Advances in Engineering Software 41 (2010) 359–367

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Artificial neural network-based kinematics Jacobian solution for serial manipulator passing through singular configurations

Ali T. Hasan [a,*], N. Ismail [a], A.M.S. Hamouda [b], Ishak Aris [c], M.H. Marhaban [c], H.M.A.A. Al-Assadi [a]

[a] Department of Mechanical and Manufacturing Engineering, University Putra Malaysia, 43400UPM, Serdang, Selangor, Malaysia
[b] Department of Mechanical Engineering, Qatar University, Doha, Qatar
[c] Department of Electrical and Electronic Engineering, University Putra Malaysia, 43400UPM, Serdang, Selangor, Malaysia

ARTICLE INFO

Article history:
Received 18 February 2008
Accepted 28 June 2009
Available online 24 July 2009

Keywords:
Neural networks
Inverse kinematics
Jacobian matrix
Singularities
Back propagation
Robot control

ABSTRACT

Singularities and uncertainties in arm configurations are the main problems in kinematics robot control resulting from applying robot model, a solution based on using Artificial Neural Network (ANN) is proposed here. The main idea of this approach is the use of an ANN to learn the robot system characteristics rather than having to specify an explicit robot system model.

Despite the fact that this is very difficult in practice, training data were recorded experimentally from sensors fixed on each joint for a six Degrees of Freedom (DOF) industrial robot. The network was designed to have one hidden layer, where the input were the Cartesian positions along the *X*, *Y* and *Z* coordinates, the orientation according to the *RPY* representation and the linear velocity of the end-effector while the output were the angular position and velocities for each joint, In a free-of-obstacles workspace, off-line smooth geometric paths in the joint space of the manipulator are obtained.

The resulting network was tested for a new set of data that has never been introduced to the network before these data were recorded in the singular configurations, in order to show the generality and efficiency of the proposed approach, and then testing results were verified experimentally.

1. Introduction

For industrial applications, it is necessary to move the end-effector of a manipulator along some desired path with a prescribed speed. To achieve this goal, the motion of the individual joints of a manipulator must be carefully coordinated. Robot control usually requires control signals applied at the joints of the robot while the desired trajectory is specified for the end-effector. Then, it is essentially important for the controller to provide both positions and velocities transformation from Cartesian to joint space coordinate [1–3].

Kinematics control, and *dynamic control* are the two main areas of robot control problem [4]. Handling of torque limits naturally leads to control algorithms based on the dynamic model of the manipulator as, e.g., in [5–7]. A problem with these algorithms is the remarkable computational load required to handle the dynamics of a full-sized manipulator, which is seldom affordable by current industrial control units. In addition, implementation of torque-based control laws requires replacement of the low-level joint servos typically available in industrial robots with custom control loops. As a matter of fact, to our knowledge, on-line dynamic-based methods have indeed been tested in experiments only on laboratory setup with arms of a few *degrees of freedom* (DOF).

A different approach to path tracking aimed at overcoming the above drawbacks is based on the so-called kinematics control. In detail, kinematics control consists in an inverse kinematics transformation which sends to the joint servos the reference values corresponding to an assigned end-effector trajectory (both position and velocity); as a first advantage, this allows simple interfacing with the standard control architecture of industrial robots. In the framework of kinematics-based methods for path tracking, the counterpart of the physically meaning joint torque limits is played by acceleration constraints and the use of full dynamic models can be avoided; this typically leads to computationally light algorithms that allow real-time implementation on standard numerical hardware even for robot arms of many (DOFs). A further advantage of kinematics control methods is the possibility of exploiting the presence of redundant (DOFs).

It must be remarked that to achieve perfect path tracking, it is necessary to know the whole trajectory beforehand, which leads to off-line control techniques [8]. In the presence of uncertainty in kinematics, it is impossible to derive the desired joint angles from the desired Cartesian path by only solving the inverse kinematics problem [9]. On the other hand, the Jacobian matrix is a

* Corresponding author. Tel.: +60 3 89466330; fax: +60 3 89422107.
E-mail address: ali.taqi.hasan@gmail.com (A.T. Hasan).

0965-9978/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2009.06.006

critical component for generating trajectories of prescribed geometry in the joint space. Most coordination algorithms employed by industrial robots avoid numerical inversion of the Jacobian matrix by driving analytical inverse solution on an ad hoc basis. Therefore, it is important that efficient algorithms be developed [1].

Many research efforts have been devoted towards solving this problem, one of the first algorithms employed was the Resolved Motion Rate-Control method [10], which uses the pseudoinverse of the Jacobian matrix to obtain the joint velocities corresponding to a given end-effector velocity, an important drawback of this method was the singularity problem.

A velocity-singular configuration is a configuration in which a robot manipulator has lost at least one motion (DOF). In such configurations, the inverse Jacobian will not exist, and the joint velocities of the manipulator will become unacceptably large that often exceed the physical limits of joint actuators [11]. To overcome the problem of kinematics singularities, the use of a damped least squares inverse of the Jacobian matrix has been later proposed in lieu of the pseudoinverse [12,13].

Since in the above algorithmic methods the joint angles are obtained by numerical integration of the joint velocities, these and other related techniques suffer from errors due to both long-term numerical integration drift and incorrect initial joint angles. To alleviate the difficulty, algorithms based on the feedback error correction are introduced [14,15]. However, it is assumed that the exact model of manipulator Jacobian matrix of the mapping from joint coordinate to Cartesian coordinate is exactly known. It is also not sure to what extent the uncertainty could be allowed. Therefore, most research on robot control has assumed that the exact kinematics and Jacobian matrix of the manipulator from joint space to Cartesian space are known. This assumption leads to several open problems in the development of robot control laws today [8].

