Radial Basis Function Network Compensators for Uncertainties of Robotic Manipulators

by

S. M. Ziauddin and A. M. S. Zalzala

Robotics Research Group,
Department of Automatic Control and Systems Engineering,
University of Sheffield

Research Report No. 514

Radial basis function network compensators for uncertainties of robotic manipulators

Abstract

This report proposes a decentralised compensation scheme for uncertainties and modelling errors of robotic manipulators. The scheme employs a central decoupler and independent joint neural network controllers. Recursive Newton Euler formulas are used to decouple robot dynamics to obtain a set of equations in terms of each joint's input and output. To identify and suppress the effects of uncertainties associated with the model each joint is controlled separately by Gaussian radial basis function network controllers using direct adaptive techniques. The effectiveness of the proposed adaptive control scheme is demonstrated by controlling the three primary joints of PUMA 560. Simulation results show that this control scheme can achieve fast and precise robot motion control under substantial payload variations.

1. Introduction

Robot manipulators have become increasingly important in the field of industrial automation. In today's automated factory robots are widely used in material handling, painting and welding tasks. High speed and high precision trajectory tracking capability is of prime concern in manipulator applications. Conventional controllers for industrial robots are based on the independent joint control schemes in which each joint is controlled separately by a simple position servo-loop with predefined constant gains. This control scheme is adequate for simple pick-and-place tasks where only point to point motion is of concern. However, in tasks where precise tracking of fast trajectories under different payloads is required, the existing robot control schemes are severely inadequate. Any significant gain in performance for tracking the desired trajectory as closely as possible over a wide range of manipulator motion and payloads requires the consideration of adaptive control techniques.

Globally stable adaptive controllers for mechanical manipulators have been proposed in literature [16,17]. These approaches, however, require a detailed description of the structure of the dynamic model. Moreover the explicit identification of inertia parameters in these controllers is computationally intensive [22]. They can therefore be vulnerable to unfavourable properties from the view point of reliability and maintenance of the controllers. Also the adaptive control theory can adapt unknown parameters in a mathematical referenced model but it can not adapt the structure of the model itself. In a real environment the inevitable uncertainty of the available model resulting from hardly measurable non-linear friction, gear non-linearities, changes in motor dynamics and payload variations makes the control task extremely difficult. In the presence of errors between actual system structure and the model structure, robustness of control system decreases. Consequently, it may be desirable to utilise learning control techniques that permit the implementation of non-linear compensators without detailed prior knowledge of the plant dynamics. Neural nets provide one such alternative.

Advances in artificial neural networks have provided the potential for new approaches to the control of systems with complex, unknown and non-linear dynamics. The advantages of using neural networks for control applications can be summarised as

three-fold. First, they have a flexible structure to express non-linear systems which may provide for a robust controller. Second, because of their structure they have a flexible learning capability as against adaptive control method which stays within unknown parameter identification of a predefined model. In addition parallel processing and fault tolerance are easily achieved.

Recently there has been considerable interest in the application of neural networks to robot control [6, 10, 12, 21]. Most of these efforts aim at generating the inverse dynamic models of manipulators in the neural networks, with the underlying assumption being that neural networks are capable in principle of approximating any well behaved function. This does not imply however, that it is equally easy to learn to represent any function from a finite amount of training data. As robot dynamics are extremely non-linear, the neural network may require an excessive amount of training data in order to yield a reasonable generalisation. Because of this difficulty some neural network based robot control schemes have been developed in which the neural network assumes a supportive role in the overall adaptive loop rather than trying to approximate the whole inverse dynamics [4, 8, 23]. Such schemes are based on the belief that neural networks perform effectively when they are not required to learn too much. Also as the inverse dynamics of the manipulators can be computed efficiently to a fair degree of accuracy using the recursive Newton Euler equations, it is reasonable to utilise them and assist them through neural networks to compensate for the uncertainties of robotic manipulators.

The commonly used neural network structure in robot control is the multi-layered perceptrons using the back propagation learning algorithm [20]. However, it is well known that this type of algorithm suffers from the drawback of slow convergence. Moreover the multi-layered perceptrons are non-linear in parameters and are therefore not amenable to direct adaptive control techniques. Although local convergence results are present for layered networks [1, 2], there is no theoretical evidence about how well layered neural networks without bias weights can approximate non-linear functions. On the other hand, radial basis neural networks [11] have the advantage of being linear in parameters, provided their centres are kept constant, and are therefore well suited for direct adaptive techniques. Several algorithms utilising Gaussian radial basis functions to adaptively compensate for the non-linearities in a certain class of plants are available [9, 13, 14]. However, these algorithms cannot be directly applied to manipulator control, largely because of the large amount of coupling present between the joints.

This report presents a new de-centralised approach to adaptive robot control using neural networks which does not require inverse model formulation. The control scheme uses a framework of de-centralised Gaussian network adaptive controllers utilising direct adaptive control techniques for weight updates. To get around the problem of inter-joint coupling, the control scheme first applies non-linear state feedback control [3] to the robot dynamics, thus obtaining a set of decoupled equations in joint co-ordinates and then makes use of Gaussian radial basis function networks to compensate for the unmodeled dynamics and to suppress tracking errors. Simulation studies on PUMA 560 indicate the effectiveness of the scheme.

The report is organised as follows: The application of Feedback decoupling theory to robot control, and its practical aspects are discussed in section 2, while sections 3.1,

3.2 give the controller structure and the overall control scheme. Simulation results are presented in section 4 and section 5 gives conclusions. Radial basis neural networks are briefly discussed in the appendix.

2. Non-linear decoupled control

The application of non-linear decoupled control to robotics is briefly discussed in this section and details about the general non-linear decoupling theory can be found elsewhere [3]. The Lagrange Euler equations of motion of an n link robot can be written in matrix notation as

$$D(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) + g(\theta) = u(t) \tag{2.1}$$

where $u(t)$ is an $n\times1$ applied torque vector for junction actuators, $\theta(t)$ is the $n\times1$ angular position vector, $\dot{\theta}(t)$ is the $n\times1$ angular velocity vector, $\ddot{\theta}(t)$ is the $n\times1$ acceleration vector, $g(\theta)$ is an $n\times1$ gravitational force vector, $h(\theta,\dot{\theta})$ is an $n\times1$ coriolis and centrifugal force vector and $D(\theta)$ is an $n\times n$ acceleration-related inertia matrix. The above model consists of n second order coupled differential equations. The differential order of the non-linear system is therefore 2. Since $D(\theta)$ is always non-singular, the above equation can be rewritten as

$$\ddot{\theta}(t) = -D^{-1}(\theta)[h(\theta,\dot{\theta}) + g(\theta)] + D^{-1}(\theta)u(t) \tag{2.2}$$

The control objective is to find a feedback decoupled controller u(t):

$$u(t) = F(\theta,\dot{\theta}) + G(\theta,\dot{\theta})w(t) \tag{2.3}$$

where $w(t)$ is an n-dimensional reference input vector, $F(\theta,\dot{\theta})$ is an $n\times1$ feedback vector for decoupling and pole assignment and $G(\theta,\dot{\theta})$ is an $n\times n$ input gain matrix so that the overall system has a decoupled input output relationship. The particular form of equation (2.2) renders it easy to visualise such a controller. Let $u(t)$ be taken as

$$u(t) = h(\theta,\dot{\theta}) + g(\theta) - D(\theta)\begin{bmatrix} \alpha_{11}\dot{\theta}_1(t) + \alpha_{01}\theta_1(t) - \lambda_1 w_1(t) \\ \vdots \\ \alpha_{1n}\dot{\theta}_n(t) + \alpha_{0n}\theta_n(t) - \lambda_n w_n(t) \end{bmatrix} \tag{2.4}$$

where α_{ij} and λ_i are arbitrary scalars. By replacing $u(t)$ in equation(2.3) by its value from equation (2.4) we get

$$D(\theta)\begin{bmatrix} \alpha_{11}\dot{\theta}_1(t) + \alpha_{01}\theta_1(t) - \lambda_1 w_1(t) \\ \vdots \\ \alpha_{1n}\dot{\theta}_n(t) + \alpha_{0n}\theta_n(t) - \lambda_n w_n(t) \end{bmatrix} = 0 \tag{2.5}$$

Since $D(\theta)$ is non-singular the above equation becomes

$$\ddot{\theta}_i(t) + \alpha_{1i}\dot{\theta}_i(t) + \alpha_{0i}\theta_i(t) = \lambda_i\omega_i(t) \qquad i = 1,2,\ldots,n \tag{2.6}$$

which indicates the final decoupled input output relationships of the system. By properly selecting α_{ij}s in equation (2.6) poles of the decoupled systems can be placed at desired positions. Hence the manipulator can be considered as n independent, decoupled, second order, time invariant systems. An efficient way to compute the

controller u(t) is through the use of Newton Euler equations by substituting their joint acceleration inputs ($\ddot{\theta}_i(t)$) by $\lambda_i w_i(t) - \alpha_{1i}\dot{\theta}_i(t) - \alpha_{0i}\theta_i(t)$.

The final decoupled input output relationships of the manipulator as derived above are achievable only if the dynamics of the manipulator are known exactly. In practice this would hardly be the case. If modelling errors or unknown disturbances exist, performance of non-linear state variable feedback control (2.4) will deteriorate. In the presence of modelling errors due to unknown friction, gear dynamics, unknown motor dynamics and payloads the independent joint equations (2.6) can be regarded as non-linear systems of the general form

$$\ddot{\theta}(t) + f(\theta, \dot{\theta}) = b(\theta, \dot{\theta})w(t) \tag{2.7}$$

where $f(\cdot)$ and $b(\cdot)$ are unknown non-linear functions of states. Effective compensation for these non-linear effects is essential for accurate tracking of joint trajectories.

