# Inverse Kinematic and Jacobian Solution for Serial Manipulator based on Optimized Neural Network

Mohammad Mahdi Azimi[1*], Payman Moallem[2]

[1]Faculty of Engineering, University of Isfahan, Isfahan, Iran

[2] Faculty of Engineering, University of Isfahan, Isfahan, Iran

*Corresponding Author's E-mail: *mo_azimi@sbu.ac.ir*

## Abstract

Singularities and uncertainty in manipulator dynamic is a major issue in kinematic control of manipulator which is obtained by applying robot model. In this paper, artificial neural networks with optimal training process and training data have been proposed as a way to solve this problem. The main idea of this approach is to use an artificial neural network to learn the characteristics of the robot instead of using an explicit model of the robot. Network is designed to have one hidden layer. The inputs are Cartesian position of the end-effctor along the X, Y, Z axis and Roll, Pitch and Yaw (RPY) orientation and linear velocity of end-effector. Network outputs is defined the position and angular velocity of each joint. In a workspace without obstacle, smooth geometric trajectorys in the joint space are designed. One of the major problems of using neural network is the selecting appropriate training data for network training, therefore, after presenting a method to generate training data, Levenberg-Marquardt Algorithm applied for training the network. At the end, results compared with other neural network and real values. The results show that the proposed neural network has good performance.

**Keywords:** Neural networks, Inverse kinematics, Levenberg-Marquardt algorithm, Robot control.

## 1. Introduction

Researchs on control of the arms manipulators by using artificial intelligence has received increased attention since the last years due to their advantages [1,2].Also, neural networks have been used extensively in the fields of engineering [3,4] such as manipulator control as robot control. Singularities and uncertainty in manipulator configurations are the

main problems in robot control resulting from applying robot model. In industrial applications, it is important that the end-effector moves along predefined trajectory and in specified velocity. For manipulator control, generally control signals must be applied in such a way which end-effector move in desired trajectory. In this situation, it is important to get position and velocity from Cartesian space to joint space.

Kinematics and dynamics control are two important areas in the manipulator control. Handling of torque limits naturally leads to control algorithms based on the dynamic model of the manipulator. One of the fundamental problems in this approach is the considerable computational load by using dynamics of a real manipulator. In addition, for using torque control method, robot control needs to change. Online approach based on dynamic control, are used only in the laboratory cases with few degrees of freedom. Another method to trajectory tracking and for overcoming to the problems mentioned previously, is a method, which is called kinematics control.

Precisely, kinematics control includes an inverse kinematic transformation, which provides control efforts for each joint, proportional to the velocity and position of the end-effector. As advantage, this approach provides a simple connection with standard control structures of industrial robots. In the framework of the trajectory tracking, based on kinematics, it is usually possible to use light computational methods, this provides online applications on manipulator conventional hardware with multiple degrees of freedom. Another advantage of the kinematics control method is that it is possible to redundany identification.

A recent research that used artificial neural networks to calculate the jacobian matrix and inverse kinematics has been presented in [5]. Networks that used in this research have a lot of input and output; this issue increases the size of network, and reduces the training speed.

In [6] a neural network with one hidden layer and 55 neurons, 7 inputs and 12 outputs is defined. Data is recorded by using sensors that placed in end-effector, and the desired trajectory is designed. The proposed method has low speed and less accuracy. The proposed method has a low learning and accuracy rate, too. Neural network was presented in [6] in terms of performance is compared with another network that has 6 inputs and 6 outputs [7] and with a network with 4 inputs and 12 outputs [8]. In comparison, the network that has 7 inputs has better performance.

Data generated for training the neural network is another topic that has been investigated in [9], and the training data is generated by using the dynamic equations of the system, but the number of data required for training the neural network, also the appropriate data is obtained by using trial and error.

In this paper, is used a neural network with 7 inputs and 12 outputs with improved configuration. The neural network training time and the network size is reduced and the accuracy is improved. In previous papers, the number of training data was determined by using trial and error, but in this paper, a method is presented to calculate the number of training data. Levenberg-Marquardt Algorithm is proposed for neural network training. After training the neural network and its training, the results are compared with other neural network and real values. The results showed that the optimal performance of the proposed method.