There have been an increasing research interest of Artificial Neural Networks (ANNs). In recent years, and many efforts have been made on applications of Neural Networks to various control problems [16–20]. ANNs, try to mirror the brain functions in a computerized way by resorting to the learning mechanism as the basis of human behavior. Utilizing the samples from the experiments, ANNs can be applied to the problems with no algorithmic solutions or with too complex algorithmic solutions to be found. Their ability of learning by examples makes the ANNs more flexible and powerful than the model based approaches [21].

An adaptive learning approach using ANN has been proposed here to control the motion of a 6DOF serial robot manipulator and to overcome the arising problems, which are mainly singularities and uncertainties in arm configurations. In this approach a network have been trained to learn desired set of angular positions and velocities from a given set of end-effector positions/orientations and velocities. Passing nearby and through singular configurations, data used were recorded experimentally from sensors fixed on each joint (as was recommended by Karilk and Aydin [4]), and training was done off-line until reaching acceptable error percentages finally training results were verified experimentally.

2. Overview of serial robot kinematics

In the programming of robot manipulators, a set of desired positions and orientations and their time derivatives, are specified in space, the problem is to find all possible sets of actuated joint variables and their corresponding time derivatives which will bring the end-effector to the set desired positions and orientations with the desired motion characteristics [1].

It is known that the vector of Cartesian space coordinates (the end-effector position and orientation) x of a robot manipulator is related to the joint coordinates q by:

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

where $f(\cdot)$ is a nonlinear differential function.

On the other hand, the mapping from the vector of joint velocities $\dot{q}$ to the vector of Cartesian space velocities $\dot{x}$ is also of interest, which is given by:

$$\dot{x} = J(q)\,\dot{q} \tag{2}$$

where $J(q)$ is the Jacobian matrix.

The inverse kinematics problem is defined as, given x the end-effector position and orientation vector, q the joint variable vector is to be found. More desirably given x and $\dot{x}$ to find q and $\dot{q}$.

In other words, the inverse kinematics problem is to solve the inverse problems [3]:

$$q = f^{-1}(x) \tag{3}$$

$$\dot{q} = J^{-1}(q)\,\dot{x} \tag{4}$$

In solving Eq. (3), Denavit and Hertenberg [22] proposed a matrix method of systematically establishing a coordinate system to each link of an articulated chain as shown in Fig. 1 to describe both translational and rotational relationships between adjacent links [18,23].

In this method each of the manipulator links is modelled, this modelling describes the "*A*" homogeneous transformation matrix, which uses four link parameters.

The forward kinematics solution can be obtained as:

$$\begin{aligned} A_{END-EFFECTOR} &= T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \\ &= \begin{bmatrix} \begin{matrix} Rotation \\ matrix \end{matrix} & | & \begin{matrix} Position \\ vector \end{matrix} \\ ---- & | & ---- \\ \begin{matrix} Perspective \\ transformation \end{matrix} & | & Scaling \end{bmatrix} \\ &= \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5}$$

where n is the normal vector of the hand. Assuming a parallel-jaw 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.

The orientation of the hand is described according to the *RPY* rotation as:

$$RPY(\phi_x, \phi_y, \phi_z) = Rot(Z_w, \phi_z) \cdot Rot(Y_w, \phi_y) \cdot Rot(X_w, \phi_x) \tag{6}$$

After T_6 matrix is solved:

$$\phi_z = ATAN2(n_y, n_x) \tag{7}$$
$$\phi_y = ATAN2(-n_z, n_x \cos\phi_z + n_y \sin\phi_z) \tag{8}$$
$$\phi_x = ATAN2(a_x \sin\phi_z - a_y \cos\phi_z, o_y \cos\phi_z - o_x \sin\phi_z) \tag{9}$$

These equations describe the orientation according to the *RPY* representation [4].

To find the inverse kinematics solution, however, joints angels are found according to the manipulator's end position, described with respect to the world coordinate system.

Inverse kinematics solution can be shown as a function:

$$IK(X, Y, Z, \phi_x, \phi_y, \phi_z) = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \tag{10}$$

Traditional methods for solving the inverse kinematics problem are inadequate if the structure of the robot is complex, besides; these

