the IEEE International Joint Conference on Neural Networks, Singapore: 2001-2006.

Köker R, Oz C, Cakar T, Ekiz H (2004). A study of neural network based inverse kinematics solution for a three-joint robot. Robot. Auton. Syst., 49: 227-234.

Kuroe Y, Yasuhiro N, Takehiro M (1993). A new neural network approach to the inverse kinematics problem in robotics. IEEE Motion Control Proceeding, pp. 112-117.

Lee GCS (1982). Robot arm kinematics, dynamics and control. Comput., 15(12): 62-79.

Manocha D., Canny JF (1994). Efficient inverse kinematics for general 6r manipulators. IEEE Transact. Robot. Autom., 10(5): 648-657.

Nearchou AC (1998). Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. Mech. and Mach. Theory, 33 (3): 273–292.

Paul RP, Shimano B. Mayer GE (1981). Kinematic control equations for simple manipulators. IEEE Transact. Syst. Man Cybernetics. SMC-11 (6): 66-72.

Uicker JJ, Denavit J, Hartenberg RS (1964). An iterative method for the displacement analysis of spatial mechanisms. Trans. ASME J. Appl. Mech., 31: 309-314.

Shanthi D, Sahoo G, Saravanan N (2009). Designing an artificial neural network model for the prediction of Thrombo-embolic stroke. International Journal of Biometric and Bioinformatics (IJBB), 3(1): 10-18.

Temurtas F, Gunturkun R, Yumusak N, Temurtas H (2004). Harmonic detection using feed forward and recurrent neural networks for active filters. Elect. Power Syst. Res., 72: 33-40.