## 2. Serial Manipulator Kinematics

For serial manipulator, cartesian space $x$ is related to the joint space by

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

where $f(\cdot)$ is a nonlinear function. If Cartesian coordinates $x$ are known, then $q$ joint coordinates can be calculated as follows:

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

if linear velocity in Cartesian coordinate be shown by $V$, then joint velocity vector $\dot{q}$ can be obtained as:

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

Where $J$ is the jacobian matrix.

Also, if $V$ be desired Cartesian velocity, then Joint Velocity vector $\dot{q}$ given by:

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

In differential control of speed, desired trajectory has been divided into separate parts, by $\Delta t$ time interval. Assume that in time $t_i$, the joint position is $q(t_i)$, conventionally, required $q$ for manipulator in $(t_i + \Delta t)$ can be obtained as:

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

Substituting equations (2) and (4) into (5) yields:

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

Equation (6) is kinematics law that used for updating the joint position $q$ and evaluated in each sampling interval. The joint $q(t_i + \Delta t)$ for each joint is then sending to motor controller. Each of the joints can be controlled separately, so manipulator can easily react to following the trajectory favorably.

Homogeneous transformation matrix, describe the links translational and rotational features to adjacent links. In this method, each link of robot is modeled, and then this modeling describe homogeneous transformation matrix "A", that using four link parameters:

$$A_{End-Effector} = \begin{bmatrix} \text{rotation matrix} & | & \text{Position Vector} \\ ---- & | & ---- \\ \text{Perspective Transformation} & | & \textit{Scaling} \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad (7)$$

where $n$ is the normal vector of the hand, it is orthogonal to the fingers of the robot arm. $s$ is the sliding vector of the hand, it is pointing in the direction of the finger motion as the gripper opens and closes. $a$ is the approach vector of the hand, it is pointing in the direction normal to the palm of the hand (i.e., normal to the tool mounting plate of the arm). $p$ is the position vector of the hand, it points from the origin of the base coordinate system to the origin of the hand coordinate system, which is usually located at the center point of the fully closed fingers.

End-effector orientation of manipulator expressed in RPY orientation as follows:

$$RPY(\varphi_x, \varphi_y, \varphi_z) = Rot(Z_w, \varphi_z).Rot(Y_w, \varphi_y).Rot(X_w, \varphi_x) \qquad (8)$$

After $T_6$ matrix is solved:

$$\varphi_z = ATAN2(n_y.n_x) \qquad (9)$$

$$\varphi_y = ATAN2(-n_z.n_x \cos\varphi_z + n_y \sin\varphi_z) \qquad (10)$$

$$\varphi_x = ATAN2(a_x \sin\varphi_z - a_y \cos\varphi_z, o_y \cos\varphi_z - o_x \sin\varphi_z) \qquad (11)$$

These equations describe the end-effector orientation according to RPY presentation. To find the solution of inverse kinematics, the angle of each joint of the robot must be calculated according to the position of end-effector. Inverse kinematics solution can be found as follows:

$$IK\left(X,Y,Z,\varphi_x,\varphi_y,\varphi_z\right)=\left(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6\right) \qquad (12)$$

If the robot have complex dynamic, conventional methods prove insufficient to solve the inverse kinematics. Moreover, these methods are another problem that is the solution to select the correct answer from the answers obtained is not presented. Therefore, users often choose the best answer to his understanding of the subject [6]. Also, singularities in the robot problem have attracted the attention of researchers and many methods have been proposed to deal with this problem. Therefore, the analysis of the singularities of the robot and develop efficient algorithms for solving the inverse kinematics singularities or in its vicinity is very important.