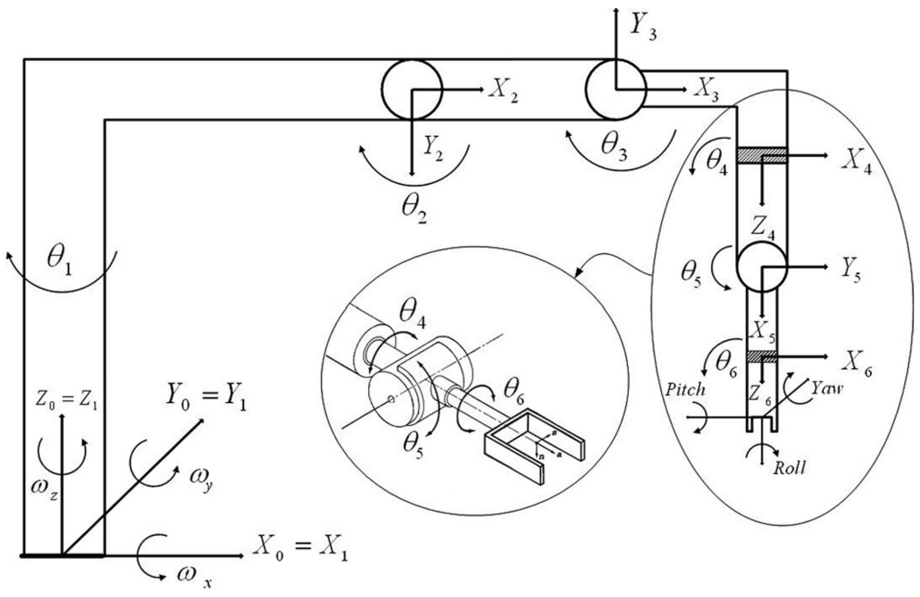

Fig. 1. Schematic diagram for a general 6DOF serial robot showing the wrist mechanism.

methods suffer from the fact that the solution does not give a clear indication on how to select an appropriate solution from the several possible solutions for a particular arm configuration, the user often needs to rely on his/her intuition to pick the right answer [19,23].

On the other hand, the manipulator singularity resolution problem has attracted many research interests, and various approaches have been proposed to tackle the problem.

Techniques of coping with kinematics singularities can be divided into four groups: avoiding singular configurations, robust inverses, a normal form approach and extended Jacobian techniques.

The first approach to copy with singularities is to keep a current configuration far away from singular configurations. Unfortunately, it causes severe restrictions on the configuration space, as well as the workspace, because the singular configurations split the configuration space into separate components. To avoid ill conditioning of the Jacobian matrix, robust inverses are used, Instead inverting the original Jacobian matrix at singularity; a disturbed, well-conditioned Jacobian matrix is inverted. The main drawback using this approach is that robust inverse methods increase errors in following a desired path.

The normal form technique, with the use of diffeomorphisms in joint and task spaces, expresses original kinematics around singularity in the simplest, normal form. Then, a piece of the path to follow, corresponding to the singular configuration mapped into the task space, is moved from the task to the joint space and trajectory planning is performed there. Far away from singularities the basic Newton algorithm is used to generate a trajectory. Finally, trajectory pieces are joined.

For most singularities the normal form approach enables to detect their types. It provides for a smooth passing through singular configurations. The main disadvantage of the normal form approach is a significant computational load in deriving the diffeomorphisms.

Finally, the extended Jacobian technique, supplements original kinematics with auxiliary functions. Then, extended Jacobian is formulated to be well-conditioned.

For nonredundant manipulators with square Jacobian matrices the extended Jacobian forms a non-square matrix and its generalized (Moore–Penrose) inversion is computationally expensive [24].

Therefore, to analyze the singular conditions of a manipulator and develop effective algorithms to resolve the inverse kinematics problem at or in the vicinity of singularities are of great importance.

3. Artificial neural networks – a brief description

A neural network is a massively parallel-distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the human brain in two respects; the network through a learning process acquires the knowledge, and interneuron connection strengths known as synaptic weights are used to store the knowledge.

An ANN is a group of interconnected artificial neurons interacting with one another in a concerted manner. The node receives weighted activation of other nodes through its incoming connections. First, these are added up (summation). The result is then passed through an activation function and the outcome is the activation of the node. The activation function can be a threshold function that passes information only if the combined activity level reaches a certain value, or it could be a continues function of the combined input, for this purpose, the most common to use is the sigmoid function as a nonlinear activation function.

However any input–output function that possesses a bounded derivative can be used in place of the sigmoid function. For each of the outgoing connections, this activation value is multiplied by the specific weight and transferred to the next node.

There are many learning rules can be used to train a network, the most used learning rule is the Generalized Delta learning Rule (GDR) which is most useful for multi layered networks training. A back propagation network trains with two-step procedure, the activity from the input pattern flows forward through the network and the error signal flows backwards to adjust the weights, where knowledge acquired by the network is stored as a set of connection weights [25].

4. Results

The solution of the kinematics Jacobian, which is mainly solved in this paper, involves the determination of the end-effectors position and orientation and their rate of change as a function of given positions and speed of the axes of motion.

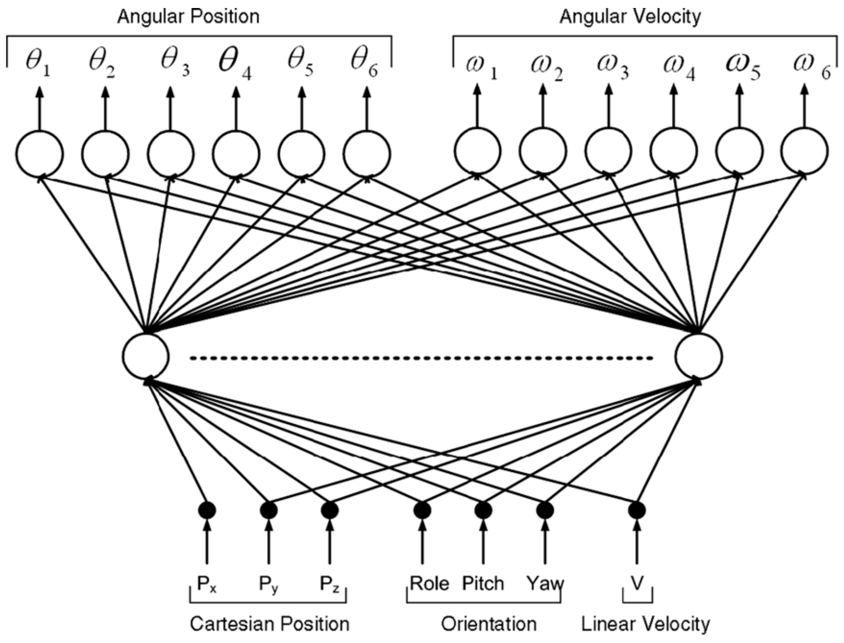
Angular Position
θ_1 θ_2 θ_3 θ_4 θ_5 θ_6
Angular Velocity
ω_1 ω_2 ω_3 ω_4 ω_5 ω_6
P_x P_y P_z
Cartesian Position
Role Pitch Yaw
Orientation
V
Linear Velocity

Fig. 2. The topology of the designed network.

