EXPERIMENTS IN ON-LINE LEARNING NEURO-CONTROL

A. G. Pipe, M. Randall, Y. Jin

Intelligent Autonomous Systems Laboratory, Faculty of Engineering
University of the West of England, Bristol BS16 1QY, UK
Anthony.Pipe@uwe.ac.uk

Abstract

We apply our on-line learning neural network approach to control of non-linear systems; first to joint level trajectory control of an industrial welding robot, and then to one front leg of a hexapod walking robot. The first application demonstrates the ability of neuro-control to model complex non-linear components of a plant, thereby yielding impressive improvements in accuracy compared to the original robot's controller. The second application illustrates the strength of an on-line learning approach in coping with disturbances to a plant's characteristics. In order to set this work in context we briefly review our on-line learning neuro-control method, used for both sets of experiments. It has a strict theoretical basis including guarantees of the whole system's stability.

Introduction

The overall aim of the work described here is to illustrate the capabilities of an on-line learning neural network approach to improving the performance of controllers operating in complex non-linear domains. To achieve this aim one must be able to design systems that can improve their performance by modelling crucial components of a given plant directly using the data normally available as part of the control process; e.g. control effort, desired and actual set-points, their derivatives and so on.

Off-line techniques suffer from some limitations. If only simulation is used for training, then a sufficiently complex model is required, which could probably have been used to build an acceptable traditional controller. Alternatively, if real plant data are used, then the learned transfer function is limited to that information which has been acquired during the data gathering phase. An *on-line* approach, wherein the processes of data gathering and training occur simultaneously with control of the real plant, can overcome these problems. However, such an approach is not without difficulties. If the controller characteristics are changing due to on-line adaptation as the real plant is exercised, then it becomes imperative to provide guarantees of learning algorithm **convergence** and **stability** of the whole system.

The new work presented in this paper is concerned with the application of these techniques to one 3-jointed front leg of a hexapod walking robot. However, to set the context, we must first review our previous work on the Yaskawa robot and the neuro-control laws we use.

Related Work

Over the years much research effort has been put into the design of neural network applications for manipulator control. Albus [1] used the Cerebellum Model Articulation Controller (CMAC) to control manipulators as early as 1975. Miller et al [11] and Kraft et al [7] extended Albus' results and developed neural network learning algorithms in 1987 and 1992 respectively. Kawato et al [6] added MLP networks to original manipulator PD control systems as feedforward compensators in 1988. Iiguni et al [3] combined manipulator linear optimal control techniques with Multi-Layer Perceptron (MLP) neural networks, which were used to compensate for the non-linear uncertainty in 1991. Others have concentrated on linking neural network based approaches to control with learning convergence and stability guarantees. Examples are Narendra [12, 13] in 1991 and 1992, Sanner & Slotine [14] in 1991, Suykens, Van de Wall & De Moor [15] in 1995, and Lewis & Parthasarathy [8, 9] in 1996 and 1999.

The Yaskawa Robot Application

Since 1994 IAS Laboratory researchers have been working on on-line learning neuro-control and its applications. The team first concentrated on establishing the necessary stability theory. After this, implementations were demonstrated for improving trajectory tracking accuracy on a simulated Puma 560 robot manipulator and on a real educational Mentor robot [4, 5]. More recently we have extended the experimental work to cover a real Yaskawa Motaman robot [2], extracted from daily use as a manufacturing welding robot in a local company.

One could consider the three major axes (base, shoulder, elbow) of a manipulator as a coupled system of links and revolute joints, driven by electric motors. The Yaskawa is fitted with a shaft encoder and tachometer on each of these joints so as to provide position and velocity feedback.

A general equation of motion for a rigid manipulator is

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \bar{g}(q) + F(q,\dot{q}) = \tau(q,\dot{q},\ddot{q}) \quad \textbf{(1)}$$

where $\boldsymbol{q}$ is the n x 1 vector of joint displacements, $\boldsymbol{\tau}$ is the n x 1 vector of applied joint torques, $H(q)$ is the n x n symmetric positive definite manipulator inertia matrix, $\mathbf{C}(q,\dot{q})\dot{q}$ is the n x 1 vector of centripetal and Coriolis torques, $\bar{g}(q)$ is the n x 1 vector of gravitational torques, and $F(q,\dot{q})$ are the unstructured uncertainties of the dynamics including friction and other disturbances.