## 3. Manipulator Kinematics

In this paper, FANUC M710i has been studied, as can be seen in Figure 2, this robot has 6 degrees of freedom (DOF). There are limitations on the speed and the angle of each joint in the robot. Therefore, it should be noted in the simulation. All joints in the robot are rotational. The robot simulated by using Matlab Robotic Toolbox [10]. For the simulation of the robot, the kinematics of the robot must be calculated. Therefore, it is necessary for robot to fill the Denavit-Hartenberg (D-H) table, after that we can obtain the kinematics of the manipulator [11]. D-H table of the robot is shown in Table 1.

**Table 1: D-H table for manipulator**

| $i$ | $\alpha$ | $a$ | $d$ | $\theta$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | $\theta_1$ |
| 2 | 90 | 8 | 0 | $\theta_2$ |
| 3 | 0 | 80 | 0 | $\theta_3$ |
| 4 | 90 | 21 | 0 | $\theta_4$ |
| 5 | -90 | 0 | 110 | $\theta_5$ |
| 6 | 90 | 0 | 0 | $\theta_6$ |
| End-Effector | 0 | 0 | 17.5 | 0 |

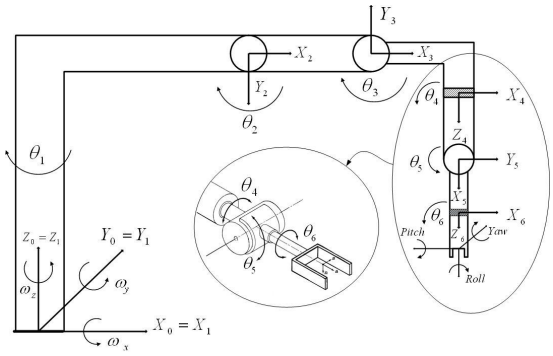


**Figure. 1: Schematic diagram of 6 DOF manipulator [6].**

Now we compute the singularities. After finding singularities, as far as possible, we designed smooth trajectory that is included the singularities of the robot.

For calculate the singularities, the robot jacobian matrix must be achieved, then, by putting jacobian matrix equal to zero, the singularities can be achieved. Jacobian is a $6\times6$ matrix; this matrix can be considered as follows:

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{13}$$

According to the obtained jacobian matrix, can be written:

$$Det(J) = Det(J_d) = Det(J_{11})Det(J_{22}) \tag{14}$$

$$Det(J_{11}) = 80 \times \sin q_3 \times (110 \sin q_4 - 21) \times f_{o.t} \tag{15}$$

$$Det(J_{22}) = -\sin q_5 \tag{16}$$

In the above equation, $f_{o.t}$ is the other terms that don't includes singularities, therefore, they have been ignored [12]. Due to physical constrains of each joint, the manipulator has 5 singularities. These singularities in radians, are: $q_3 = 0, \pm\pi$ , $q_4 = 0.19$ , $q_5 = 0$ .

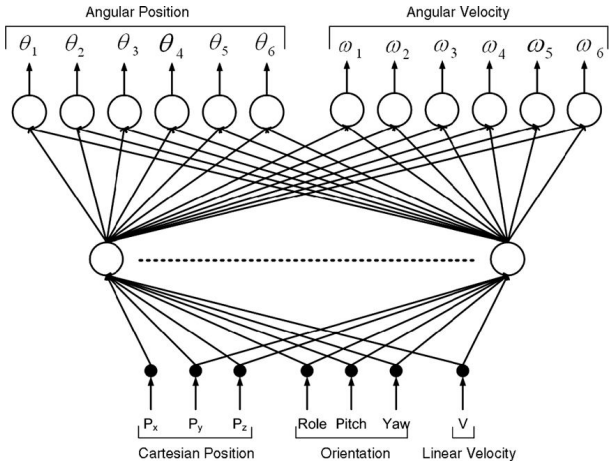


**Figure 2: The topology of the designed network [6].**

# 4. Artificial neural network design

## 4.1. Network topology

The fundamental idea underlying the design of a network is that the information entering the input layer is mapped as an internal representation in the units of the hidden layer(s) and the outputs are generated by this internal representation rather than by the input vector [6].

