International Journal of Control and Automation
Vol. 4, No. 4, December, 2011

Neural Network Based Method for Estimation of Robot Trajectory Control Parameters

Shital S. Chiddarwar[1] and N. Ramesh Babu[2]

[1] *Shri Ramdeobaba Kamla Nehru Engineering College, Nagpur, India*
[2] *Indian Institute of Technology Madras, Chennai, India*
s.chiddarwar@hotmail.com, nrbabu@iitm.ac.in

Abstract

This paper explains about a neural network based control scheme to determine trajectory control parameters i.e. velocity, acceleration and jerk for a industrial robot performing a defined task. The proposed approach is the blend of quintic B-spline and neural network system for prediction of joint control parameters. The proposed approach is applied to a case of three robot trajectory control problem. The same problem is solved using a RobotiCad toolbox of matlab. The performance of the proposed approach is compared with the output of toolbox based on the total time and the torque consumed for the completion of the desired task.

Keywords: *Trajectory tracking, radial basis function, quintic B-spline, neural networks, algebraic spline*

1. Introduction

Industrial robots are being primarily used in industrial applications like spot welding, assembly, material handling and spray painting due to their great ability of speed and precision and their cost-effectiveness in repetitive tasks. However, these powerful machines are hardly autonomous in the sense that they require preliminary actions, such as calibration or motion planning to achieve specified tasks. The control of such a machine using computer software makes it both flexible in the way it works and versatile in the variety of tasks it can achieve. However, the full benefits of a robot can be obtained only if the programming phase is well accomplished. In general, the process of programming industrial robots is divided into two major levels: trajectory planning and trajectory control. The first consists of determining the time history of the robot kinematic parameters (position, velocity, acceleration) that correspond to the desired task. The second consists of converting these computed parameters into control signals in order to command the actuators during the execution of the task and to compensate for any deviation.

The establishment of the robot kinematics essentially contends with mapping between vectors in two spaces namely joint space and Cartesian space. It is well acknowledged that the relationship between Cartesian space position vector and joint space position vector is a non-differentiable mapping. Moreover, the velocity vector of the task space coordinates is related to that of the joint coordinates by the Jacobian matrix of the mapping. However, in order to achieve proper control of the robot, it is essentially significant to design the controller with propensity to provide transformation of both the positions and velocities from Cartesian space to the joint space coordinate [1].

The robot path tracking or control problem can be divided into two main areas: kinematics control (the coordination of the links of kinematics chain to produce desire motions of the robot) and dynamic control (driving the actuators of the mechanism to follow the commanded position/velocities) [2]. In order to achieve the dynamic control of the robot, there is a need to develop a control algorithm based on dynamics of the robots, which is capable of handling torque limits as well as other parameters like acceleration and jerk limits. The task of handling dynamics of a full sized industrial robot is computationally intensive, which is hardly ever affordable by current industrial control units. Moreover, implementation of torque based control laws requires replacements of typically available low-level joint servos with custom control loops. As a matter of fact, online dynamic based methods have indeed been tested in experiments only in a laboratory setup with arms of a few degrees of freedom (DOFs) [3].

These shortcomings coupled with dynamic control can be surmounted by adopting kinematic control or path tracking approach. Kinematics control consists of transformation that sends joint angle values corresponding to an assigned end-effector trajectory (both position and velocity) to the joint servos. This facet of kinematics control demonstrates simple interfacing with the standard control architecture of an industrial robot. In case of kinematic control, the role of torque constraints can be played by acceleration constraints and hence use of full dynamics model is evaded. This obviously leads to computationally simple algorithm that permits real-time accomplishment of standard numerical hardware even for robots with multiple DOFs. Further advantage of kinematic control is the possibility of exploiting the presence of redundant DOFs [4].

Numerous research efforts are reported in the literature in order to resolve the issue of kinematic control while achieving appropriate path tracking. One of the preliminary methods employed was the motion rate-control approach which uses pseudo-inverse of the Jacobian matrix in order to determine the joint velocities corresponding to a given end-effector velocity [5]. This method suffers from the basic disadvantage of perseverance of singularity condition.

A velocity singular configuration is the configuration of the robot at which a robot manipulator will lose at least one degree of freedom. If robot moves through such configuration, the inverse of the Jacobian matrix will not exist and joint velocities of the manipulator will become inaptly large and surpass the physical limits of the joint actuators [6]. To overcome this problem of velocity singularities, use of a damped least squares inverse of Jacobian matrix was proposed in order to substitute pseudo-inverse of Jacobian [7, 8]. Since in these methods, joint angles are obtained by numerical integration of joint velocities, these methods have basic disadvantage of integration drift.

Therefore, most research on robot control has assumed that the accurate kinematics of the manipulator are known precisely. This supposition leads to several open problems in the advancement of robot control laws. The in general intricacy of the robot control problem and pursuit for a strictly autonomous robot system have caused substantial interest to be dedicated to the application of the neural network technology to robot control.

Kuroe et al (1994) proposed a learning method of a neural network such that the network signifies the relations of both the positions and velocities from Cartesian coordinate to the joint coordinates [9]. They have derived a learning algorithm for arbitrary connected recurrent neural networks for the original neural networks. On-line training was performed for a two DOF robot. It was essentially an online learning process whereas Graca and Gu (1993) [10] developed a fuzzy learning algorithm. Based on the robotic differential motion process, the Jacobian matrix was treated as a fuzzy matrix and was learned through the fuzzy regression process. It was noteworthy that the fuzzy learning control algorithm neither requires an exact kinematics model of the robotic manipulator nor a fuzzy inference engine, as it is typically

done by a conventional fuzzy control. This is regardless of the fact that, unlike most learning control algorithms, multiple trials are not necessary for the robot to learn desired trajectory. A major drawback was that it only remembers the most recently introduced data points. Hence, researchers have recommended neural network so that it will remember the trajectory itself and not the most recently visited data points.

Koker et al (2004) used three-joint robotic manipulator simulation software [11]. First, they generated initial and final points in the work volume of the robotic manipulator by using cubic trajectory. Then all the angles according to the Cartesian coordinates were recorded and the neural network was trained online until the error was acceptable.

Recently, a trajectory tracking method based on adaptive kinematics Jacobian method is proposed by [12]. They have made use of feed forward neural network for predicting joint angles and angular velocities at the desired Cartesian configurations. The data required for training of the neural network was obtained from the sensors attached to each joint of robot.

Studying trajectory tracking of a serial robot by using artificial neural network has two problems. One is the selection of appropriate network type and other is the generation of suitable training data set [13, 14]. Researchers have applied different methods for gathering training data like kinematics equations of the robot [3], generation of numerous trajectories out of the available Cartesian configurations using cubic splines [11], network inversion method [9] and data collection from the sensors attached to robot joints [12].

From literature review presented above, it can be observed that most of the researchers have limited the resolution of kinematic control problem to the determination of joint angle and angular velocity. None has reported the determination of all the kinematic control parameters like angular velocity, angular acceleration and jerk, which are of prime importance whilst achieving desired path tracking. Since, the earlier approaches have avoided dynamics of the robot while estimating trajectory tracking parameters; there must be some assurance about the safe and jerk free movement of the robot with the predicted joint angle velocities. This aspect of limiting the jerk is very important since high jerk values can wear out the elements of robot structure and heavily excite its resonance frequencies. In addition, vibrations induced by non-smooth trajectories can damage robot actuators, and introduce large errors while robot is performing tasks such as trajectory tracking. Moreover, low-jerk trajectories can be executed more rapidly and accurately. The importance of generating trajectories that do not require abrupt torque variations has already been remarked.