On-Line Learning for Neural Networks

Although the static performance of the PID controllers commonly used in such situations is good, the dynamic performance leaves much to be desired. By exploiting the neural network universal approximation feature we can assume that the left hand side of the above equation can be approximated by a neural network, i.e.

$$H\ddot{q} + C\dot{q} + \bar{g} + F = G(q,\dot{q},\ddot{q})^T W + \xi(q,\dot{q},\ddot{q}), \quad \textbf{(2)}$$

where $\boldsymbol{G}$ is the non-linear mapping matrix of an RBF or CMAC neural network, $\boldsymbol{W}$ is its weight vector, which is initially unknown, and ξ is its approximation error, which should be as small as possible by careful design.

The main objective of on-line learning is to force the manipulator to follow the desired trajectory, i.e. $\boldsymbol{q}_d$. In order to simplify the final control structure, we will use the desired joint values as the neural network inputs, however the actual joint values are used to train the neural network on-line.

During on-line learning, the neural network is used as a part of the manipulator controller. Its output forms part of the control signals which are used to drive manipulator joints. The joint values in turn are used to train the neural network. Therefore the on-line learning algorithm must guarantee that these two coupled systems (neural network & manipulator) work in a converging fashion. Our main theoretical research result is presented in the following theorem. The structure combines a simple two-term linear controller with the neuro-controller. The control law guarantees the asymptotic stability of the whole system (neural network and manipulator).

Theorem: Consider equation 1 with the neuro-control law

$$\tau = \mathrm{K}_p\tilde{\mathrm{q}} + \mathrm{K}_v\dot{\tilde{\mathrm{q}}} + \mathrm{G}(\mathrm{q}_d,\dot{\mathrm{q}}_d,\ddot{\mathrm{q}}_d)^T\hat{\mathrm{W}} \quad \textbf{(3)}$$

and the on-line learning algorithm

$$\dot{\hat{\mathrm{W}}} = \Gamma\mathrm{G}(\mathrm{q}_d,\dot{\mathrm{q}}_d,\ddot{\mathrm{q}}_d)(\dot{\tilde{\mathrm{q}}} + c\tilde{\mathrm{q}}) \quad \textbf{(4)}$$

where $\mathbf{q}$ is actual position and $\mathbf{q}_d$ is desired position, $\boldsymbol{\Gamma}$ is a constant positive definite matrix and is called the learning rate. If $\mathbf{K}_p$ and $\mathbf{K}_v$ are positive definite matrices and sufficiently large, and c is a small enough positive constant, then the closed-loop manipulator system is asymptotically stable and

$$\lim_{t\to\infty}\mathrm{q} = \mathrm{q}_d\,,\ \lim_{t\to\infty}\dot{\mathrm{q}} = \dot{\mathrm{q}}_d\,.$$

Proof: The proof is rather complicated. We only present the outline here. Those interested in further details are referred to [5].

It is well known in the robotics field that,

i) the inertia matrix, $\boldsymbol{H}$, is positive definite,

ii) for one choice of $\boldsymbol{C}$, $\dot{\mathrm{H}} = \frac{1}{2}(\mathrm{C} + \mathrm{C}^T)$,

iii) $\boldsymbol{H}$ and $\bar{\mathrm{g}}$ have bounded norms,

iv) $\boldsymbol{H}$ and $\bar{\mathrm{g}}$ only contain trigonometric function of $\boldsymbol{q}$ hence their partial derivatives with respect to $\boldsymbol{q}$ also have bounded norms.

By extending the work of Wen and Bayard [16] it can be proved that the following control law results in exponential stability if $\boldsymbol{K}_p$ and $\boldsymbol{K}_v$ are sufficiently large.

$$\tau = \mathrm{K}_p\tilde{\mathrm{q}} + \mathrm{K}_v\dot{\tilde{\mathrm{q}}} + \bar{g}(q_d) + F(q_d,\dot{q}_d) + C(q_d,\dot{q}_d)\dot{q}_d + H(q_d)\ddot{q}_d \quad \textbf{(5)}$$

The Lyapunov method [10] is used to prove this result. The Lyapunov function is

$$V = \frac{1}{2}\dot{\tilde{\mathrm{q}}}^T\mathrm{H}\dot{\tilde{\mathrm{q}}} + \frac{1}{2}\tilde{\mathrm{q}}^T\mathrm{K}_p\tilde{\mathrm{q}} + c\,\tilde{\mathrm{q}}^T\mathrm{H}\dot{\tilde{\mathrm{q}}} + \frac{1}{2}c\,\tilde{\mathrm{q}}^T\mathrm{K}_v\tilde{\mathrm{q}} \quad \textbf{(6)}$$