The inputs are Cartesian position of the end-effctor along the X, Y, Z axis and Roll, Pitch and Yaw (RPY) orientation and linear velocity of end-effector. Network outputs is defined the position and angular velocity of each joint.The proposed neural network is a network with hidden layer with 10 neurons in this layer. A tangent hyperbolic function selected as network activation function. this network topology is shown in Figure 2.

## 4.2. Levenberg-marquardt algorithms

The Levenberg-Marquardt method is a standard technique used to solve nonlinear least squares problems. Least squares problems arise when fitting a parameterized function to a set of measured data points by minimizing the sum of the squares of the errors between the data points and the function. Nonlinear least squares problems arise when the function is not linear in the parameters. Nonlinear least squares methods involve an iterative

improvement to parameter values in order to reduce the sum of the squares of the errors between the function and the measured data points.

The Levenberg-Marquardt curve-fitting method is actually a combination of two minimization methods: the gradient descent method and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the direction of the greatest reduction of the least squares objective. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic, and finding the minimum of the quadratic. The Levenberg-Marquardt method acts more like a gradient-descent method (GDM) when the parameters are far from their optimal value, and act more like the Gauss-Newton Algorithm (GNA) when the parameters are close to their optimal value.

The Levenberg-Marquardt algorithm (LMA) adaptively varies the parameter updates between the gradient descent and Gauss-Newton update,

$$[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\mathbf{I}]\delta = \mathbf{J}^T\mathbf{W}(\mathrm{y} - \hat{\mathrm{y}}) \qquad (17)$$

where $\mathbf{J}$ is the jacobian matrix whose $i$ th row equals $J_i$ and where $\mathrm{y}$ and $\hat{\mathrm{y}}$ are vectors . This is a set of linear equations which can be solved for $\delta$ [10].

Small values of algorithmic parameter $\lambda$ result in a Gauss-Newton update and large values of $\lambda$ result in a gradient descent update. At a large distance from the function minimum, the steepest descent method is utilized to provide steady and convergent progress toward solution. As the solution approaches the minimum $\lambda$ is adaptively decreased, the LMA approaches the Gauss-Newton Algorithm (GNA), and the solution typically converges rapidly to the local minimum [10]. Marquardt's suggested update relationship:

$$\left[\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\ \mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})\right]\delta = \mathbf{J}^T\mathbf{W}\left(\mathrm{y} - \hat{\mathrm{y}}\right) \qquad (18)$$

is used in the Levenberg-Marquardt algorithm.

### 4.3. Training process

Selection of training data for training artificial neural networks is discussed in this section. Several factors can cause restrictions in number of samples received from sensors for neural network training. Some of these problems are due to performance limitations of the sensor used in joints. Typically, the sensors are not the limiting factor at the time, because they have more speed than the speed of motors. These constrains determine the appropriate number of samples to train the neural network. Upper bound of the training data, where it is limited by constrains of the sensor or neural network overtraining.

Here, the minimum training data is determined by the physical limits of motors, sensors and complexity of the designed trajectory. Minimum accuracy requirements in degree are other factors should also be considered. Maximum motor speed and minimum resolution of angle are the basic criteria to select the number of data required for training the neural network.

Speed limits for each joint of the robot with 6DOF should be considered in trajectory design. In Table 2, the sampling time resolution of $1^{\circ}$ and $0.5^{\circ}$ and $0.1^{\circ}$ at each joint is shown. Sampling time is an interval of time that the motor of any joint, with attention to speed limits, needs to provide the desired rotation. Care must be taken that the error in any of the joints, causing errors in the End-effector.