With this point of view, in this work an attempt is made to find out the angular velocity, acceleration and jerk provided that the joint angle configurations are known in advance. The data required for execution of this approach necessarily demands for joint angle configurations and time required to reach to that particular joint angle configuration. Hence, the proposed approach assumes that the problem of inverse kinematics is already solved and the desired trajectory is obtained in the form of joint angles. The proposed approach makes use of an artificial neural network to predict the angular velocity, angular acceleration and jerk at desired joint angles. The basic problems associated with determination of kinematic control parameters for a robot are the selection of appropriate network type and other is the generation of suitable training data set. Among the existing networks, BPANN as well as perceptron neural network are time intensive due to the requirements of more number of epoch (iterations) for training of network [16]. On the contrary, RBF neural network shows faster convergence rate and high accuracy due to its ability of local approximation [17]. Moreover, RBF can handle large database very effectively and converges quickly which makes it efficient to predict the solutions within short period. Hence, in this work radial basis function neural network is utilized.

Further, in order to resolve the issue of network selection, two neural network architectures are devised which makes use of joint angle configurations and time as input parameters and predict either kinematic control parameters for each joint independently or for all the joints together. The task of predicting all the kinematics control parameter attempted in this work, needs the information vis-à-vis joint angle velocity, acceleration and jerk at particular joint angle configuration. Hence, the use of kinematics equations or online generation of data by attaching sensors is not viable. Moreover, while using spline fitting, a spline that can offer higher order continuity is mandatory so as to ensure continuous values of the derivatives of trajectory up to third derivative. For this purpose, either fourth order trigonometric spline [18] or higher order B-spline [19] can be used. The use of fourth order trigonometric spline helps to get the best position function. However, the superiority of the trigonometric splines in the position function is paid by having significantly higher values in the acceleration and jerk functions. For trigonometric splines, the procedure to determine the values of the position derivatives at the knots has to be applied, considering the spline times as equal to each other‟s and therefore disregarding their actual values. When spline times are not related to the intermediate points of the trajectory, then it might be worth employing the trigonometric splines, taking into account that they give smaller values of overshoots but also bigger values of the acceleration and jerk functions. However, for the case of robot moving through via-points and objective is to determine the angular velocity, acceleration and jerk, the spline time also plays an important role [20]. Hence, in this work the data required for training of the neural network is determined with the help of fifth order (quintic) B-spline which gives continuous derivatives up to third order.

The scope of this work is limited to computation of angular velocity, angular acceleration and jerk at the various joint angle coordinates obtained for various poses of robot using suitable inverse kinematics technique. The proposed ANN based trajectory tracking algorithm along with the strategies adopted for data generation and formulation of ANN is described in the next section. The results obtained with the proposed approach, the conclusions derived along with future scope of the approach are presented in the subsequent sections.

2. Proposed Approach for Trajectory Tracking

Figure 1 presents the proposed RBF ANN based approach to compute trajectory tracking/ kinematic control parameters i.e. angular velocity, angular acceleration and jerk at various joint angle configurations. The accuracy of prediction of neural network approaches mainly depends on the accuracy of training data and the configuration of the neural network architecture used for training and prediction. In view of these aspects, the proposed approach consists of two important phases a) generation of data for training of neural network and b) selection of appropriate neural network architecture. In order to generate the requisite training data, the temporal information about the movement of robot in the joint space is necessary. For obtaining joint angle coordinates (0), inverse kinematics solutions are obtained using fusion technique [21]. The temporal information (t) is obtained using the relationships jacobian relationship. The database thus obtained in the form of 0-t is further utilized to obtain the requisite training data i.e. angular velocity, angular acceleration and jerk. For this purpose, a fifth order B-spline is fitted to the database which in the form of 0-t and the first, second and third derivative of these splines are obtained. In order to select an appropriate network configuration, two types of architectures are used and their performance is compared. The details of the methodology developed for generation of training data are given in the next section.

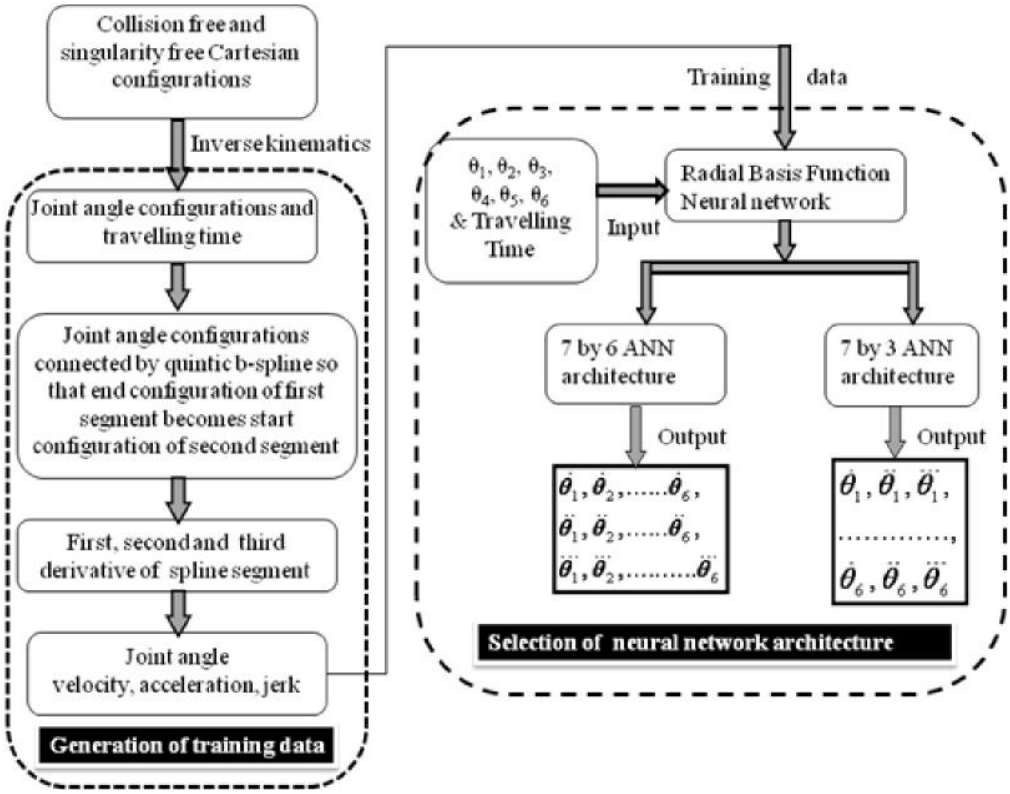


Figure 1. Schematic representation of the proposed RBF ANN based approach for estimation of trajectory tracking parameters