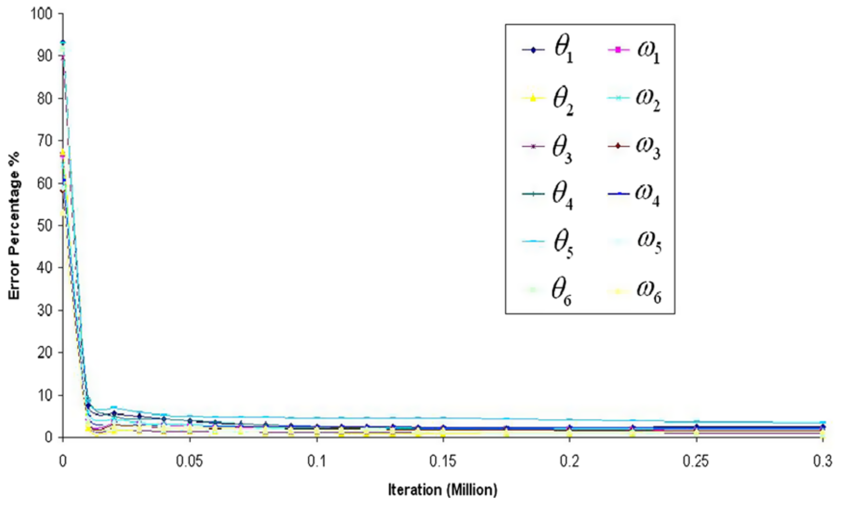

Fig. 3. The learning curve for the training set.

Table 1
Total error percentage for the training data set.

	Joint 1 (%)	Joint 2 (%)	Joint 3 (%)	Joint 4 (%)	Joint 5 (%)	Joint 6 (%)
Angular position (θ)	2.37	0.795	0.907	1.948	3.688	0.77
Angular velocity (ω)	1.683	2.02	1.525	2.19	1.375	1.313

A supervised feed forward ANN was designed using C programming language, the network consists of input, output and one hidden layer as can be seen in Fig. 2, every neuron in the network is fully connected with each other, sigmoid transfer function was used as an activation function, generalized backpropagation delta learning rule (GDR) algorithm was used in the training process.

The input vector for the network consists of the position of the end-effector of the robot along the *X*, *Y* and *Z* coordinates of the global coordinate system, the orientation according to the *RPY* representation and the linear velocity of the end-effector, while the output vector was the angular position and velocities of each of the six joints, respectively.

Studying the Inverse Kinematics of a serial manipulator by using ANNs has two problems, one of these is the selection of the appropriate type of network and the other is the generating of suitable training data set [26]. Different researchers have applied different methods for gathering training data, some of them have used the kinematics equations [4,27], some of them have used the network

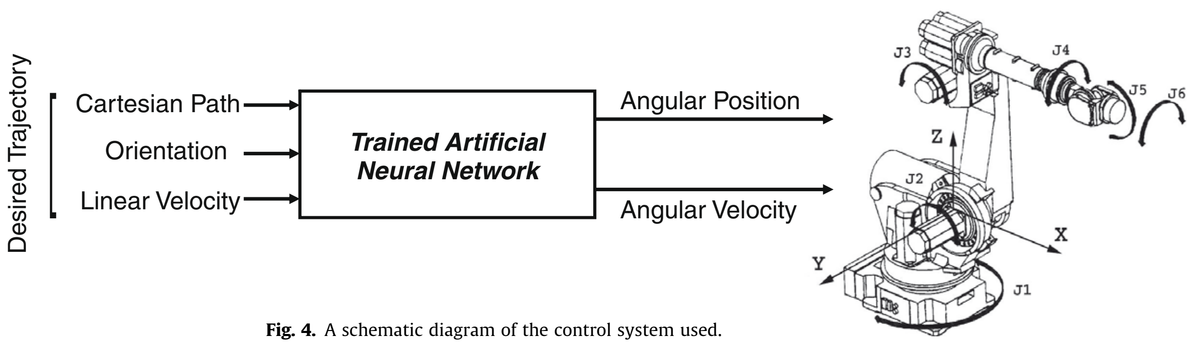

Fig. 4. A schematic diagram of the control system used.