The goal is to find the minimum sum of the square of the errors (SSE) by using the LMA. SSE is a suitable performance index. However, the accuracy of angle of any joint information can be selected. Since the maximum velocity of joints is close to each other, the maximum speed is selected for all the joints in common, so we have:

$$V_{\max} = \max\left\{V_j : j = 1,..,6\right\} \tag{19}$$

**Table 2: Relationship between angle accuracy and sampling time**

| Joint Number | Sample Time for 1 degree (sec) | Sample Time for 0.5 degree (sec) | Sample Time for 0.1 degree (sec) |
|---|---|---|---|
| 1 | $6.25\times10^{-3}$ | $3.125\times10^{-3}$ | $6.25\times10^{-4}$ |
| 2 | $8.3\times10^{-3}$ | $4.16\times10^{-3}$ | $8.3\times10^{-4}$ |
| 3 | $8.3\times10^{-3}$ | $4.16\times10^{-3}$ | $8.3\times10^{-4}$ |
| 4 | $4.4\times10^{-3}$ | $2.2\times10^{-3}$ | $4.4\times10^{-4}$ |
| 5 | $4.4\times10^{-3}$ | $2.2\times10^{-3}$ | $4.4\times10^{-4}$ |
| 6 | $4.4\times10^{-3}$ | $2.2\times10^{-3}$ | $4.4\times10^{-4}$ |

where $V_j$ is maximum angular velocity of each joint and $V_{\max}$ is the maximum velocity among all joints velocity.
Appropriate sampling time, can be computed as follows:

$$T_s = dq/V_{\max} \tag{20}$$

By knowing this is an offline approach, the design has already been done. Thus, the sampling time can be achieved $dq$ is the amount of accuracy required for angle of each joint by using the maximum speed that exists in the designed trajectory.

By calculating the forward kinematics, it can be seen that the end-effector position and orientation is a set of sinusoidal sentences. Suppose $q_i : i = 1,...,6$ is equal to the angle of each joint. In end-effector position and orientation equations, which is obtained by using the forward kinematics, In most cases, the fifth joint angle are involved in the position and orientation equation. In worst case, it is assumed that all joint angular velocity equal to the maximum possible velocity.

After converting the obtained forwad kinematics relations as the sum of the frequencies, the maximum frequency which the end-effector can move, calculated as:

$$\sin(q_1 + q_2 + q_3 + q_4 + q_5) \tag{21}$$

with maximum velocity and the relation $\Delta\theta/\Delta t = \omega$ and maximum possible velocity in designed trajectory for each joint selected as $\omega$ . For maximum robustness against errors, the relation (21) as modified as follows:

$$\sin\left(5q_{\max}\right) = \sin\left(\omega_{\max} t\right) \tag{22}$$

From equation (22) we obtain the value of $\omega_{\max}$ . By using the following equation:

$$T = \frac{2\pi}{\omega} \tag{23}$$

$T$ is obtained as the minimum sampling time for proper training the ANN. In this paper, the designed trajectory $\omega_{\max} = 7.305\ rad/s$ is obtained. Therefore, the maximum sampling time, to deal with most disturbances is obtained 10. Obtained sampling time, meaning that in the presence of maximum frequency joint motion and robot uncertainty, for this time of sampling, always the neural network is trained properly. If large amount of sampling time is selected, tracking is done with greater swings and weak performance.

## 5. Results and analysis

First, the desired trajectory designed, then by using forward kinematics, the end-effector position and orientation is obtained. We choose the sampling time 0.1 seconds. However, these data can be used to train the neural network.

In this section, two different neural network structures and optimization algorithms is used to find the inverse kinematics, and the results are compared with each other. Both types of ANN are feed forward neural network, ANN and all simulations was designed using Matlab software.

The first network has 55 neurons in the hidden layer and the first layer activation function is sigmoid function, and Gradient Descent Method (GDM) is used to train the ANN. In back propagation networks, the number of neurons in the hidden layer determines how well the network can learn. Constant learning rate of 0.9 was chosen. The difference between the desired output and the real output of the system can be chosen as the performance index of ANN, so that the neural network learning is done in a manner that internal structure can achieve a better response. Trajectory is designed in such a way that 600 training data is obtained for the robot every 1 second interval. From this set, 400 data used for the training process and the remaining 200 data were used for test process. This network is used in [6].