2.1 Generation of Training Data Using Quintic B-splines

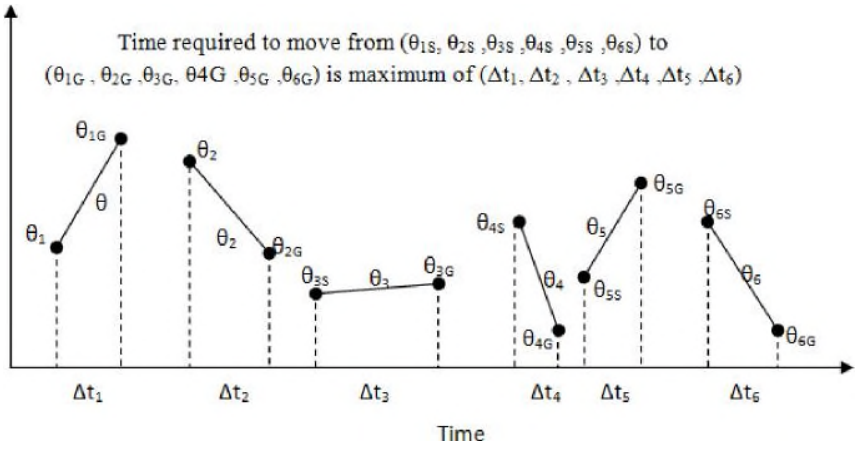


Figure 2. Computation of Time Required for Robot to move from Start Configuration to Goal Configuration

The proposed approach essentially demands for generation of accurate training data. In this approach, the data required for training of the neural network architectures is obtained by

fitting a quintic B-spline to the joint angle configurations and taking their derivatives. This data set is determined by fitting quintic B-splines to the selected configurations of robot joint space. These locations are selected in such a way that they obey the kinematic constraints of robot and are non-singular. From the entire work volume of Kuka-Kr 15 robot [25], 591 such configurations were selected and their joint space coordinates i.e. $\{0_1, 0_2, 0_3, 0_4, 0_5, 0_6\}$ were obtained using RBF ANN based fusion approach [21]. The time required for robot to travel from one configuration to consecutive configuration was computed by using Jacobian matrix and linear velocity at each joint of robot. The optimal time required for travelling from one configuration to the next configuration is determined by using strategy given in Fig. 2. Suppose robot has to move from joint configuration $(0_{1S}, 0_{2S}, 0_{3S}, 0_{4S}, 0_{5S}, 0_{6S})$ ^{to} $(0_{1G}, 0_{2G}, 0_{3G}, 0_{4G}, 0_{5G}, 0_{6G})$. The time required for six joints to move from their respective start configuration to goal configuration is Δt_1, Δt_2, Δt_3, Δt_4, Δt_5 and Δt_6 respectively. Since all the joints of robot are moving simultaneously, the actual time required for robot to move from its start position to the goal position is the maximum of $(\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4, \Delta t_5, \Delta t_6)$. In this way, the temporal information about the selected 591 joint angle configurations is determined. A fifth order B-spline is fitted to this database obtained in the form of 0-t in a sequential way i.e. end configuration of first segment becomes start configuration of the second segment. As shown in Fig. 3, the goal configuration G_1 of segment 1 becomes the start configuration S_2 for segment 2. In this way, all 591 configurations are connected and fifth order B-spline is fitted to obtain 590 spline segments. The procedure for spline fitting is adopted from Rogers and Adams, 2002. In this way, the values of angular velocity, angular acceleration and angular jerk at each joint angle configuration are obtained. The joint angle configuration and time $\{0_1, 0_2, 0_3, 0_4, 0_5, 0_6, Time\}$ is used as an input data to the neural network and the corresponding joint velocity, acceleration and jerk become the output from the neural network. The neural network architectures designed to predict trajectory control parameters are explained in the next section.

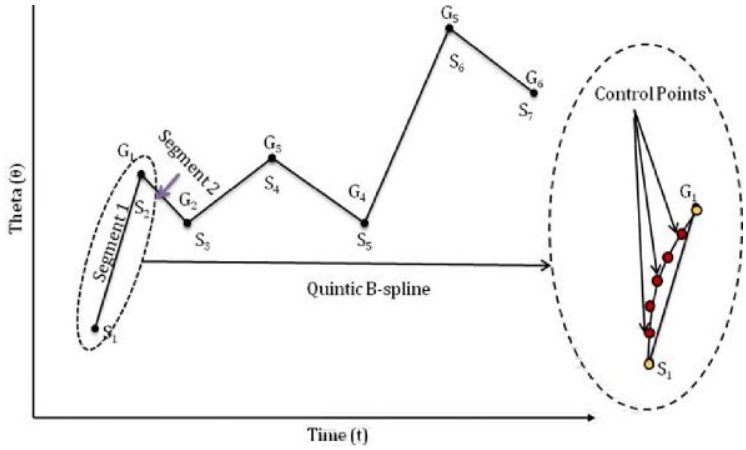


Figure 3 Generation of training data by fitting quintic B-spline to O-t database

2.2 Selection of Neural Network Architectures

The performance of the neural network system depends on the architecture used for training and prediction [3] (Karlik and Aydin, 2000). In order to determine appropriate neural network architecture for the accurate prediction of kinematic control parameters, two types of

architectures are developed. The first architecture is designed to consider {$\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$, *Time}* as input and predict all the trajectory tracking parameters for one joint of robot at a time. The second architecture is designed to predict one trajectory tracking parameter for all the six joints of robot at a time. The first neural network architecture has 7 inputs and 6 outputs, and will be referred as 7by6 network. This architecture predicts the values of various kinematic control parameters for all the six joints of robot i.e. angular velocity (Fig.4), angular acceleration (Fig. 5) and jerk (Fig. 6). For prediction of all the kinematic control parameters for all the six joints of a robot, totally three 7 by 6 networks are required. Fig. 7 shows the second type of the architecture where the inputs to the network are {$\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$, *Time*} and the output from the neural network is for each joint separately. Hereafter it will be referred as 7 by 3 network. With the second type of architecture, totally six 7 by 3 networks are required for prediction of the kinematic control parameters for all six joints of robot. The problem of estimating the kinematic control parameters for a robot at any given joint coordinate is based on the joint angle coordinates (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6) and time. This entails finding an approximating function for prediction of kinematic control parameters.

$$\sum_{j=1}^{M} w_j \phi_j(js) + b_o \qquad (2)$$

$\phi_j(js) = \exp\left[-\frac{\|js - c_j\|^2}{\sigma_j^2}\right]$

$\phi_j(js)$ is the output of j *th* node in hidden layer, c_j is the center of j *th* RBF node for the k *th* input variable, σ_j is the width of Gaussian function

w_j

bo is the bias term

M is the number of hidden layer neurons

is the weight between *j th*RBF unit and output layer neuron

The approximating function $KC(js)$ representing trajectory in joint space is an N_c dimensional vector in this case the components of is $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$ are independent variables. In the RBF network $KC(js)$ can be expressed as a linear combination of univariate Gaussian basis function as shown in the equation (1) and (2). The training of RBF ANN architectures shown in Figs. 4 – 7 is done by using the training data generated by means of the approach explained in the previous section. The performance of these two types of architectures is compared by means of % error in the prediction of kinematic control parameters. The details of this comparison are given in the next section.