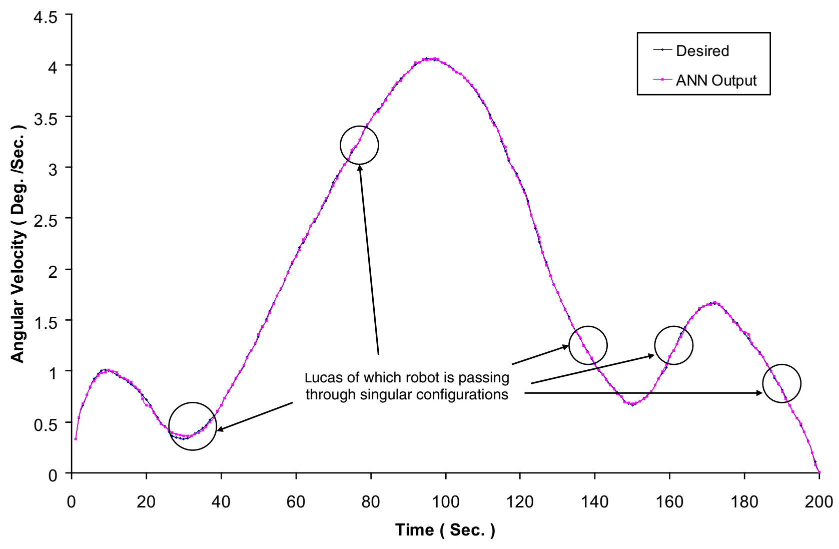

Fig. 5. Angular velocity of the fourth joint during testing process, showing the behavior of the network when passing nearby and through singular configurations.

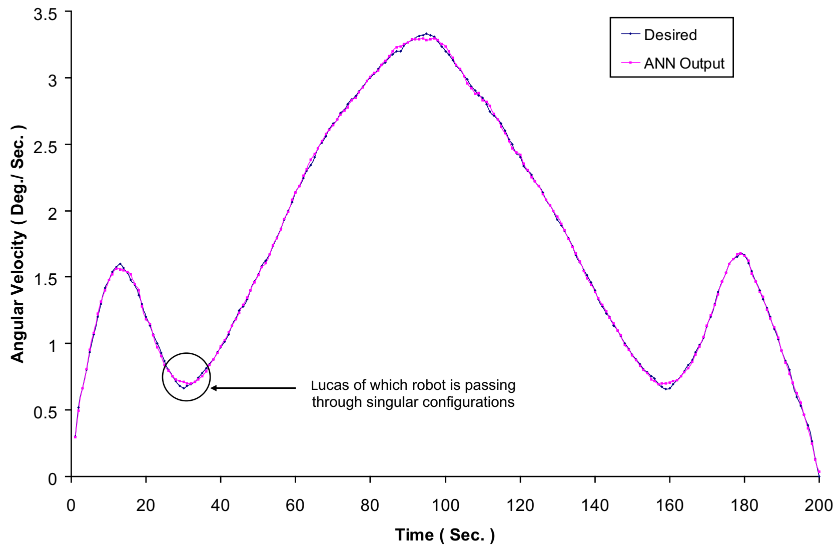

Fig. 6. Angular velocity of the fifth joint during testing process, showing the behavior of the network when passing nearby and through singular configurations.

inversion method [3,18], some of them have used the cubic trajectory planning [28] and other some have used a simulation program for this purpose [29]. However, there are always kinematics uncertainties presence in the real world such as ill-defined linkage parameters, links flexibility and backlashes in gear train, in this approach, although this is very difficult in practice [30], training data

Table 2
Total error percentage for the testing data set.

	Joint 1 (%)	Joint 2 (%)	Joint 3 (%)	Joint 4 (%)	Joint 5 (%)	Joint 6 (%)
Angular position (θ)	0.915	0.135	0.57	4.79	4.81	1.11
Angular velocity (ω)	1.265	2.02	1.205	1.41	1.15	1.175

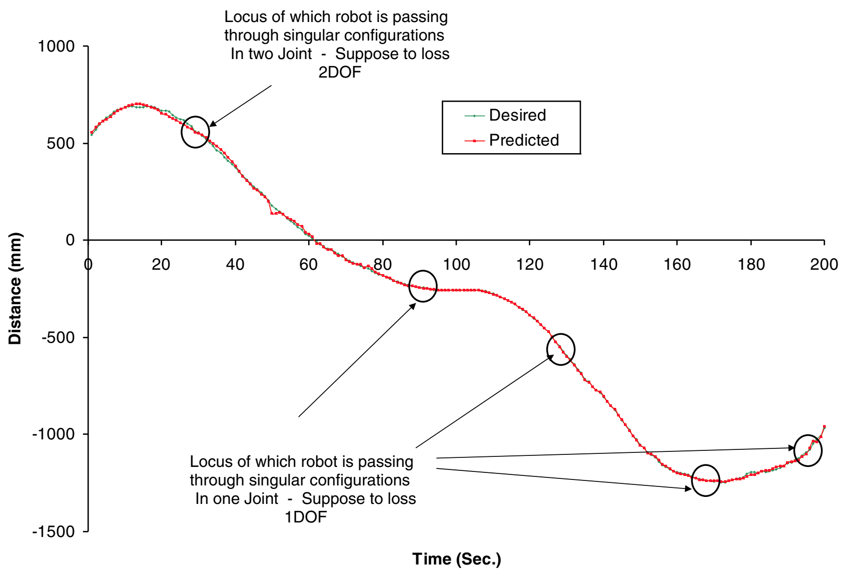

Fig. 7. Trajectory tracking for the *X* coordinate.

were recorded experimentally from sensors fixed on each joint (as was recommended by Karilk and Aydin [4]).

Trajectory planning was performed to get 600 data set for every 1-s interval for the FANUC M-710i robot; 400 sets of them were used in the training process while the remaining 200 were used for the testing process. All input and output values are usually scaled individually such that overall variance in the data set is maximized, this is necessary as it leads to faster learning, all the vectors were scaled to reflect continuous values ranges from -1 to 1.