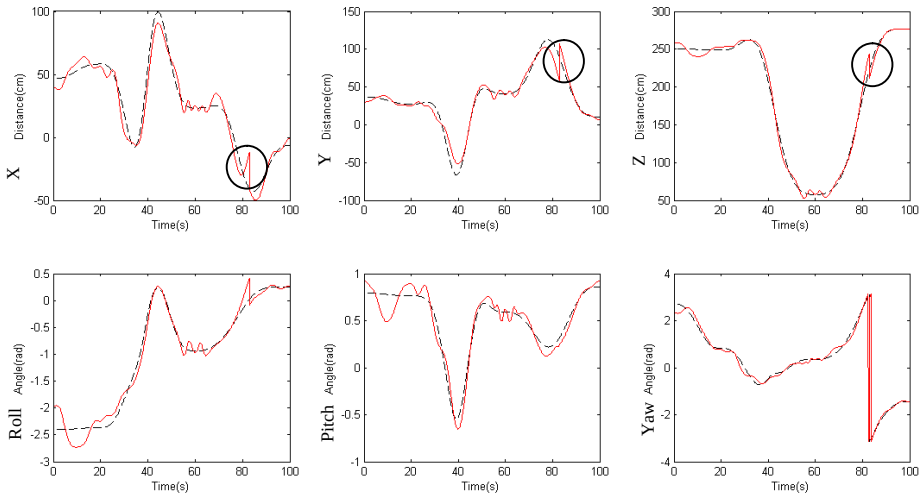


**Figure 3: End-effector position and orientation by using Gradient Descent method**

The neural network proposed in this paper, 10 neurons in the hidden layer is selected and activation function of hidden layer is hyperbolic tangent. The LMA is used to train the network. According to the calculations of the pervious section, 10 samples are collected every second. 60% of the data used for training, 20% for testing and 20% for performance verification is used.

In order to evaluate the ability to generalize and predict, in singularities and related uncertainty in robot model, after training, the new data that was not previously presented, is applied to network. Singularities are points where the jacobian matrix is zero. These points are shown in Figure 3 and Figure 4 by black circles.

The results obtained by using neural network that trained by GDM, is shown in Figure 3. The end-effector crossing singularities are marked with black circle. As can be seen, there are larger errors in singularities. The results obtained by using neural network trained by using LMA, is shown in Figure 4. End-effector crossing singularities are shown in this figure too.

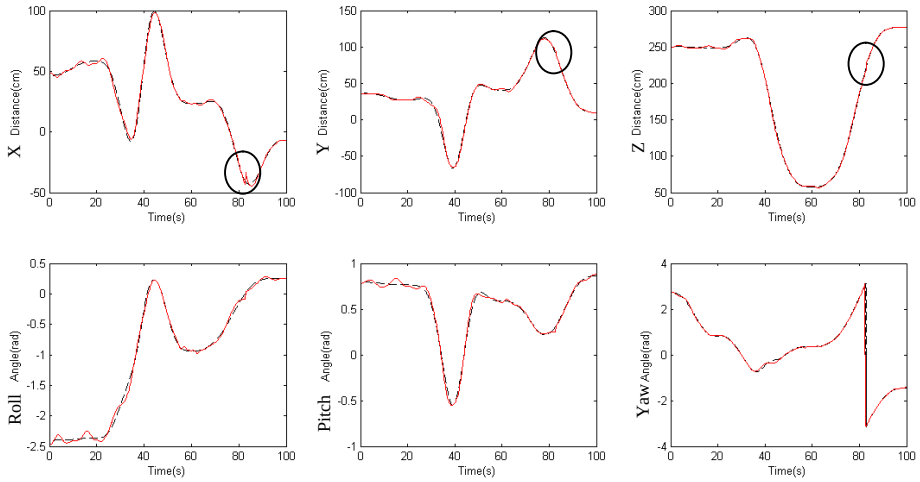


**Figure 4: End-effector position and orientation by using Levenberg-Marquardt Algorithm**

By comparing Figure 3 and 4, it can be seen, the LMA training has less error than the GDM training method. LMA have much fewer errors than networks trained by using GDM. LMA algorithm compared to GDM method reaches the optimal solution, with fewer epochs. Also it can be seen that the curve fitting by GDM has more swings in comparison training by LMA. In this compared algorithms, in all respects, LMA has better performance than the GDM.