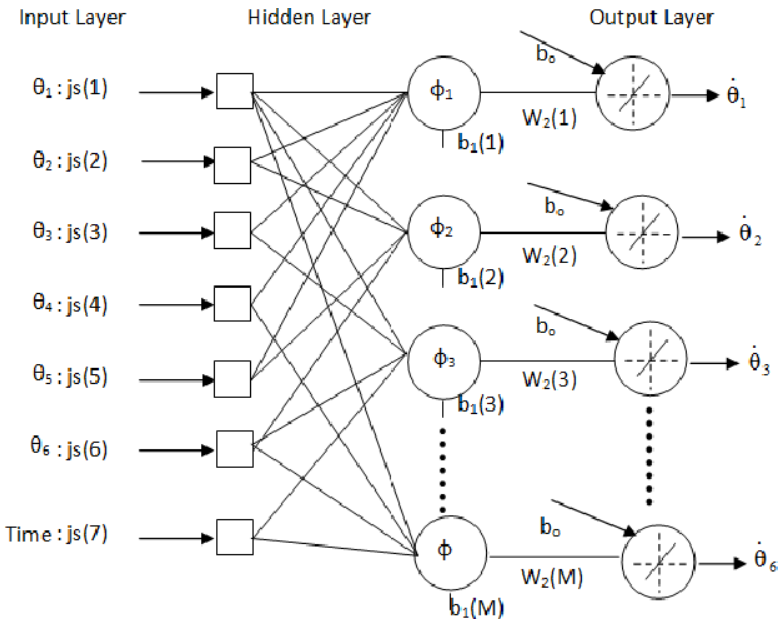


Figure 4. Architecture of RBF neural network used for prediction of angular velocity of six joints of robot (7by6 network)

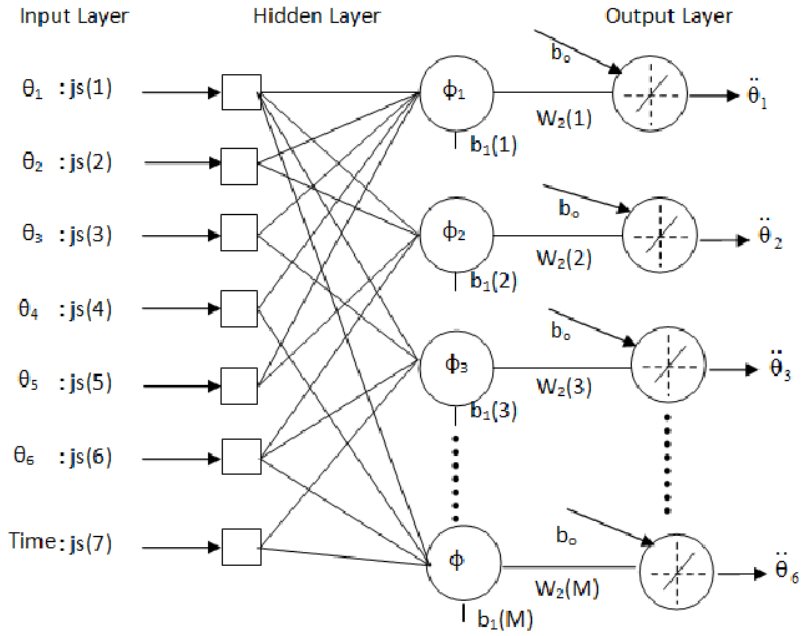


Figure 5. Architecture of RBF neural network used for prediction of angular acceleration of six joints of robot (7by6 network)

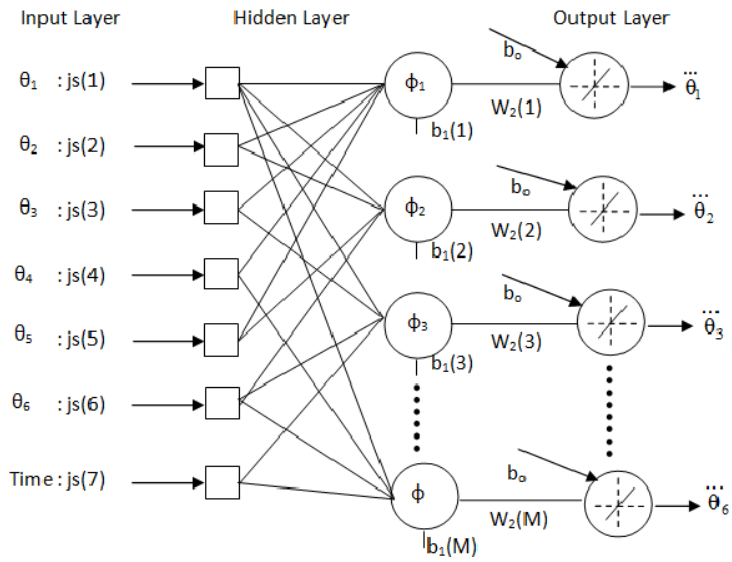


Figure 6. Architecture of RBF neural network used for prediction of jerk at six joints of robot (7by6 network)

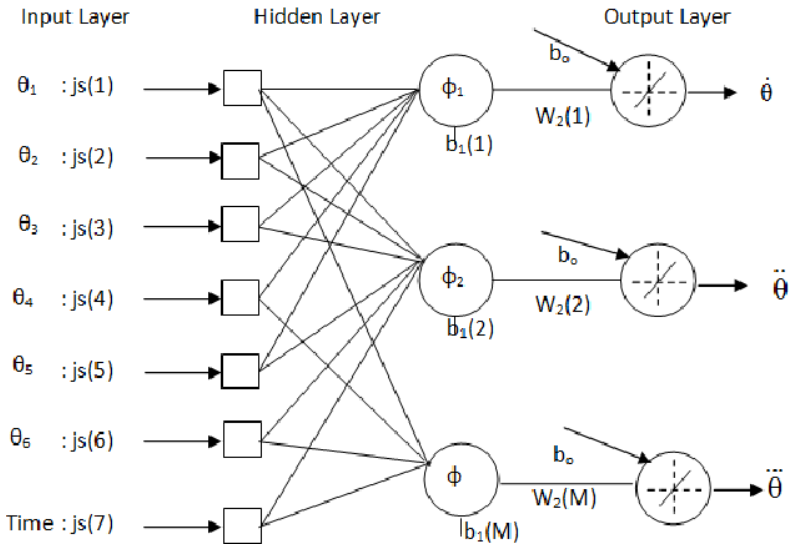


Figure 7. RBF network for prediction of angular velocity, angular acceleration and jerk of each joint (7by3 network)