4.1. Training process

In backpropagation networks, number of hidden neurons determines how well a problem can be learned. If too many are used, the network will tend to try to memories the problem and thus not generalize well later, if too few are used the network will generalize well but may not have enough power to learn the patterns well. Getting the right number of hidden neurons is a matter of trial and error, since there is no science to it, number of hidden neurons was set to be 55 by trial and error with a constant learning factor of 0.9.

The difference between desired and actual system output could be the performance measure for the network, as a learning system adapts its internal structure to achieve better response.

In the GDR the system is modified following each iteration, which leads to the learning curve shown in Fig. 3.

Table 1 shows the error percentages of each of the six joints after 0.25 million iterations.

4.2. Testing process

New data that has never been introduced to the network before have been fed to the trained network in order to test its ability to make prediction and generalization to any set of data later overcoming the singularity and uncertainty in the arm configuration resulting from applying the robot model, Fig. 4 shows a schematic diagram of the control system used.

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

Figs. 5 and 6 show the velocity tracking of the fourth and fifth joints, respectively showing the behavior of the network when passing nearby and through singular configurations during testing process.

Table 2 shows the percentages of error for the testing data set.

4.3. Experimental verification

In order to verify the testing results, experiment has been performed to make sure that the output is the same or sufficiently close to the desired trajectory, and to show the combined effect of error, Figs. 7–12 show the tracking of the Cartesian paths for the *X*, *Y*, and *Z* coordinates with the Roll, Pitch and Yaw orientation angles, respectively. The locus of which robot is passing through singular configurations are also shown.

The error percentages in the set are shown in Table 3.

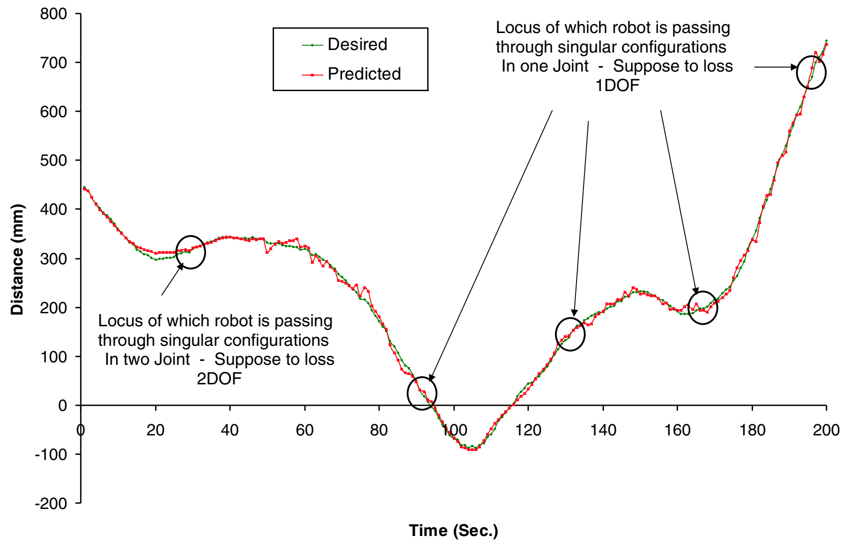

Fig. 8. Trajectory tracking for the *Y* coordinate.

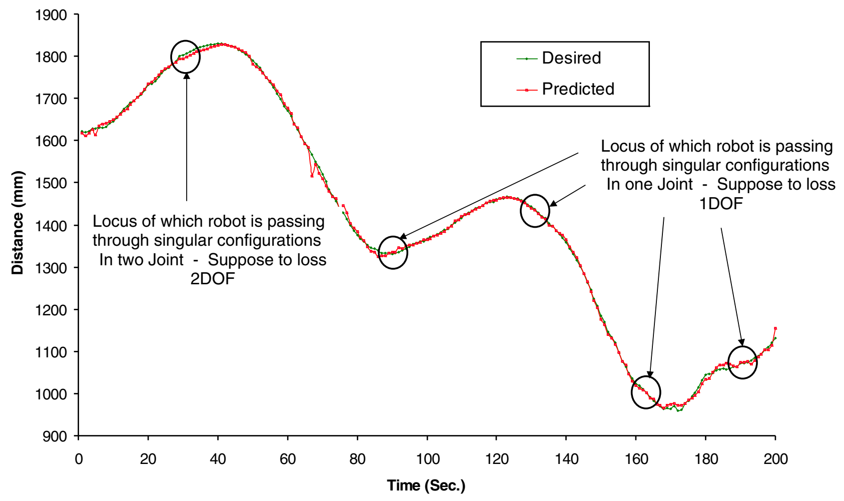

Fig. 9. Trajectory tracking for the *Z* coordinate.