To control non-linear systems of the type (2.7) using conventional adaptive control techniques, $f(\cdot)$ and $b(\cdot)$ are represented by the relations:

$$\begin{aligned} f &= \sum_{i=1}^{N} k_{1i} f_i \\ b &= \sum_{i=1}^{M} k_{2i} b_i \end{aligned} \tag{2.8}$$

with $f_i(\cdot)$ and $b_i(\cdot)$ being continuous known basis functions of states. The coefficients k_{1i} and k_{2i} are then estimated using adaptive control theory [15] and the adaptive control law u(t) is synthesised.

However, If the basis functions are not known, as in the case of unknown robot dynamics, it may still be possible to estimate the required control input if an attempt is made to reconstruct the unknown functions $f(\cdot)$ and $b(\cdot)$, and neural networks can be used for this purpose. Theoretically it is possible for neural networks to approximate a non-linear function through learning. In the following sections a decentralised Gaussian neural net based control strategy is developed to cancel the effects of modelling errors in manipulators and thereby achieve accurate trajectory tracking at high speeds.

3. Adaptive Robot control

3.1. Gaussian neural network based control

Gaussian radial basis function networks (please see appendix) have been found to be a powerful scheme for learning complex input output mappings. It is well known that the solution of finding the weights of such networks can be formulated as a linear problem. To develop a Gaussian network adaptive controller for an independent joint of the manipulator (2.7), the input space for that joint is determined and the centres for the Gaussian networks are fixed in the input space at pre-determined lattice points. A mesh of such centres encompassing the whole input space is thus formed as shown in figure 1. In this figure, Ad represents the set comprising these centres. It is assumed that no state trajectory, i.e. joint angles and there respective velocities, will fall outside the set Ad. Within the set Ad the control input to every joint is a sum of PD control and adaptive control. We define the unknown non-linear function $h = b^{-1}f$ and let h_A and

b_A^{-1} be the radial Gaussian network approximations to the functions h and b^{-1}, respectively. A tracking error metric $s(t)=(\dot{\theta}-\dot{\theta}_d)+\alpha(\theta-\theta_d)$ is defined where $\dot{\theta}_d$ and θ_d are the desired values of joint velocities and joint angles, respectively, as given by the trajectory generator, and α is a positive constant.

The adaptive control law is given by

$$u_{ad}(t)=-b_A^{-1}[\lambda(\dot{\theta}-\dot{\theta}_d)-\ddot{\theta}_d]+h_A+PD \qquad (3.1)$$

where λ is a constant and $\ddot{\theta}_d$ is the desired value of joint acceleration determined by the trajectory generator. The structure of adaptive control law is shown in figure 2. In figure 2, g_i represent the Gaussian radial basis functions, μ_i represent the respective centres, c_i and d_i represent the respective weight connections to the outputs. It can be shown that if the following weight update laws are used

$$\begin{aligned}\dot{c}_i &= -k_1 s_\Delta(t) g(\bar{\theta},\mu_i) \\ \dot{d}_i &= k_2 s_\Delta(t) g(\bar{\theta},\mu_i)[\lambda(\dot{\theta}-\dot{\theta}_d)-\ddot{\theta}_d]\end{aligned} \qquad (3.2)$$

where k_1 and k_2 are positive constants and $s_\Delta(t)=s(t)-\Phi sat(s(t)/\Phi)$ with Φ being a small constant dead zone width and 'sat' representing the saturation function then asymptotic convergence of tracking errors is achieved. For details of the algorithm and proof the reader is referred to [13].

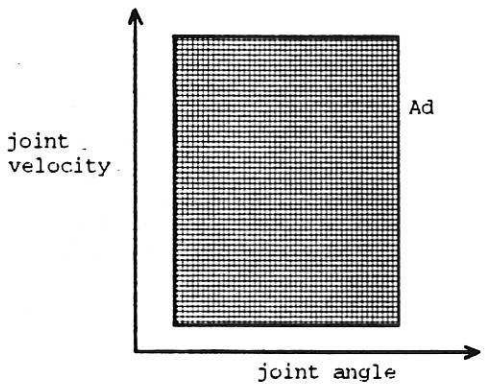

Figure 1. subset of joint state space involved in the design of the adaptive controller.

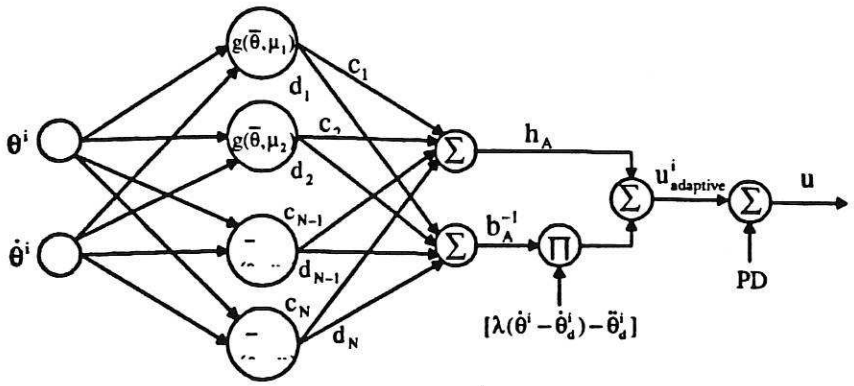

Figure 2. Structure of the adaptive control law for the *i*th joint.

3.2. Model based neural network control for robots

To apply the Gaussian network controllers for adaptive robot control in an efficient manner, the large coupling effects between joints of the manipulators have to be compensated for. For this purpose effective decoupling of the robotic system into independent joint subsystems was carried out as discussed in section 2. The overall control structure for an n link manipulator is shown in figure 3. In figure 3, the objective of the Newton Euler equations is to decouple the dynamics into n independent systems, each of which can be controlled separately. There is one neural network adaptive controller per joint of the robot whose function is to suppress the effects of uncertainties (modelling errors) which otherwise would deteriorate trajectory tracking and introduce large errors.

The Newton Euler block receives the desired values for the joint angles and joint velocities from the trajectory generator, while the acceleration inputs come from the respective neural network output for each joint. The neural network weight update rules as given in equation (3.2) are dependent on the actual and desired values of the joint angles and their velocities. The overall idea behind the scheme is to vary the acceleration inputs of the Newton Euler block to affect changes in the torques to the manipulator in a manner that joint angles follow the desired values as closely as possible.

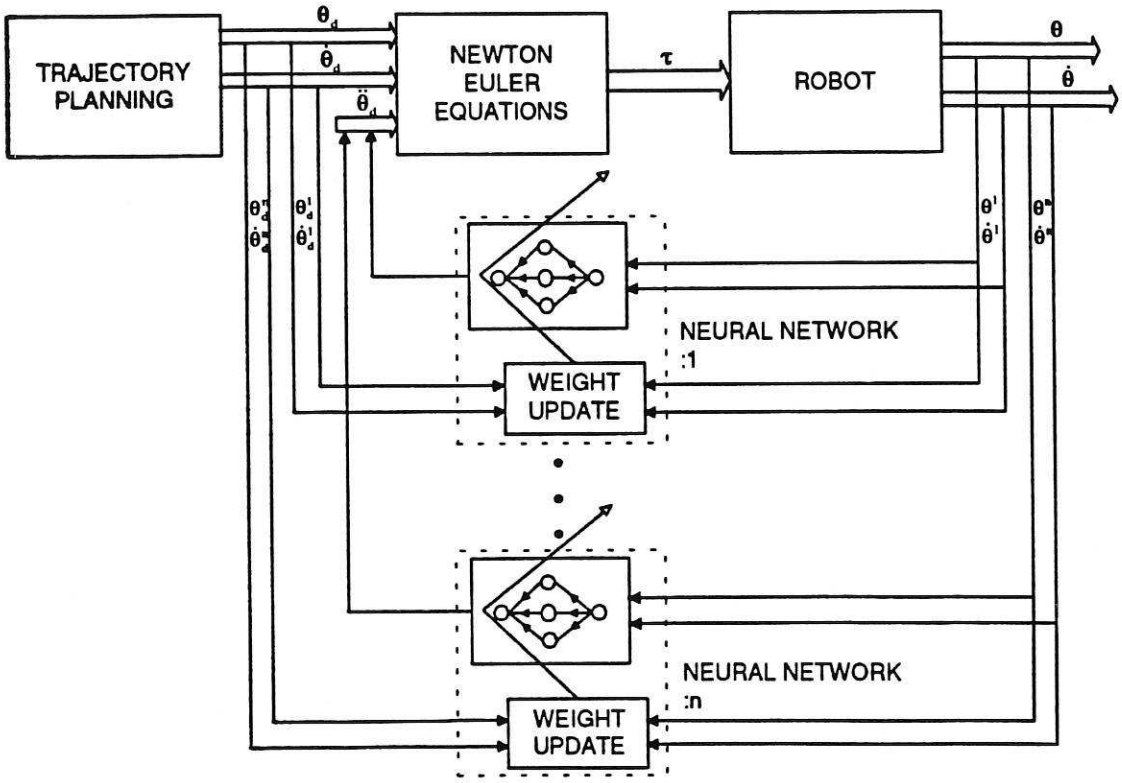

Figure 3. Structure of the decentralised adaptive control scheme for an n link arm.