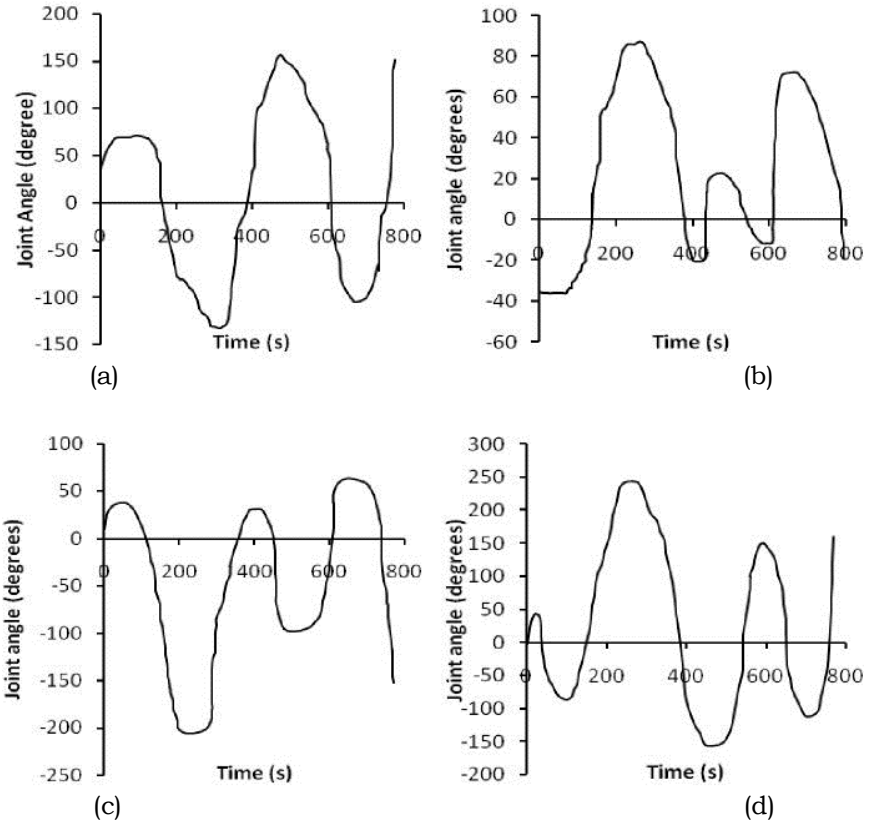
3. Results and Discussion

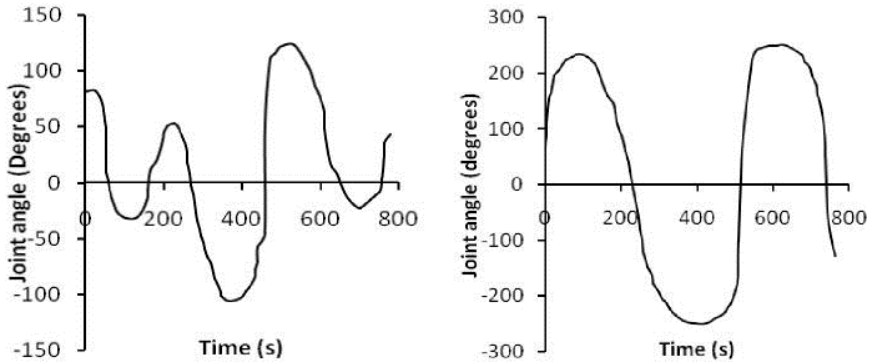
The various aspects of the proposed RBF ANN based approach for estimation of kinematic control parameters are explained in the previous sections. Among various issues, the determination of data for training of a neural network and the selection of appropriate neural network architecture are important. For generating the required training data, 591 configurations are selected from the entire workspace of the robot. The joint angle trajectory i.e. joint angle configuration and the time required to reach that configuration for each joint of

the robot is as shown in Fig. 8(a)-(f). After fitting quintic B-spline to this data shown in Fig.8 and determining the derivatives, the output data was obtained as explained earlier. For training purpose, 70% of total data was used and testing was done with reaming 30% data. Using this training data, training of the 7by6 as well as 7by3 RBF ANN architecture was carried out. The parameters used for training of 7by6 and 7by3 networks i.e. the spread constant, number of epochs, the expected training goal and the achieved training goal are given in Table 1.

Table 1 Training Parameters used for Training of RBF ANN

Configuration	Spread constant	No. of epochs	Training goal expected	Training goal achieved
7 by 6 - velocity	0.46	3550	0.001	0.004
7 by 6 - acceleration	0.46	3570	0.001	0.003
7 by 6 - jerk	0.42	3560	0.001	0.001
7 by 3- Joint 1	0.49	3590	0.001	0.005
7 by 3 -Joint 2	0.42	3590	0.001	0.003
7 by 3 -Joint 3	0.42	3590	0.001	0.016
7 by 3 -Joint 4	0.42	3590	0.001	0.005
7 by 3 -Joint 5	0.42	3590	0.001	0.017
7 by 3 -Joint 6	0.42	3590	0.001	0.008





(e) (f)

Figure 8. Joint angle trajectory obtained for (a) joint 1 (b) joint 2 (c) joint 3 (d) joint 4 (e) joint 5 (f) joint 6

Table 2 % Error in Prediction of Velocity, Acceleration and Jerk at Various Joints by using 7by6 and 7by3 Architecture

RBF ANN Architecture	% Error in velocity		% Error in acceleration		% Error in jerk	
	7by 6	7by 3	7by 6	7by 3	7by 6	7by 3
Joint 1	0.467	2.0628	1.1098	2.7382	1.0411	1.6373
Joint 2	2.001	2.0015	3.1388	5.7162	0.9128	1.9685
Joint 3	1.8963	3.8202	3.1995	4.8073	8.3838	9.4545
Joint 4	1.7784	2.5383	1.6804	2.3135	5.2359	9.5074
Joint 5	3.5550	3.5657	3.7244	5.0593	2.5	3.9623
Joint 6	3.2399	5.4212	3.6580	7.8119	5.9837	8.3041

Table 2 shows the percentage error in prediction of the joint velocity, acceleration and jerk by implementing using 7by6 and 7by3 RBF ANN architecture for each joint. This percentage of error is determined from 30% of total dataset used for testing purpose. From the results presented in Table 2, it can be seen that RBF ANN with 7by6 architecture shows a percentage error about 0.467-3.55 in prediction of velocity, 1.1-3.72 in prediction of acceleration, whereas 0.9-8.38 in prediction of jerk for all the six joints. On the other hand, for RBF ANN with 7by3 architecture shows a percentage error of about 2.06-5.42 in prediction of velocity, 2.73-7.81 in prediction of acceleration whereas 1.63-9.50 in prediction of jerk for all the six joints. These results are presented in Fig. 9, 10 and 11 in the form of a bar chart shown for velocity, acceleration and jerk respectively. From the results presented in these bar charts and in Table 2, it noticed that the 7by6 architecture shows comparatively less percentage of error in the prediction of all the kinematic control parameters as compared to those obtained with 7by3 architecture. The main reason for this can be attributed to the nature of training data used by each of these networks. The data used for training of 7by3 neural network consists of training patterns of three types i.e. angular velocity, acceleration and jerk. On the contrary, 7by6 architecture utilized training patterns of only one type at a time i.e. either velocity or acceleration or jerk. Moreover, from the training parameters presented in the Table 1, it can be seen that the training goal achieved by 7by6 architecture is nearly the same as the desired goal.