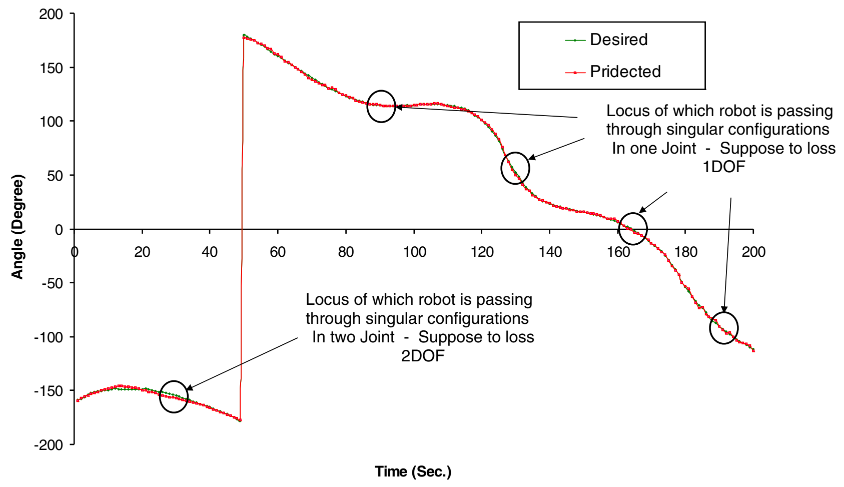

Fig. 10. Trajectory tracking for the Roll orientation angle.

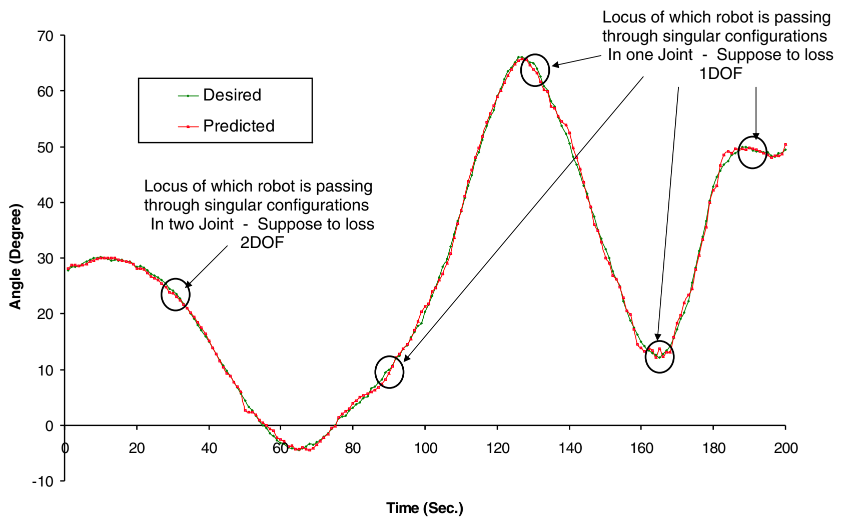

Fig. 11. Trajectory tracking for the Pitch orientation angle.

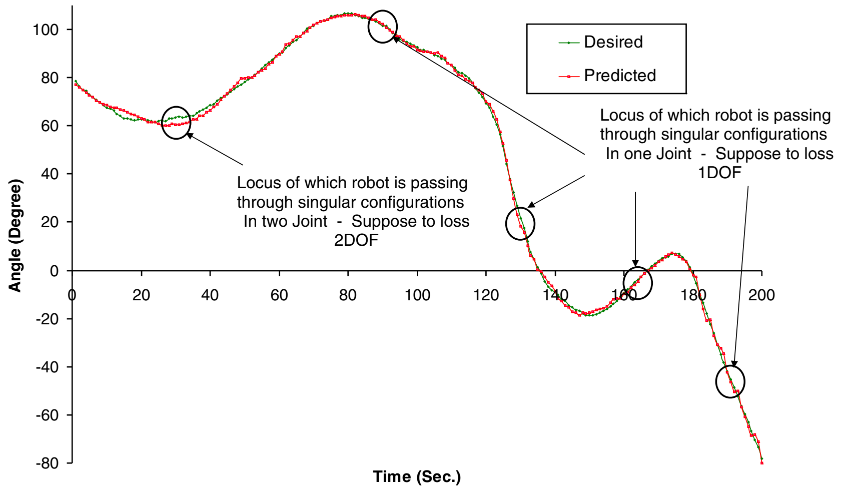

Fig. 12. Trajectory tracking for the Yaw orientation angle.

Table 3
Total error percentage for the experimental verification.

Cartesian position			Orientation		
P_x (%)	P_y (%)	P_z (%)	Roll (%)	Pitch (%)	Yaw (%)
3.34	6.72	0.35	2.55	5.54	5.79

5. Conclusions

Inverse Kinematics problem is very important to be solved in close-form for the robots using the Kinematics control approach, closed-form analytical solutions can only be found for manipulators having simple geometric structures. A number of algorithmic techniques mainly based on inversion of the mapping established between the joint space and the task space of the manipulator's Jacobian matrix have been proposed for those structures that cannot be solved in closed form.

In order to overcome the arising problems from applying the system kinematics model, which are mainly singularities and uncertainties in the arm configuration, Artificial Neural Network has been used trying to solve these problems. The proposed technique does not require any prior knowledge of the kinematics model of the system being controlled, the basic idea of this concept is the use of the ANN to learn the characteristics of the robot system rather than to specify explicit robot system model. Any modification in the physical set-up of the robot such as the addition of a new tool would only require training for a new trajectory without the need for any major system software modification, which is a significant advantage of using neural network approach.

This proposed approach possesses several distinct advantages: First, it can be applied to any general serial manipulator with positional degrees of freedom since learning is only based on observations of input/output relationships of the system being controlled, Second, reasonable accuracy can be achieved along the desired path, Third, the proposed approach can be adapted to any general serial manipulator including both redundant and non-redundant systems.

Since one of the most important issues in using ANNs is the selection of the appropriate type of network, for future research, we suggest that different types of networks (different topology, different activation function, different learning mode) to be used in order to get, if possible, more accurate trajectory tracking.