The scheme is computationally efficient and is amenable to parallel processing implementation within a distributed computing architecture. The Newton Euler formulas for an n link manipulator can be calculated in linear time on a single processor. Moreover very efficient implementations of Newton Euler formulas have been achieved on multiple general purpose processors [5], also, pipelined dedicated

hardwares have been proposed [7] to further cut down the computation time. The adaptive controller part is inherently parallel and two levels of parallelism can easily be identified. A higher level of parallelism exists because of the independent nature of the algorithm at every joint. In addition a much finer level of parallelism exists because of the use of neural networks at each joint. Since a PD controller is acting in parallel with the adaptive controller a comparison of their outputs would give a fair idea of the relative contribution of either of the two. As will be demonstrated in the simulation example, the contribution of the PD controller decreases as the neural network learns and after a few excursions of the trajectory the PD controller can be unplugged altogether without any effect on the overall loop.

The scheme does not require the measurement of joint accelerations nor does it require any force feedback to determine the changing payloads. By dividing the burden of controlling the manipulator among n neural networks the overall state space is conveniently divided into n spaces of reduced order. The structure of each controller is therefore simple and each controller has to learn lesser amount of data. Faster convergence of joint trajectories to their desired values is therefore possible. As will be shown in the next section the model based neural network control scheme produces fast convergence of errors and has very good generalisation properties. The control scheme is particularly robust to changes in payloads. Moreover, as against other neural control approaches it does not require any off-line training phase for the neural nets.

4. Simulation application

In order to demonstrate the effectiveness of the proposed scheme, the six degree of freedom PUMA 560 arm is used in the dynamic simulations [18, 19]. The distributed controller is applied to the three primary joints of the manipulator, while due to its decentralised nature, implementation on the full six degrees of freedom manipulator is a simple extension of this control scheme. Network parameters as used with the PUMA simulator are given in table 1.

mesh size	0.25×0.25
variance	0.02
α_{0i}	100
α_{1i}	20
λ_i	100
No. of Gaussian neurones in mesh	348
NN. weight update rate	100 Hz.
Φ	0.005
k_i	150
α	10

Table 1. Neural network parameters used in PUMA simulations

The first task is to illustrate the capability of the control method to converge to desired trajectories in the presence of modelling errors and unknown large payloads. A non-linear friction term of the form $\tau_{friction} = v \times \dot{\theta} + c \times \mathrm{sgn}(\dot{\theta})$ where v and c are constants, was introduced at each joint and a payload of 10 kg was attached to the end of the

third link. The controller had no prior knowledge of these forces. The desired trajectories are designed so that the end effector starts from an initial point, reaches a final point, stays there for a while, and then returns to the original point. All the joints were required to move along their respective desired trajectories simultaneously so as to induce coupling effects upon each other. The desired and actual trajectories for the three joints, for the 1st, 10th and 30th excursions of the trajectory, are plotted in figures 4 to 6 and the respective errors are plotted in figures 7 to 9. It can be seen in figure 4 that, for joint 1, trajectory following was quite poor at the start (figure 4(a)) but improved as the neural nets gradually got trained and errors were almost unnoticeable at the 30th excursion (figure 4(c)). The reduction in errors is much more evident from figure 7(a to c). Here the errors along the 1st, 10th and 30th excursions of the trajectory are plotted on the same scale. A similar trend exists for joint 2 (figures 5 and 8) and joint 3 (figures 6 and 9). The relative contributions of the PD and the neural net controllers for the three joints are plotted in figures 10 to 12. Figures 10(a) and 10(b) give the outputs for joint 1 at the first excursion. It can be seen that in figure 10(a) the neural net output is quite haphazard during the first excursion and the PD controller has to generate large correction values (figure 10(b)). However, at the 30th excursion the neural net has almost taken over the entire control effort (figure 10(c)) and the PD output remains close to zero (figure 10(d)), throughout the course of the trajectory. Quite similar trends were observed for joints 2 and 3 in figures 11 and 12 respectively. These figures indicate fast convergence of errors. For the sake of comparison, non-linear feedback control without neural net compensation, was applied to the arm. Both poles for each independent sub-system were placed at -10 through suitable choice of α_{ij} (equation(2.6)) as given in table 1. The results are plotted in figures 13 and 14. Figure 13 (a to c) give the desired and actual trajectories for the three joints when all the dynamics are known. It can be seen that exact tracking occurs under known conditions, while figure 14 (a to c) give the same results when unknown friction and payload are added, which cause very large tracking errors.

It may be noticed from the previous results, that being an on line learning scheme, it is eventually able to reduce the errors introduced by any unknown payload, along the desired trajectory to very small values. However, there may be occasions where, within a certain range of payloads, high speed payload-invariant trajectory tracking may be desired. The next task is designed to demonstrate the robustness of the control scheme against payload variations and also illustrates the generalisation properties of the scheme in an environment where payloads may be undergoing large variations. Eleven payloads from 0 to 10 Kg., with an increment of 1 Kg., were selected in a random sequence and the manipulator was moved along the desired trajectory carrying the payload. During the initial phase of the experiment tracking errors were observed, as the payload varied randomly, which diminished rapidly and were almost insignificant at the end of 100 traversals of the trajectory. After this the manipulator was able to perfectly track the desired trajectory carrying any payload in this range. The results for three different payloads, are plotted in figures 15 to 17, where It can be seen that once the neural nets were trained each of the manipulator joint was able to follow its desired trajectory accurately over a wide range of payloads. Also, it should be noted that out of the three payloads of figures 15 to 17, two: 5.5 Kg. and 9.5 Kg., were not used during the initial runs. It may be pointed out that the coupling effects introduced by the unknown payloads are being effectively compensated by the neural nets. The above experiments illustrate fast convergence of trajectories and robustness under partially known dynamics.

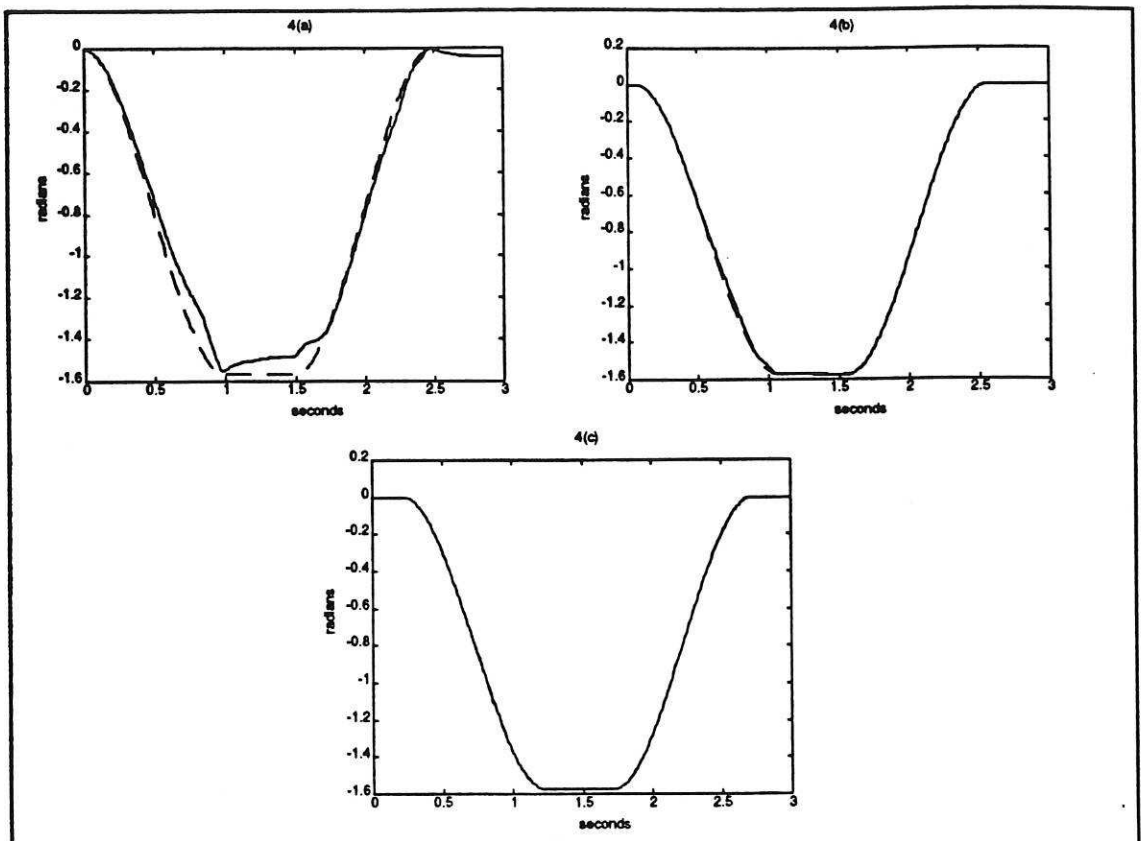

Figure 4. Joint 1 desired and actual trajectories, (a) 1st excursion, (b) 10th excursion, (c) 30th excursion.