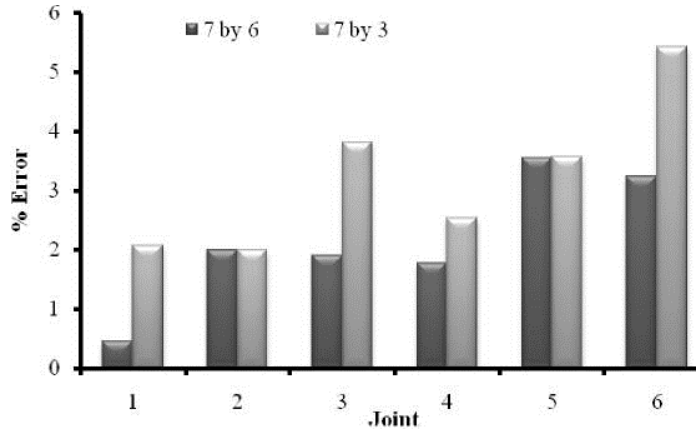


Figure 9. % Error in prediction of velocity using 7by 6 and 7by 3 ANN architecture

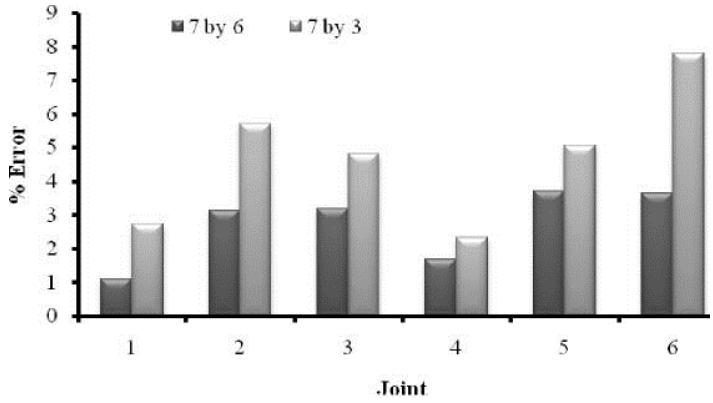


Figure 10. % Error in prediction of acceleration using 7by 6 and 7by 3 ANN architecture

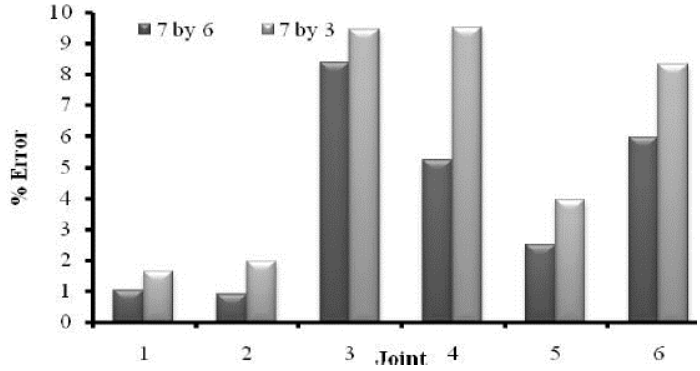
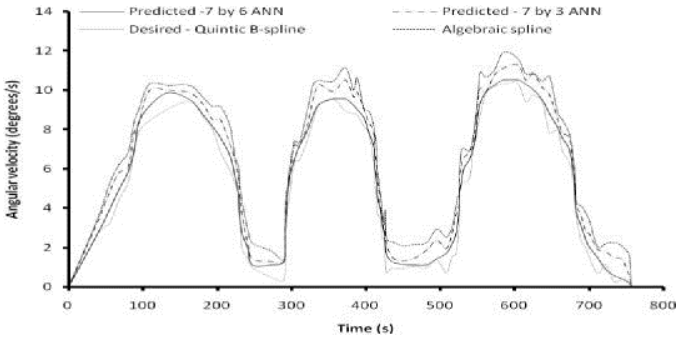
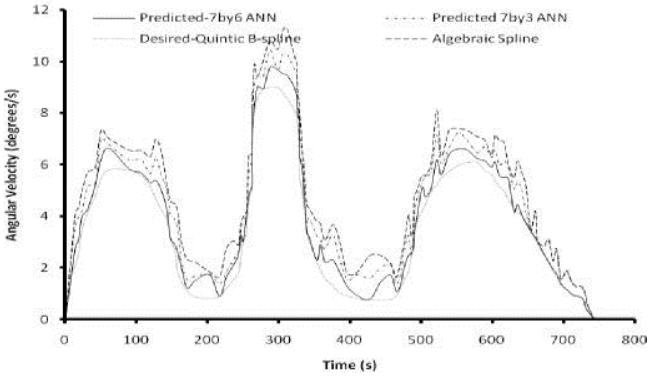


Figure 11. % Error in prediction of jerk using 7by 6 and 7by 3 ANN architecture

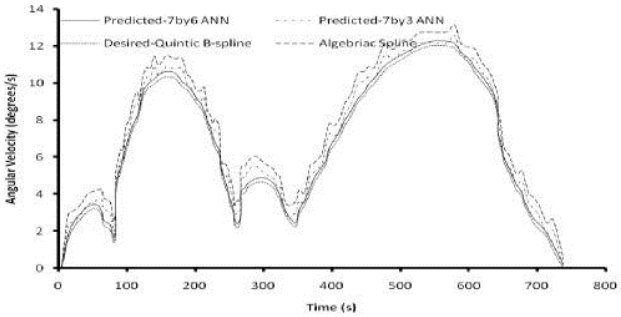
In order to observe the performance of the proposed approach, angular velocity for 30% of testing data was determined by using two types of the splines 1) quintic B-spline and 2) algebraic spline and two types of the neural networks a) 7by6 and b) 7by3 RBF ANN architectures. The 30% of testing data was fitted with quintic B-spline as explained earlier and the corresponding derivative of B-spline was obtained to get the values of angular velocity. To the same data, algebraic spline was fitted in order to determine angular velocity using "jtraj" function of robot toolbox for MATLAB [24]. The "jtraj" function considers input as joint angles and time and gives out trajectory control parameters. On similar lines, the values of the joint angle velocities for 30% testing data were obtained by training 7by6 and 7by3 RBF ANN networks for prediction of trajectory control parameters. These values of the joint velocities with respect to time are presented for each of six joints in Fig. 12(a)-12(f). Fig. 12 (a) shows the values of velocities obtained for joint 1 of robot. From the results presented in Fig. 12 (a), it can be inferred that the values of velocities predicted by 7by6 RBF ANN architecture are nearly same as of obtained using quintic B-spline which is considered as the desired solution. Further, the values of joint velocities predicted by 7by3 RBF ANN architecture show some deviation from the desired values. Moreover, it can be seen that the joint velocities obtained by fitting algebraic spline using "jtraj" function show large deviation from the desired values. Similar trends can be seen for joint 2-6 of robot in Figs. 12(b)-12(f). From these results, it can be again inferred that, 7by6 RBF ANN network not only predicts values better than 7by3 RBF ANN network but also shows superior performs than algebraic spline.