## 6. Conslusion

Robot control as nonlinear system, have always been a favorite for control engineers. In this case does not require any prior knowledge of the kinematics model of the system being controlled. One of the proposed methods for controlling is the use of intelligent network. A configuration of the neural network was proposed and the algorithm was introduced to train the network. Maximum sampling time to produce training data by using features of the robot introduced. After training, we compared the performance of a neural network with a network by another configuration. Observed, the proposed neural network, training algorithm and training data has better performance in singularities and along the designed trajectory.

## References


[1]. Fatima Zahran Baghli, Larbi El Bakkali, "Artificial intelligence application's for a robot manipulator with two degrees of freedom position control," vol. 4(11), pp.349-368, 2014.

[2]. Hossien Nejatbakhsh Esfahani, Vahid Azimirad, "A new fuzzy sliding mode controller with PID sliding surface for underwater manipulators," vol. 3(9): pp. 224-249, 2013.

[3]. Hamze Ravaee, Saeid Farahat, Faramarz Sarhadd, "Artificial Neural Network Based Model of Photovoltaic Thermal (PVT) Collector", The journal of mathematics and computer science, vol. 4, pp. 412-417, 2012.

[4]. Hamed Azami, Milad Malekzadeh, Saeid Sanei , "A New Neural Network Approach for Face Recognition based on Conjugate Gradient Algorithms and Principal Component Analysis", The journal of mathematics and computer science, vol. 6, pp. 166-175, 2013.

[5]. Bekir Karlik and Serkan Aydin, "An improved approach to the solution of inverse kinematics problems for robot manipulators," Engineering Applications of Artificial Intelligence, vol. 13, pp. 159-164, 2000.

[6]. Ali T.Hasan, N.Ismail, "Artificial neural network-based kinematics jacobian solution for serial manipulator passing through singular configurations", Advances in Engineering Software, vol. 41, pp. 359-367, 2010.

[7]. Ali T.Hasan and H.M.Assadi, "Performance Prediction Network for Serial Manipulators Inverse Kinematics solution Passing through Singular Configurations", International Advances Robotic system, vol. 7, pp. 11-24, (2010).

[8]. Ali T.Hasan and H.M.Assadi, "An Improved Adaptive Kinematics Jacobian Trajectory Tracking of a Serial Robot Passing through Singular Configurations", Advanced Strategies for Robot manipulator, 2008.

[9]. Z.Bingul, H.M Ertunc and C.oysu, "Comparison of Inverse Kinematics Solutions Using Neural Network for 6R Robot Manipulator with Offset", ICSC Congress on Computational Intelligence Methods and applications, 2005.

[10]. Peter Corke, "Robotics, Vision and Control Fundamental Algorithms in Matlab," Springer, 2011.

[11]. John Craig, "Introduction to Robotics Mechanics and Control", Third Edition, Prentice Hall, 2004.

[12]. Denny Oetomo, Marcelo Ang Jr, Ser Yong Lim, "Singularity Handling on Puma in Operational Space Formulation," Lecture Notes in Control and Information Sciences, vol 271, pp. 491-500, 2001.


## Authors


**Mohammad Mahdi Azimi** is the PhD student in Control Engineering at Shahid Beheshti University, Tehran, Iran. He received the M.Sc. in Control Engineering from The University of Isfahan, Isfahan, Iran, in 2013. His current research interests are in nonlinear control, robust control, adaptive control, intelligent control and robotic systems.

**Payman Moallem** received his BSc in Electronic Engineering, Isfahan University of Technology, 1992, M.Sc. Electronic Engineering, Amir-Kabir University of Technology, 1996, and Ph.D. Electronic Engineering, Amir-Kabir University of Technology, 2003. His current research interests include Image Processing, Machine Vision, Neural Networks, Pattern Recognition, Real time Video Processing, DSP.