References

[1] Tsai LW. Robot analysis, the mechanics of serial and parallel manipulators. John Wiley & Sons, Inc.; 1999.

[2] Zurada JM. Introduction to artificial neural systems. Singapore: West Publishing Company; 1992.

[3] Kuroe Y, Nakai Y, Mori T. A new neural network learning on inverse kinematics of robot manipulators. In: International conference on neural networks, IEEE world congress on computational intelligence, vol. 5; 1994. p. 2819–24.

[4] Karilk B, Aydin S. An improved approach to the solution of inverse kinematics problems for robot manipulators. J Eng Appl Artif Int 2000;13:159–64.

[5] Dahl O, Nielsen L. Torque-limited path following by on line trajectory time scaling. IEEE Trans Robot Automat 1990;6:554–61.

[6] Kirćanski M, Kirćanski N. Resolved-rate and resolved-acceleration based robot control in the presence of actuators constraints, In: Proceedings of the 1997 IEEE international conference on robotics and automation, Albuquerque, NM; 1997. p. 235–40.

[7] Faiz N, Agrawal SK. Trajectory planning of robots with dynamics and inequalities. In: Proceedings of the IEEE international conference on robotics and automation, San Francisco, CA; 2000. p. 3977–83.

[8] Antonelli G, Chiaverini S, Fusco G. A new on-line algorithm for inverse kinematics of robot manipulators ensuring path-tracking capability under joint limits. IEEE Trans Robot Automat 2003;19(1):162–7.

[9] Cheah CC, Lee K, Kawamura S, Arimoto S. Asymptotic stability of robot control with approximate Jacobian matrix and its application to visual servoing. In: Proceedings of the 39th IEEE conference on decision and control, Sydney, Australia; 2000. p. 3939–44.

[10] Whitney E. Resolved motion rate control of manipulators and human prostheses. IEEE Trans Man–Mach Syst 1969;MMS–10:47–53.

[11] Hu Z, FU Z, Fang H. Study of singularity robust inverse of Jacobian matrix for manipulator. In: Proceedings of the first international conference on machine learning and cybernetics, Beijing; 2002. p. 406–10.

[12] Nakamura Y, Hanafusa H. Inverse kinematic solutions with singularity robustness for robot manipulator control. J Dyn Syst Measure Control 1986;108:163–71.

[13] Wampler CW. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. IEEE Trans Syst, Man, Cybern 1986;16:93–101.

[14] Balestrino A, De Maria G, Sciavicco L. Robust control of robotic manipulators. In: International proceedings of the 9th IFAC world congress, vol. 6, Budapest, Hungary; 1984. p. 80–5.

[15] Wampler CW, Leifer LJ. Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators. J Dyn Syst Measure Control 1988;110:31–8.

[16] D'Souza A, Vijayakumar S, Schaal S. Learning inverse kinematics. In: Proceedings of the 2001 IEEE/RSJ international conference on intelligent robots and systems Maui, Hawaii, USA; 2001. p. 298–303.

[17] Ogawa T, Matsuura H, Kanada H. A Solution of inverse kinematics of robot arm using network inversion. In: Proceedings of the 2005 international conference on computational intelligence for modelling, control and automation.

[18] Köker R. Reliability-based approach to the inverse kinematics solution of robots using Elman's networks. J Eng Appl Artif Int 2005;18:685–93.

[19] Hasan AT, Hamouda AMS, Ismail N, Al-Assadi HMAA. An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F. serial robot manipulator. J Adv Eng Software 2006;37:432–8.

[20] Al-Assadi HMAA, Hamouda AMS, Ismail N, Aris I. An adaptive learning algorithm for controlling a two-degree-of-freedom serial ball-and-socket actuator. Proc I Mech E Part I J Syst Control Eng 2007;221(7):1001–6.

[21] Hagan MT, Demuth HB, Beale M. Neural network design. Boston: PWS Publishing Company; 1995.

[22] Denavit J, Hertenberg RSA. Kinematics notation for lower pair mechanism based on matrices. Appl Mech 1955;77:215–21.

[23] Fu KS, Gonzalez RC, Lee CSG. Robotics control, vision, and intelligence. Singapore: McGraw-Hill book Co.; 1987. International edition.

[24] Dulęba I, Sasiadek JZ. Modified Jacobian method of transversal passing through singularities of nonredundant manipulators. In: Proceedings of the American control conference Chicago, Illinois, June 2000. p. 2839–43.

[25] Hasan AT, Hamouda AMS, Ismail N, Al-Assadi HMAA. A new adaptive learning algorithm for robot manipulator control. Proc I Mech E, Part I: J Syst Control Eng 2007;221(4):663–72.

[26] Funahashi KI. On the approximate realization of continuous mapping by neural networks. Neural Networks 1998;2(3):183–92.

[27] Bingual Z, Ertunc HM, Oysu C. Comparison of inverse kinematics solutions using neural network for 6R robot manipulator with offset. In: 2005 ICSC congress on computational intelligence.

[28] Köker R, Öz C, Çakar T, Ekiz H. A study of neural network based inverse kinematics solution for a three-joint robot. Robot Autonom Syst 2004;49:227–34.

[29] Driscoll JA. Comparison of neural network architectures for the modeling of robot inverse kinematics. In: Proceedings of the IEEE, south aston; 2000. p. 44–51.

[30] Hornik K. Approximation capabilities of multi-layer feed forward networks. IEEE Trans Neural Network 1991;4(2):251–7.