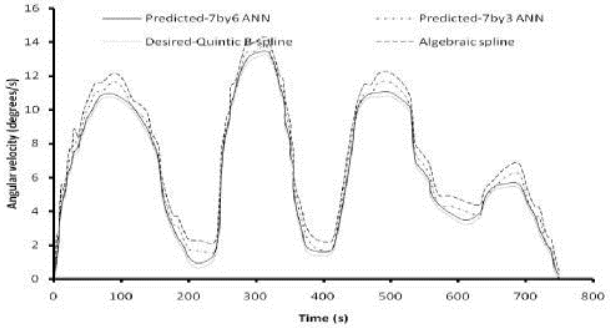
(a)



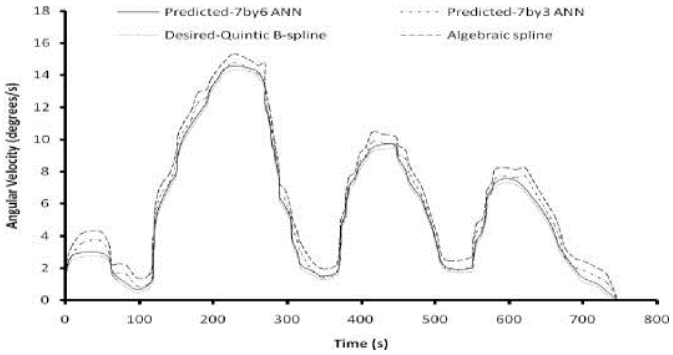
(b)



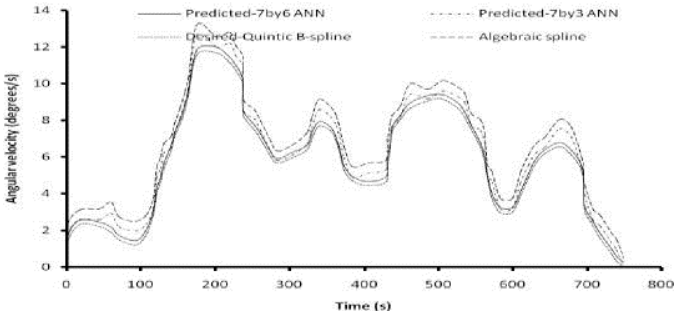
(c)



(d)



(a



(d)

Figure 12. Angular velocity tracking for (a) joint 1 (b) joint 2 (c) joint 3 (d) joint 4 (e) joint 5 (f) joint 6

The proposed approach and the developed control scheme are implemented to determine trajectory control parameters for the three robots which are sharing common workspace to execute a pick and place task independently. The details of the developed control scheme, three robot case study and comparison of performance is given in the next section.

3.1 Three Robot Case Study

The typical example considered for this study is as shown in Fig. 13. The workcell consists of three identical robots sharing common workspace along with stationary obstacles. The objective is to move the end effector of the robot R_1 from start position P_1(-779.38,66.81,360.24) to goal position P_5(1097.04,-36.93,406.25), robot R_2 from start position P_6(-764.59,61.57,366.7) to goal position P_8(932.36,-131.77,414.06) and robot R_3 from start position P_9(-197.17,220.7,357.02) to goal position P_{12}(944.37,-43.60,440.81).

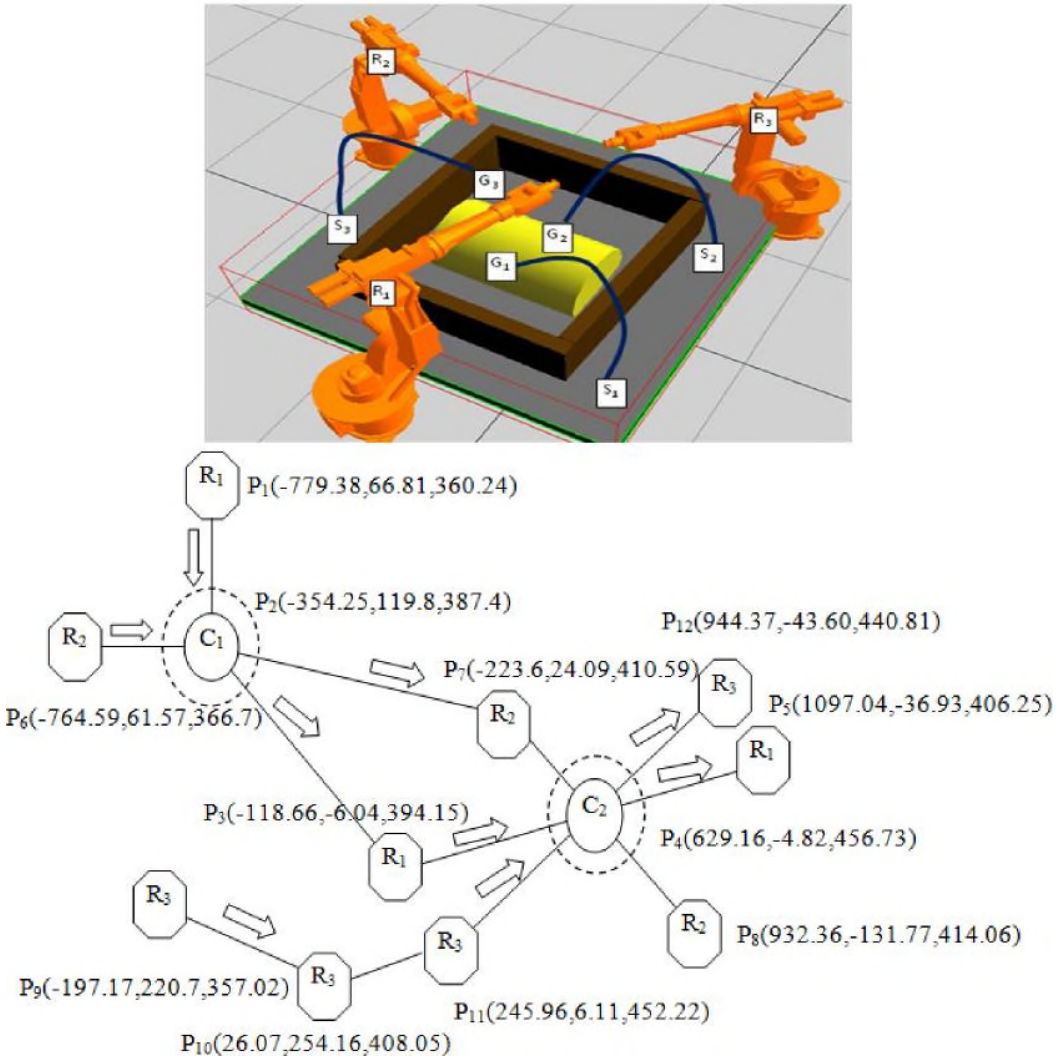


Figure 13 Three Robot Scenario

The proposed method of trajectory tracking using radial basis function is implemented to find out the kinematic control parameters i.e. joint velocity, acceleration and jerk. The maximum and average values of these control parameters were obtained using both the methods. These values were used to determine the total time required to complete the task by each robot and torque required to complete the task was also computed. From the data obtained, it was seen that the values of velocity, acceleration and jerk obtained using algebraic spline is more than those obtained using proposed approach. Because of this, the time required to achieve the task by each robot is less.

The major problem associated with this is the possibility of conflict among participating robots as there is significant difference between desired time and computed time. Whereas, this difference is insignificant when trajectory control parameters are computed using the proposed approach. These findings can be seen from the data presented in Table 3. When these values of control parameters were used for computation of total torque required for each robot then form results presented in Table 4 it can be concluded that algebraic spline technique demands for more torque than proposed approach. With the proposed approach, the reduction in velocity is 4-5%, acceleration is 5-9% and in jerk it is 1.5-7.5% as compared with control algorithm. Because of this, the reduction in torque required to complete the task is 3-6 % than the algebraic spline.