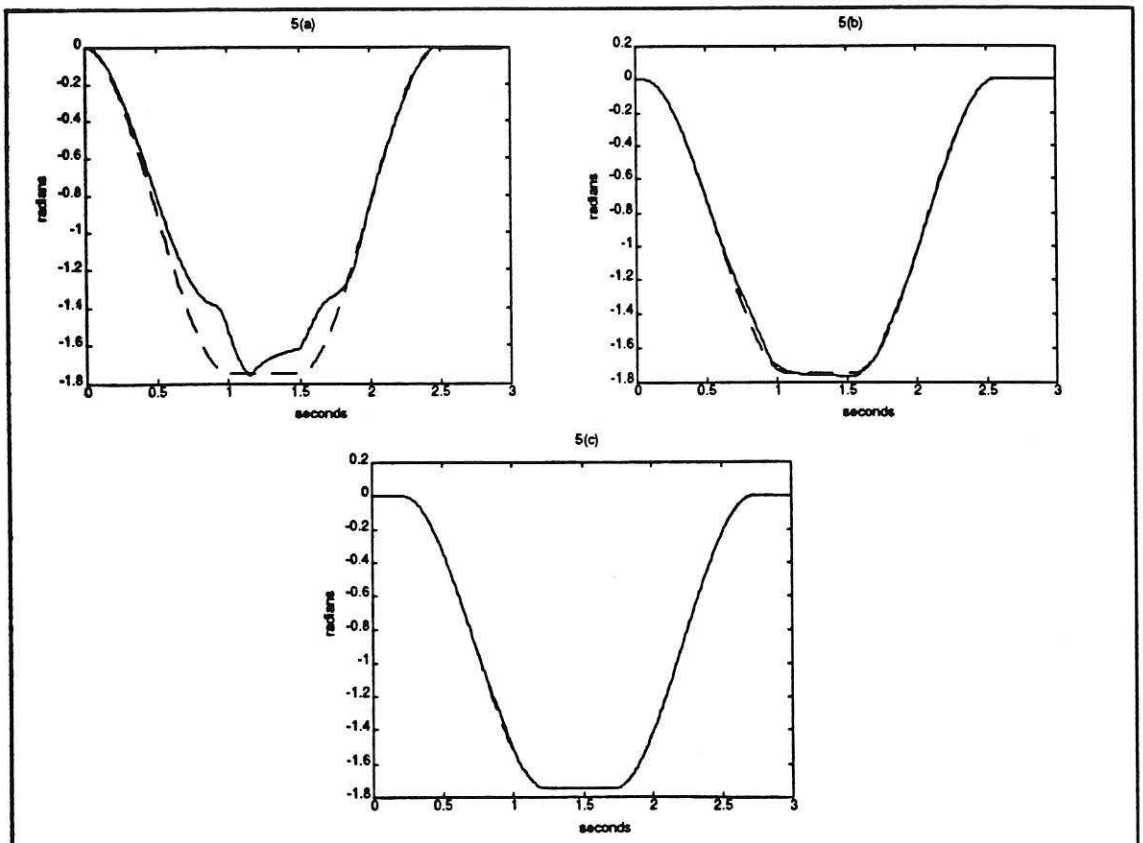

Figure 5. Joint 2 desired and actual trajectories, (a) 1st excursion, (b) 10th excursion, (c) 30th excursion.

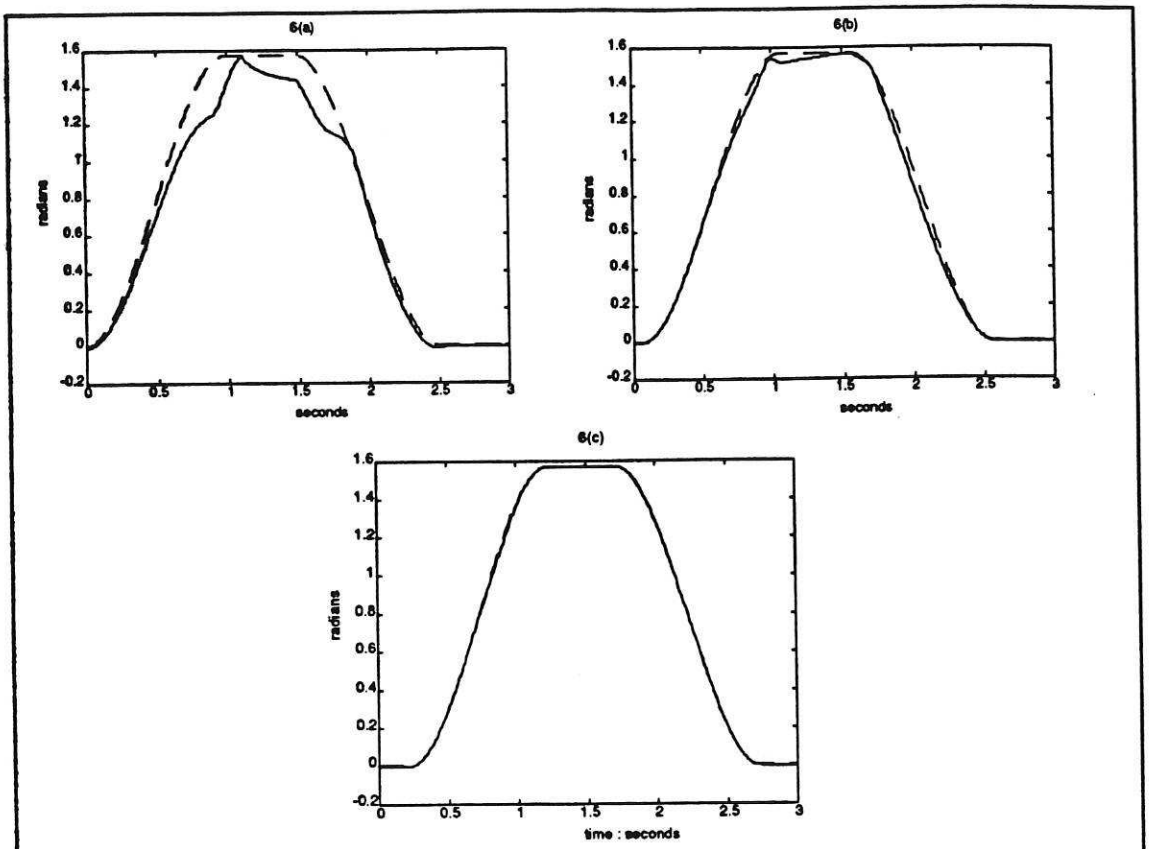

Figure 6. Joint 3 desired and actual trajectories, (a) 1st excursion, (b) 10th excursion, (c) 30th excursion.

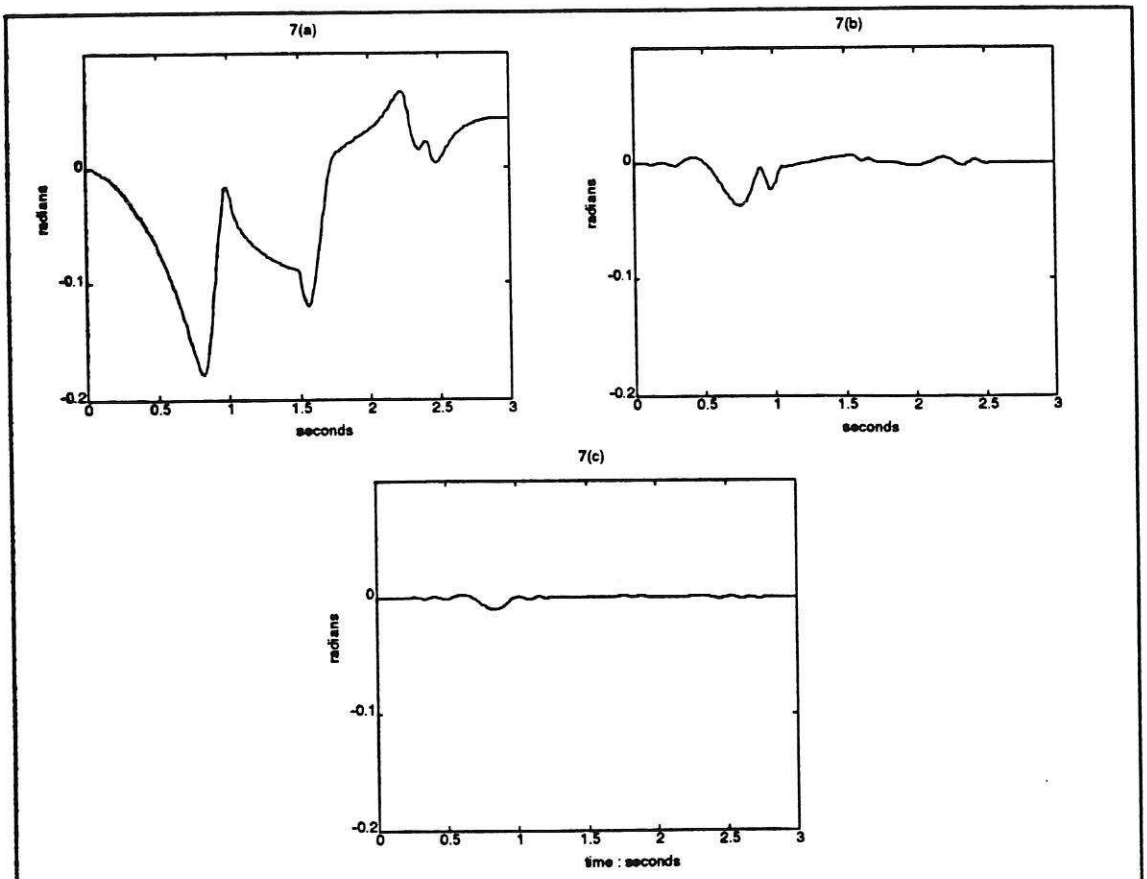

Figure 7. Joint 1 trajectory errors, (a) 1st excursion, (b) 10th excursion, (c) 30th excursion.

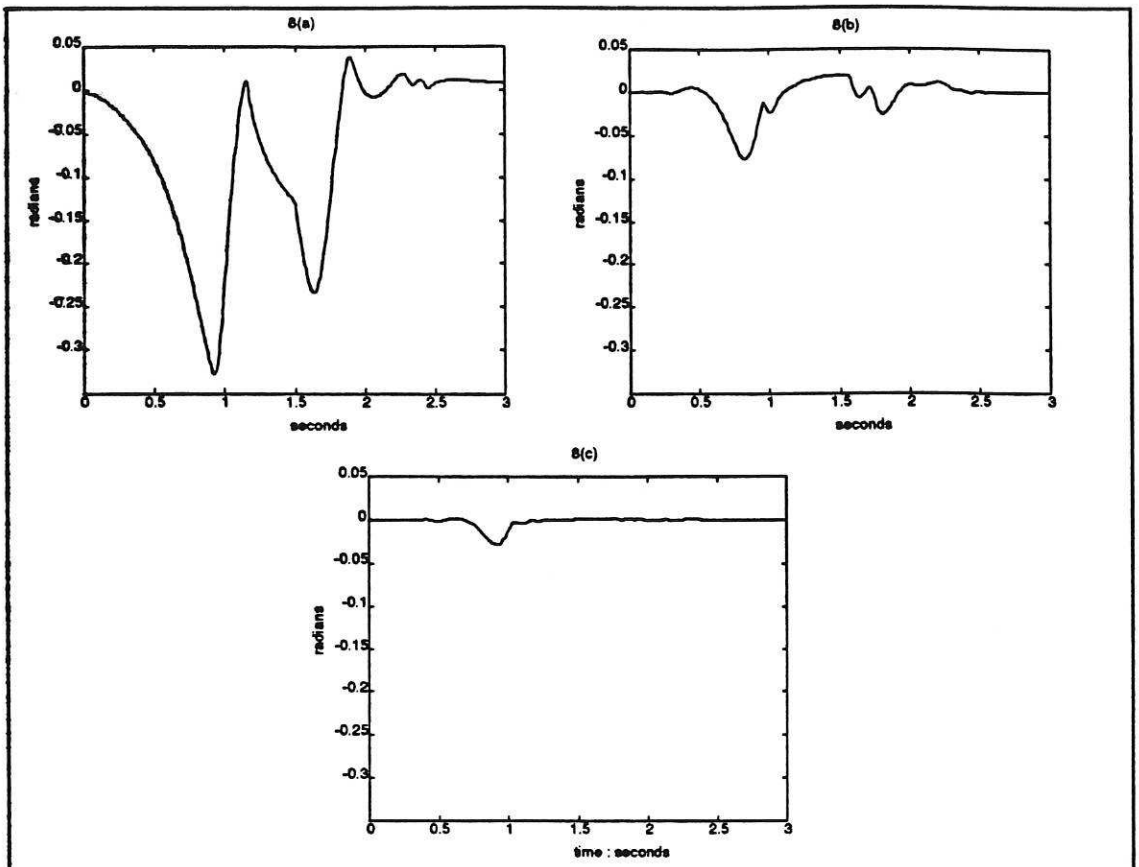

Figure 8. Joint 2 trajectory errors, (a) 1st excursion, (b) 10th excursion, (c) 30th excursion.

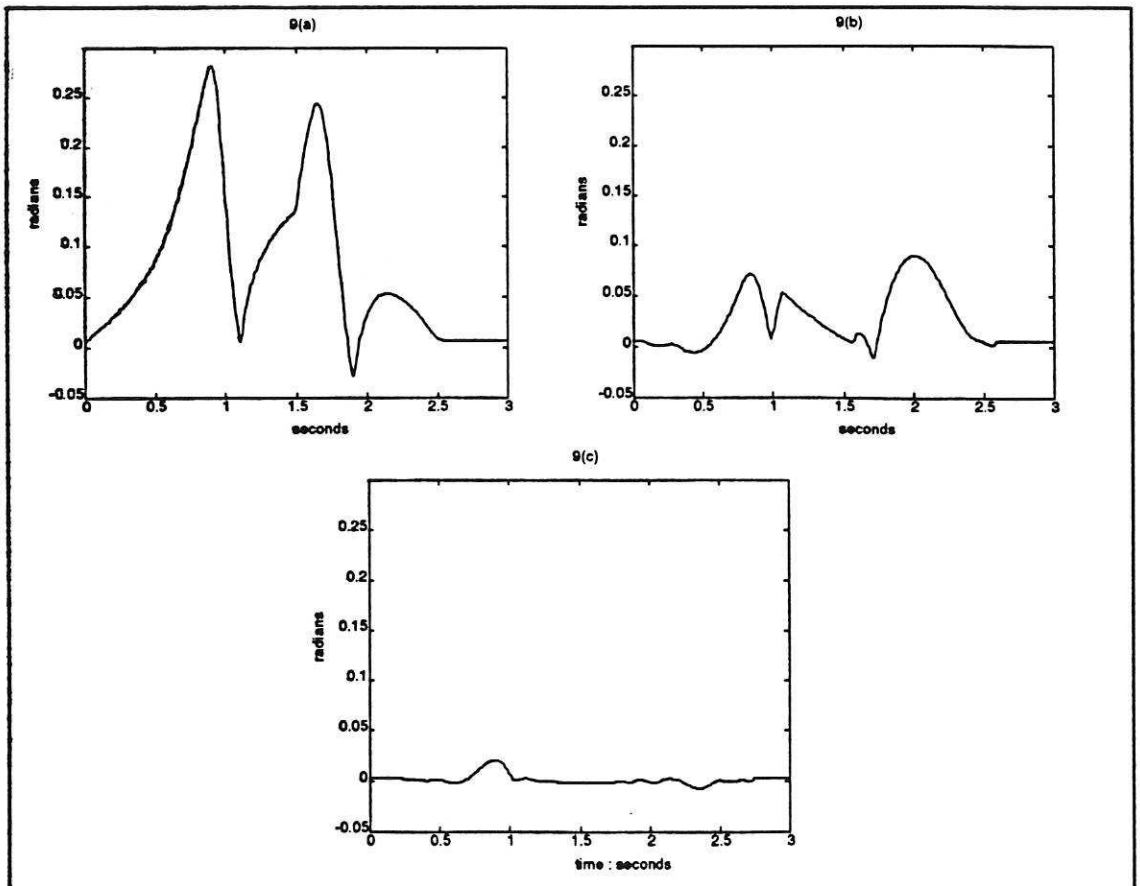

Figure 9. Joint 3 trajectory errors, (a) 1st excursion, (b) 10th excursion, (c) 30th excursion.

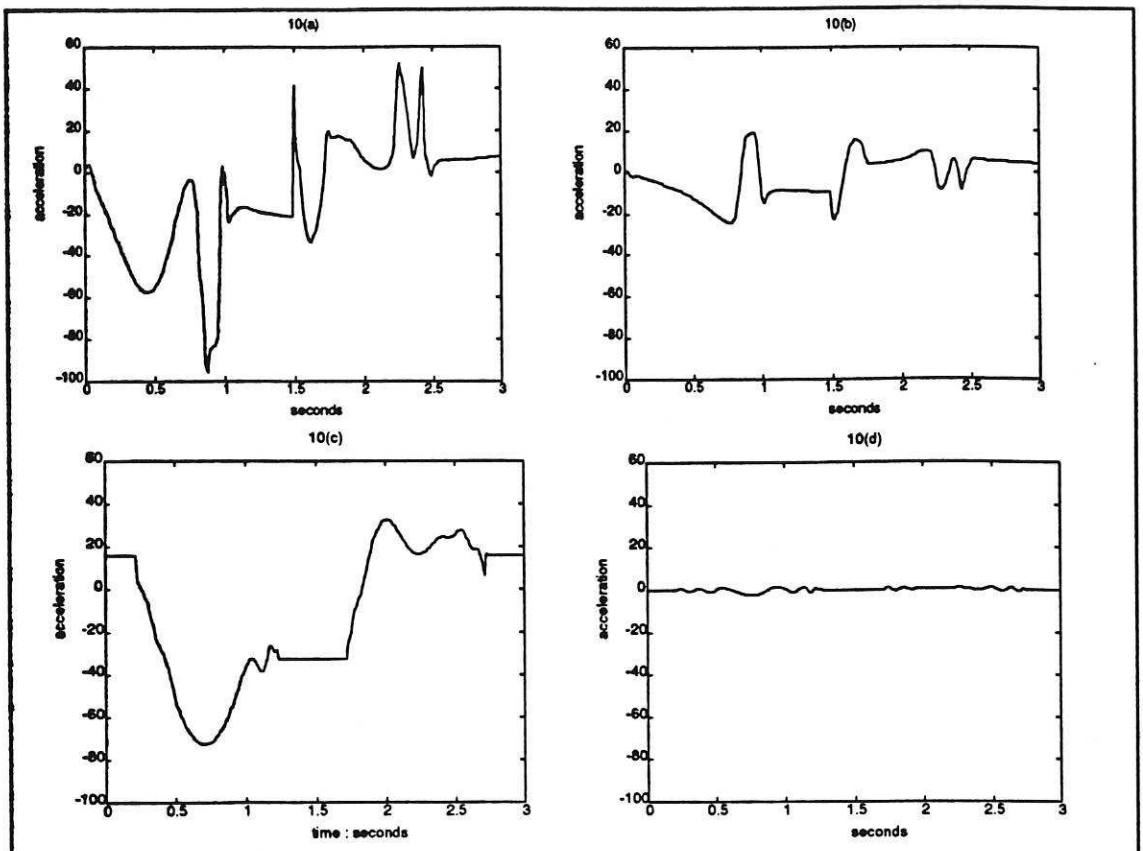

Figure 10. Neural network and PD contribution joint 1, (a) NN contribution 1st excursion, (b) PD contribution 1st excursion, (c) NN contribution 30th excursion, (d) PD contribution 30th excursion.