Table 3. Time required for completion of the task by using velocity profiles generated by various trajectory generation techniques (seconds)

Robot	Distance travelled (m)	Expected time	RobotiCad	Proposed approach
Robot 1	2.553	7.58	6.35	7.41
Robot 2	2.218	7.48	6.89	7.85
Robot 3	1.589	9.37	7.86	9.62

Table 4. Computation of total torque using various trajectory generation techniques (Nm)

Robot	RobotiCad	Proposed Approach
Robot 1	1525.28	1452.65
Robot 2	522.6	490.7
Robot 3	1330	1285.018

The main reason behind the better performance of the proposed approach is the use of quintic B-spline for obtaining training data. For obtaining training data, entire workspace is considered and fitting of b-spline is done which assures smooth trajectory and continuity up to third order that is velocity, acceleration and jerk are continuous. Whereas, joint trajectory function of RobotiCad toolbox carries out by fitting fifth order algebraic spline, which ensures the continuity of velocity, acceleration, and jerk. Though with algebraic splines, computation of parameters is easy, the computational load increases with increase in degree

of the polynomial used for fitting which may show large oscillations of position function and its time derivatives.

4. Conclusions

In this work RBF ANN based approach for determination of trajectory tracking/kinematic control parameters is developed and implemented successfully. The ANN based approach proposed in this work tries to overcome the disadvantage associated with control schemes that depend upon the model of the system being controlled. Neural network does not rely on the kinematic and dynamic model of robot and hence the neural controller can be used for any type of the robot. In addition, any change in the physical setup of the system such as addition of a new tool would only involve training and would not need any major software modification. This approach could be particularly useful for industrial robots that can be taught a small set of paths depending upon the task specified. The generation of training data by fitting a quintic B-spline seems to be more practical than obtaining data from sensors or by fitting cubic spline. The use of quintic B-spline ensures continuity up to third derivative and facilitates the prediction of joint angle velocity, acceleration, and jerk. Moreover, since it is an interpolation technique, more number of training patterns can be obtained easily from limited number of joint angle configurations. The neural network based approach presented in this paper is independent of type of the robot and does not need the dynamic model of robot to ensure the movement of robot within the kinematic limits of the robot.

References

[1] K. S. Fu, R. C. Gonzalez and C. S. G. Lee, "Robotics control, sensing, vision and intelligence", McGraw-Hill, New York, **(1987)**.

[2] G. Antonelli , S. Chiaverini and G. Fusco, "A new online algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits", IEEE Transactions on Robotics and Automation, vol. 19, no. 1, **(2003)**, pp.162-167.

[3] B. Karlik and S. Aydin, "An improved approach to the solution of inverse kinematics problem for robots", Engineering Applications of Artificial Intelligence, vol. 13, **(2000)**, pp. 159-164.

[4] A. T. Hasan, N. Ismail, A. M. S. Hamouda, I. Aris, M. H. Marhaban, "Trajectory tracking for a serial robot manipulator passing through singular configurations based on the adaptive kinematics Jacobian method", Proc. ImechE vol. 223, **(2009)**, Part I: Systems and control engineering, doi: 10.1243/09596518JSCE655.

[5] E. Whitney, "Resolved motion rate control of manipulators and human prostheses", IEEE transaction on Man-Machine Systems, MMS-10, **(1969)**, pp. 47-53.

[6] Z. Hu, Z. Fu and H. Fang, "Study of singularity robust inverse of Jacobian matrix for manipulator", In proc. of the First international Conference on Machine Learning and cybernetics, Bejing, **(2002)**, pp. 406-410.

[7] Y. Nakamura and H. Hanafusa, "Inverse kinematics solutions with singularity robustness for robot manipulator control", International Journal of Dynamics Systems Measurements Control, vol. 108, **(1986)**, pp. 163-171.

[8] C. W. Wampler, "Manipulator inverse kinematics solutions based on vector formulations and damped least squares methods", IEEE Transaction on Systems Man and Cybernetics, vol. 16, **(1986)**, pp. 93-101.

[9] Y. Kuroe, Y. Nakai and T. Mori, "A new neural network learning on inverse kinematics of robot manipulators", In the International Conference on Neural Networks, IEEE world congress on computational intelligence, vol. 5, **(1994)**, pp. 2819-2824.

[10] R. A. Graca and Y. Gu, "A fuzzy learning algorithm for kinematic control of a robotic system", In proceedings of the 32nd Conference on Decision and control , San Antonio, Texas, **(1993)**, pp. 1274-1279

[11] R. Koker, C. Öz, T. Çakar, H. Ekiz, "A study of neural network based inverse kinematics solutions for a three-joint robot", Robotics and Autonomous Systems, vol. 49, **(2004)**, pp. 227-234.

[12] A. T. Hasan, A. M. S. Hamouda, N. Ismail and H. M. A. A. Al – Assadi, "A new adaptive learning algorithm for robot manipulator control", Proc. ImechE, vol. 221, **(2007)**, Part I: Systems and control engineering, pp. 663-672.

[13] K. I. Funahashi, "On the approximate realization of continuous mapping by neural networks", vol. 2, no. 3, **(1998)**, Neural Networks, pp. 183-192.
[14] A. T. Hasan, A. M. S. Hamouda, N. Ismail and H. M. A. A. Al – Assadi, "An adaptive learning algorithm to solve the inverse kinematics problem of a 6 DOF serial robot", Advances in Engineering Software, vol. 37, **(2006)**, pp. 432-438.
[15] Z. Bingul, H. M. Ertune 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**)**, pp. 1- 5.
[16] M. T. H. Baheshti, A. K. Tehrani and B. Ghanbari, "An optimized adaptive fuzzy inverse kinematics solution for redundant robots", In Proceedings of International Symposium On Intelligent Control, Houston ,Texas, **(2003)**, pp. 924-929.
[17] P. Y. Zhang, T. S. Lu and L. B. Song, "RBF network based inverse kinematics of 6R robot", International Journal of Advance Manufacturing Technology, vol. 26, **(2005)**, pp. 144-147.
[18] D. Simon, C. Isik, "A trigonometric trajectory generator for robotic arms", International Journal of Control, vol. 57, no. 3, **(1993)**, pp. 505-517.
[19] A. Gasparetto and V. Zanotto, "A new method for smooth trajectory planning of robot manipulators", Mechanism and Machine Theory, vol. 42, **(2007)**, pp. 455-471.
[20] E. Dyllong and A. Visioli, "Planning and real-time modifications of a trajectory using spline techniques", Robotica, vol. 21, **(2003)**, pp. 475-482.
[21] S. S. Chiddarwar and N. Ramesh Babu, "Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach", Engineering Applications of Artificial Intelligence,doi:10.1016/ j.engappai. 2010.01.028, **(2010)**.
[22] D. F. Rogers and J. A. Adams, "Mathematical elements for computer graphics", Tata Mc-Graw hill, **(1991)**.
[23] C. De Boor, "A practical guide to splines", Springer-Verlag, NewYork, **(1978)**.
[24] R. Falconi and C. Melchiorri, "RobotiCad : an educational tool for robotics", Proc of 17th world congress of Automatic Control, Seoul, Korea, **(2008)**, pp. 9111-9116.
[25] W. Verdonck, "Experimental robot payload identification with application to dynamic trajectory compensation", Ph.D thesis, Katholieke University, Belgium, **(2004)**.
[26] S. S. Chiddarwar and N. Ramesh Babu, "Dynamic priority allocation for conflict free coordinated manipulation of multiple agents", Proceedings of IEEE Conference on Automation Science and Engineering, Bengaluru, **(2009)** Aug. 22-24, pp. 549-554.