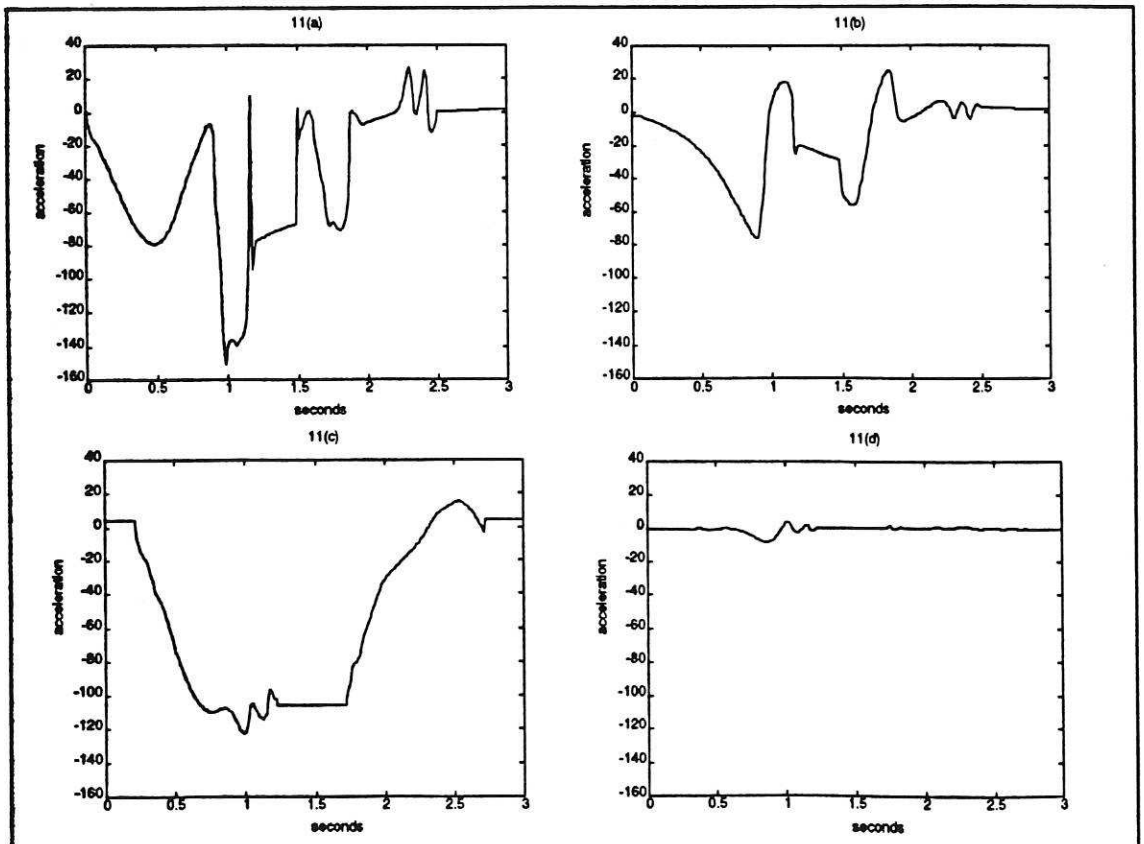

Figure 11. Neural network and PD contribution joint 2, (a) NN contribution 1st excursion, (b) PD contribution 1st excursion, (c) NN contribution 30th excursion, (d) PD contribution 30th excursion.

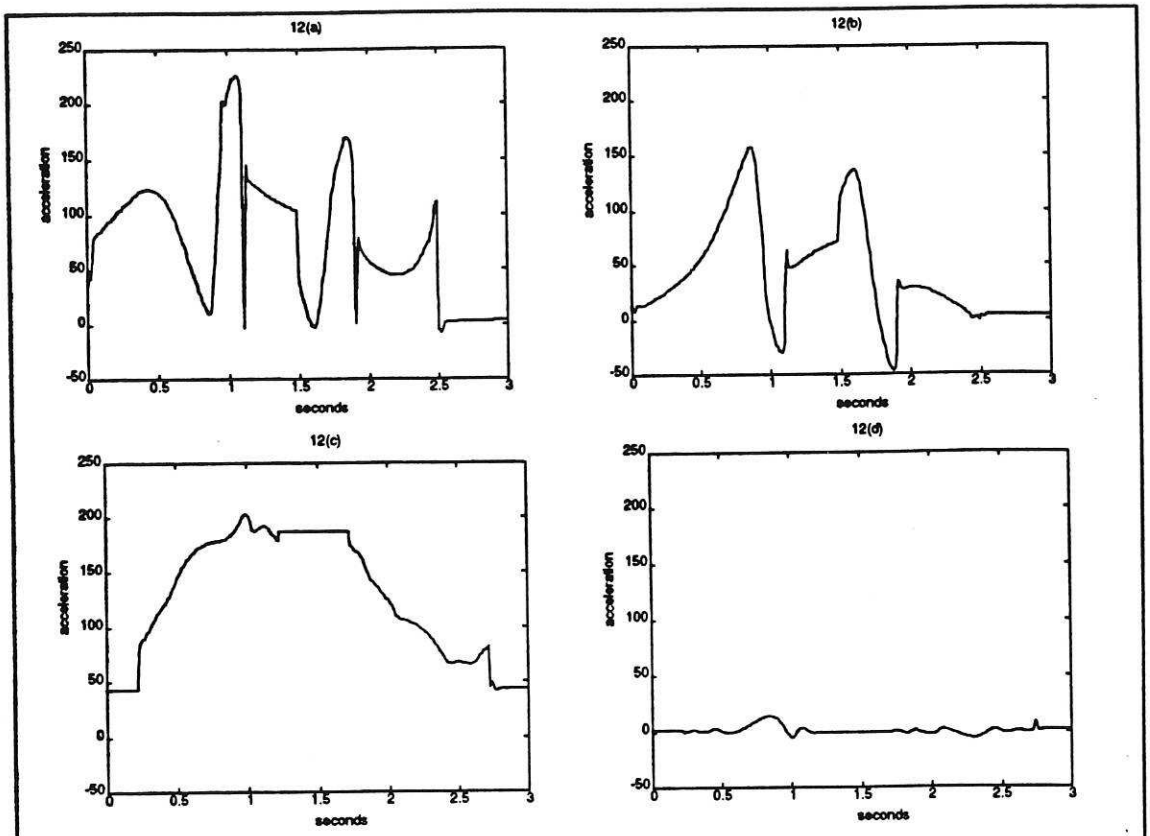

Figure 12. Neural network and PD contribution joint 3, (a) NN contribution 1st excursion, (b) PD contribution 1st excursion, (c) NN contribution 30th excursion, (d) PD contribution 30th excursion.

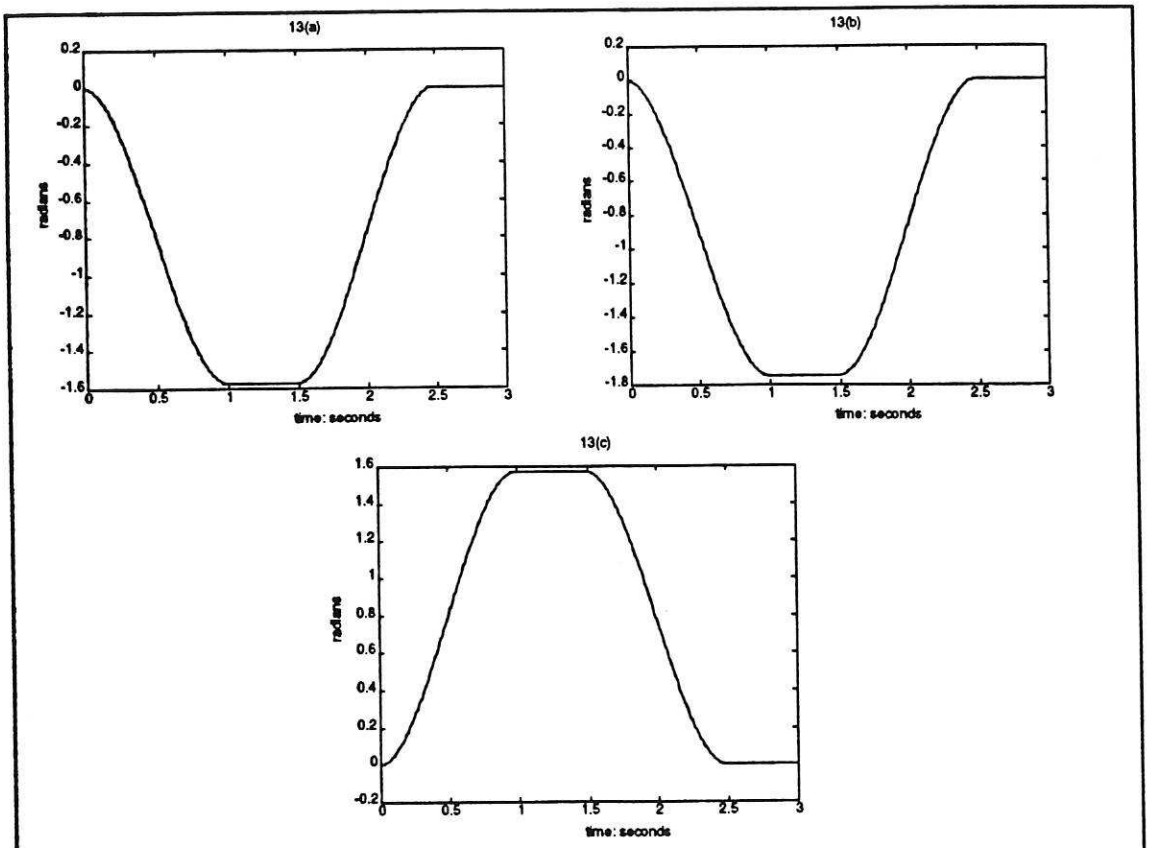

Figure 13. Nonlinear feedback control response for known dynamics, (a) joint 1, (b) joint 2, (c) joint 3.

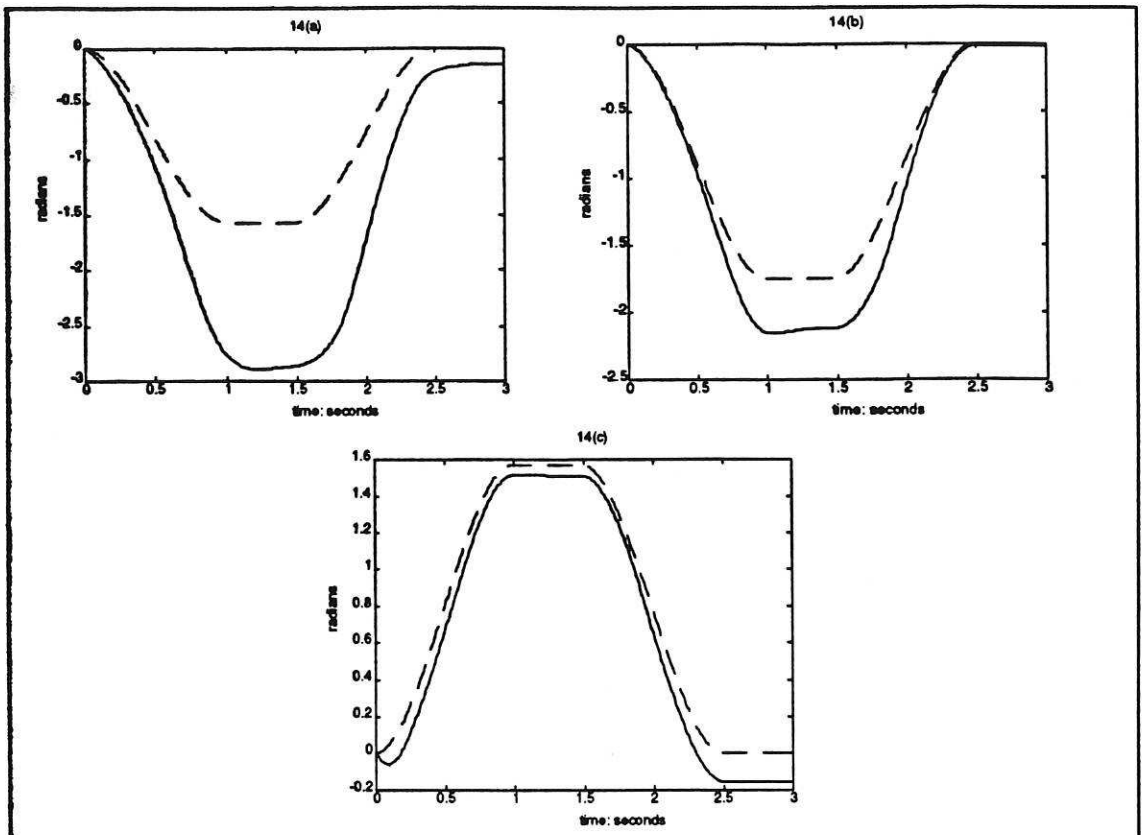

Figure 14. Nonlinear feedback control response for unknown joint friction and payload. (a) joint 1, (b) joint 2, (c) joint 3.

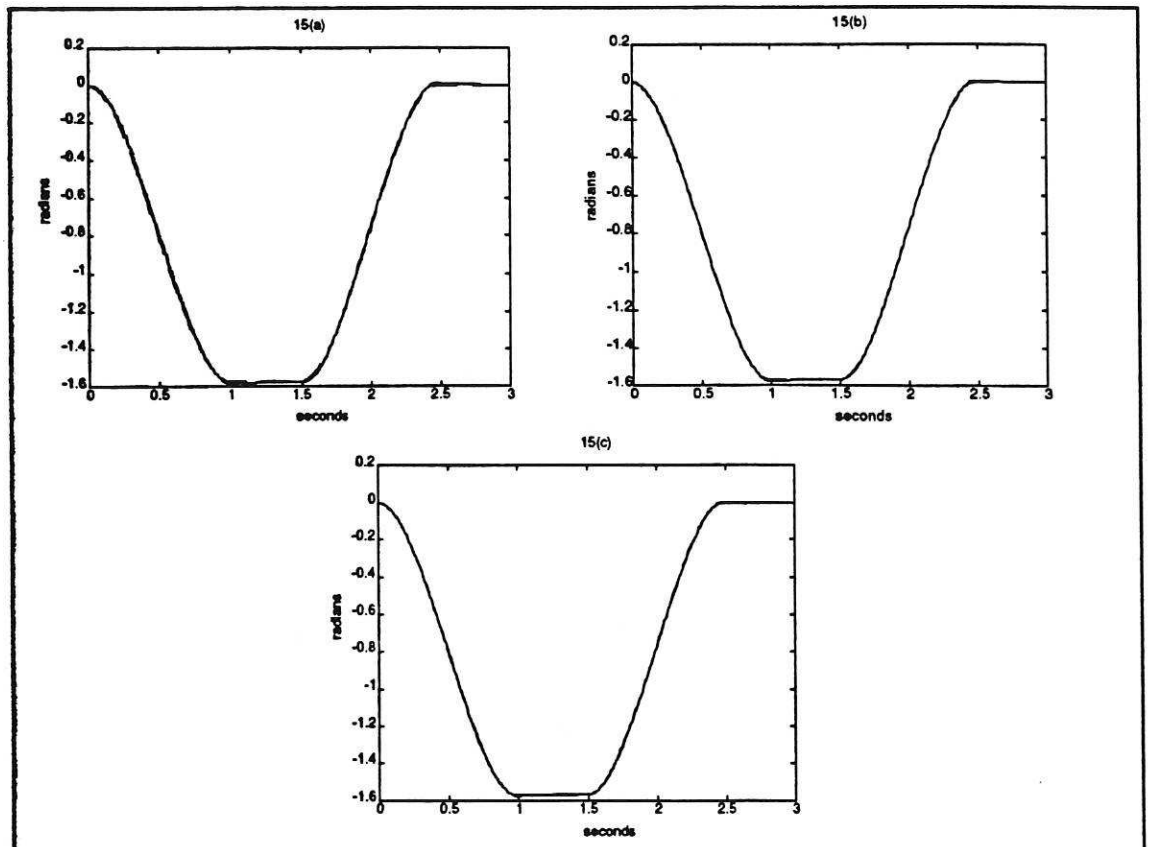

Figure 15. NN control response of joint 1 for variable payloads. (a) 0 Kg., (b) 5.5 Kg., (c) 9.5 Kg.

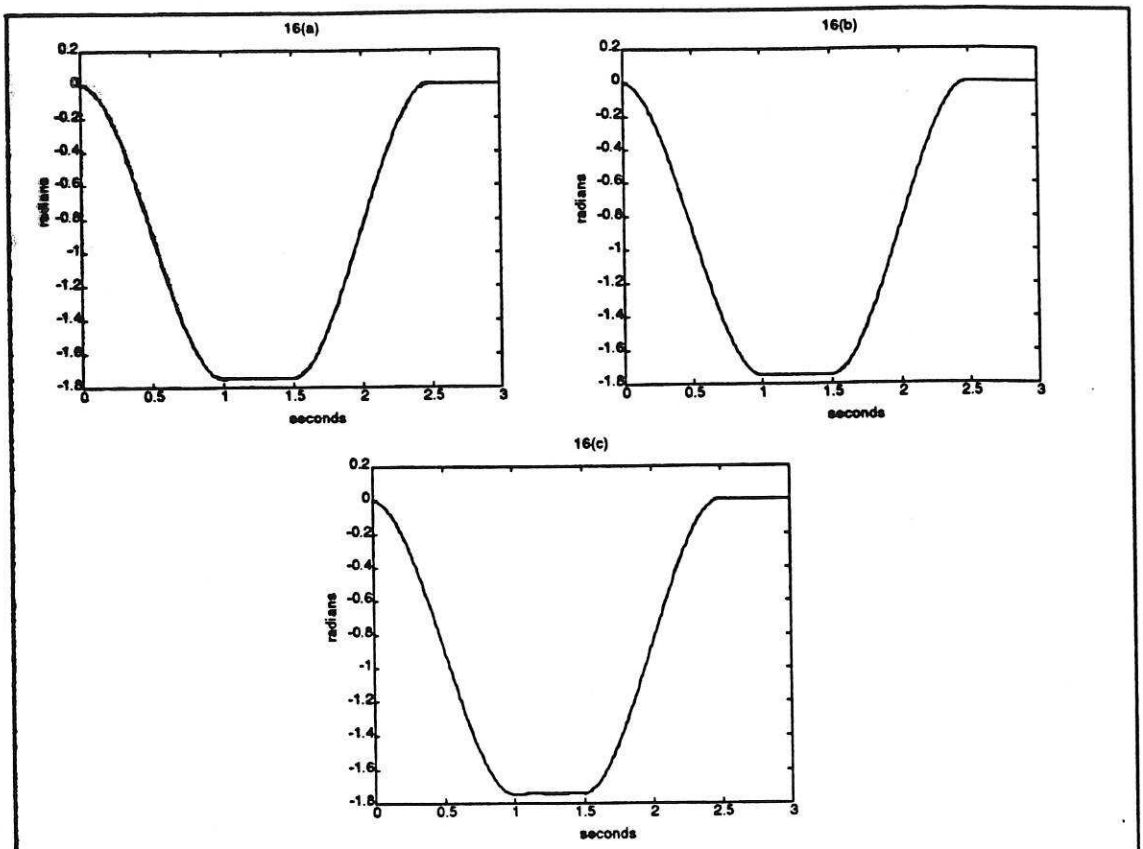

Figure 16. NN control response of joint 2 for variable payloads. (a) 0 Kg., (b) 5.5 Kg., (c) 9.5 Kg.

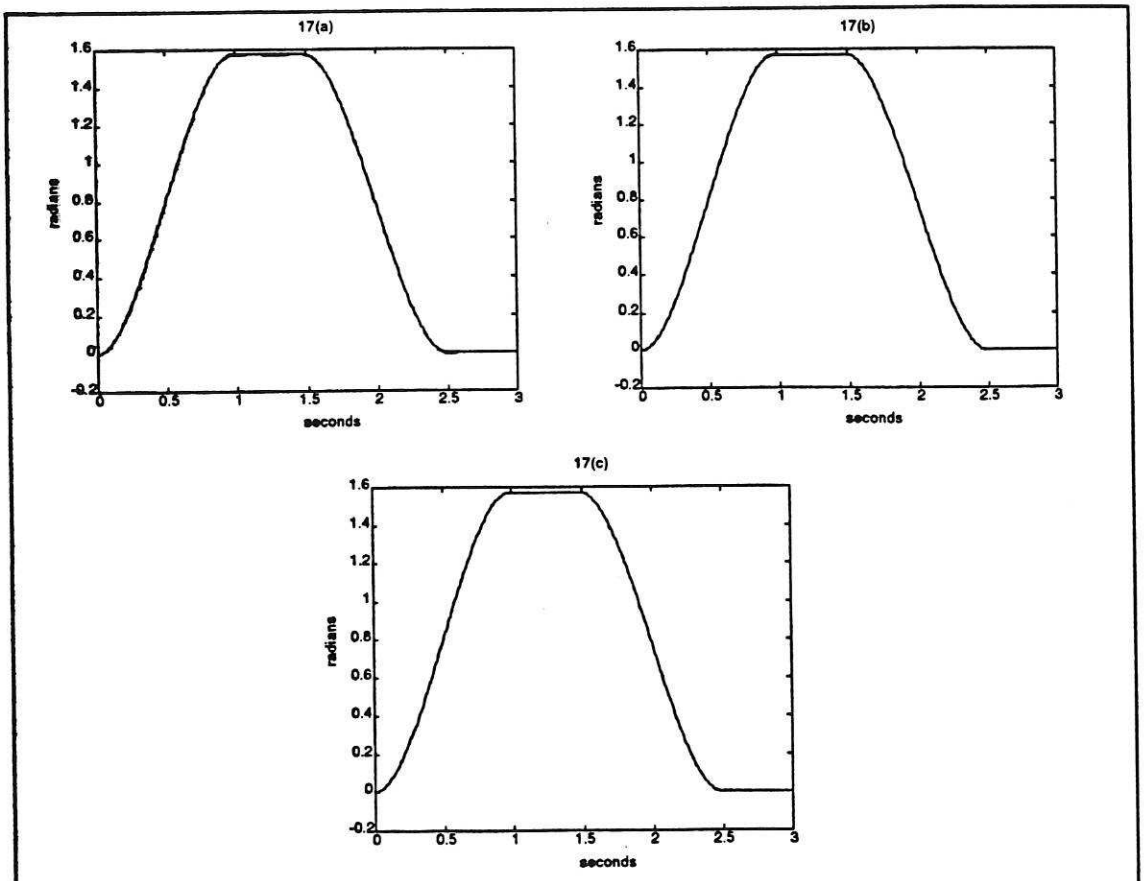

Figure 17. NN control response of joint 3 for variable payloads. (a) 0 Kg., (b) 5.5 Kg., (c) 9.5 Kg.

5. Conclusions

In this report a new decentralised neural network based strategy for accurate tracking control of robotic manipulators, in the presence of partially known dynamics, has been presented. Gaussian radial basis function controllers using direct adaptive techniques have been effectively used to compensate for the uncertainties of the system. The scheme does not require off-line neural net training sessions nor does it require any force feedback to determine the unknown payloads and is computationally efficient and inherently parallel. Simulation results show that the control scheme produces fast convergence of errors and is robust to variable payloads. The formulation of this control scheme can be easily extended to a general manipulator having any number of links.

References

1) Chen F. C., Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control. *IEEE Control Systems Magazine*, Vol. 10 No. 3, (1990).
2) Chen F. C. and Khalil H. K., Adaptive control of nonlinear systems using neural networks. *Int. J. Control*, Vol. 55 No. 6, (1992).
3) Freund E., Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators. *Int. J. Robotics Research*, Vol. 1, No. 1, (1982).
4) Ishiguro A., Furuhashi T., Okuma S. and Uchikawa Y., A neural networks compensator for uncertainties of robotics manipulators. *IEEE Trans. Industrial Electronics*, Vol. 39, No. 6, (1992).
5) Kasahara H., "Parallel processing of robot arm dynamic control computation on multiprocessors," *Microprocessors and Microsystems*, Vol. 14 No. 1, (1990).
6) Kawato M., Uno Y., Isobe M. and Suzuki R., Hierarchical neural network model for voluntary movement with application to robotics. *IEEE Cotrol Sys. Magazine*, pp. 8-15, (April 1988).
7) Lathrop R. H., "Parallism in arms and legs," MIT, M. Sc. thesis, (1982).
8) Leahy M. B., Johnson M. A. and Rogers S. K., Neural network payload estimation for adaptive robot control. *IEEE Trans. Neural Networks*, Vol. 2, No. 1, (1991).
9) Lee T. H. and Tan W. K., Real time parallel adaptive neural network control for nonlinear servomechanisms - an approach using direct adaptive techniques. *Mechatronics*, Vol. 3, No. 6, (1993).
10) Li W. and Slotine J. J. E., Neural network control for unknown nonlinear systems, *Proc. Amer. Contr. Conf*, pp. 1136-1141, (1989).
11) Poggio T. and Girosi F., Networks for approximation and learning. *Proc. of IEEE*, Vol. 78, No. 9, (1990).
12) Psaltis. D, Sideris A. and Yamamura A. A., A multilayered neural network controller. *IEEE Control Systems Magazine*, pp. 17-21, (April 1988).
13) Sanner R. M. and Slotine J. J. E., Gaussian networks for direct adaptive control. IEEE Trans. *Neural Networks*, Vol. 3, No. 6, (1992).
14) Sanner R. M. and Slotine J. J. E., Stable adaptive control and recursive identification using radial gaussian networks. *IEEE conference on decision and control*, Vol. 3, pp. 2116-2123, (1991).
15) Sastry S. S. and Isidori A., Adaptive control of linearizable systems. *IEEE Trans. Auto. Control*, Vol. 34, No. 11, (1989).

16) Slotine J. J. E. and Gunter N., "Performance in adaptive manipulator control," *Int. J. Robotics Research*, Vol. 10 No. 2, (1991).
17) Slotine J. J. E. and Li W., "On the adaptive control of robot manipulators," *Int. J. Robotics Research*, Vol. 6 No. 3, (1987).
18) Tarn T. J., Bejczy A. K., Han S. and Yun X., Dynamic equations for puma 560 robot arm. Robotics Lab. Report SSM-RL-85-02, Washington University, St. Louis, Missouri, (1985).
19) Tarn T. J., Bejczy A. K., Han S. and Yun X., Inertia parameters of puma 560 robot arm. Robotics Lab. Report SSM-RL-85-02, Washington University, St. Louis, Missouri, (1985).
20) Widrow B. and Lehr M. A., 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proc. IEEE*, Vol. 78, No. 9, (1990).
21) Zalzala A. M. S. and Morris A. S., A neural network approach to adaptive robot control. *Int. J. Neural Networks*, Vol. 2, No. 1, (1991).
22) Ziauddin S. M. and Zalzala A. M. S., Parallel processing for real time adaptive robot control : Theoretical issues and practical implementation. *IEE colloquium on high performance applications of parallel archetectures*, (Feb. 1994).
23) Zomaya A. Y. and Nabham T. M., Centralised and decentralised neuro-adaptive robot controllers. *Neural Networks*, Vol. 6, pp. 223-244, (1993).

Appendix

Brief review of Gaussian radial basis function neural networks

A Gaussian radial basis function network with n inputs and m outputs implements a mapping $f: \Re^n \rightarrow \Re^m$ according to

$$f(\bar{x}) = \sum_{i=1}^{N} \bar{\Lambda}_i g(\|\bar{x} - \bar{\mu}_i\|)$$

where $\bar{x} \in \Re^n$ is the network input vector; $\mu_i \in \Re^n$ i = 1, ..., N are the radial basis function centres; $\bar{\Lambda}_i \in \Re^m$ i = 1, ..., N are the weight vectors; $\|\cdot\|$ denotes the Euclidean norm; and $g(\cdot)$ is the Gaussian radial basis function given by

$$g(v) = \exp(-v^2 / \sigma^2)$$

where σ^2 is called the variance of the function, its value essentially defines the extent of localisation of the effects of the input. At the input the input space is divided into grids with a Gaussian function at each node having μ_i as its centre. At the output there is a linear node calculating the weighted sum of the Gaussian outputs.

Given any function that is sufficiently regular it is known that an approximation can be provided by the Gaussian radial basis function. Details can be found in [11].