The control law of Equation 3 is the control law of Equation 5 with an additional term $-\mathrm{G}(\mathrm{q}_d,\dot{\mathrm{q}}_d,\ddot{\mathrm{q}}_d)^T\tilde{\mathrm{W}}$, where $\tilde{\mathrm{W}} = \mathrm{W} - \hat{\mathrm{W}}$. This additional term will produce additional terms in the first time derivative of the Lyapunov function. These additional terms are cancelled by the on-line learning algorithms if we use a new Lyapunov function, i.e.

$$V = \frac{1}{2}\dot{\tilde{\mathrm{q}}}^T\mathrm{H}\dot{\tilde{\mathrm{q}}} + \frac{1}{2}\tilde{\mathrm{q}}^T\mathrm{K}_p\tilde{\mathrm{q}} + c\,\tilde{\mathrm{q}}^T\mathrm{H}\dot{\tilde{\mathrm{q}}} + \frac{1}{2}c\,\tilde{\mathrm{q}}^T\mathrm{K}_v\tilde{\mathrm{q}} + \frac{1}{2}\tilde{\mathrm{W}}\Gamma^{-1}\tilde{\mathrm{W}} \quad \textbf{(7)}$$

Therefore the whole system is asymptotically stable. The control diagram is shown in Figure 1. The neural network

acts as a feedforward controller. The desired joint values (displacements, velocities, accelerations) are its inputs. In the feedback loop, there is a PD controller. A linear combination of joint displacement errors and velocity errors is used to train the neural network, as shown in figure 1.

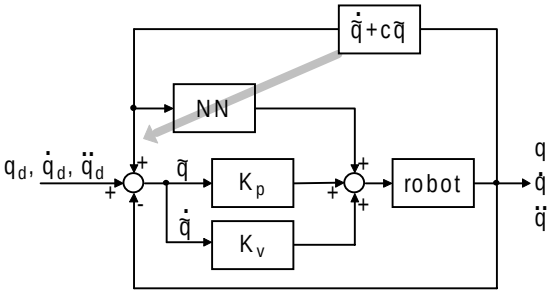

Figure 1: Robot manipulator neuro-control architecture

Remark: Although the theorem guarantees that the whole system will be asymptotically stable, the transient performance and the conditions which K_p, K_v and c must satisfy all depend on the initial Lyapunov function value. Unlike conventional adaptive control where the number of adjustable unknown parameters is small, the number of adjustable neural network weights is very large, so the initial Lyapunov function value will be very large. In order to reduce this initial value and improve on-line learning convergence velocity, it is often desirable to train the neural network off-line before it is used for on-line control.

Neuro-Control of the Yaskawa Manipulator

Three neuro-controllers are used, one for each of the three major axes. In the experimental approach adopted for the Yaskawa robot a CMAC (Cerebellum Model Articulation Controller) neural network [1] is effectively "strapped around" each existing joint controller rather than replacing it, as shown in figure 2. They are implemented via a PC-resident Texas Instruments DSP board, based around the 40MHz 320C40 processor, which executes the code for all three neural networks.

Each neural network utilises the same demand and feedback information available to the existing controller, and the output signals from the existing and neuro-controllers are combined by simple analogue voltage addition just before the power stages of the motor drive circuit. Each neural network therefore simply adds its control effort to that of the existing controller in order to help reduce errors.

The control arrangement for one joint is as defined by equations 3 and 4 above, and depicted in figure 1. Each composite joint controller receives new position, velocity and acceleration demands every 2ms. The existing two term linear controller operates on only the position and velocity error signals to derive control actions. The neural network takes position, velocity and acceleration demands as inputs and is trained on-line using a combination of velocity and position errors. Each CMAC has 32,768 adjustable weights. For a particular input set, 120 weights are selected for each output. Between each 2 millisecond sampling period and the next, the 320C40 must therefore complete one pass of the learning algorithm and a forward pass of the neural network for each CMAC. Over repeated trials, performance improves; though the bulk of this is obtained in the first five trials of a given trajectory.

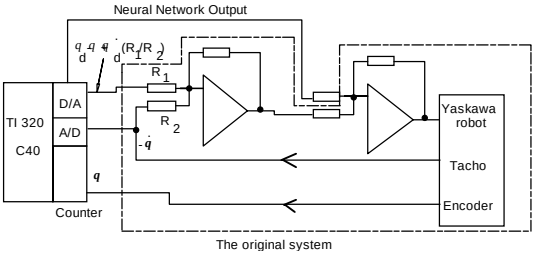

Figure 2: The Experiment Yaskawa Setting

Figure 3 shows an example desired trajectory which the robot should follow. The minimum ($q_{\min}$) and maximum ($q_{\max}$) joint values are -256 and 30720 counts of the joint position encoder for joint 1, -7168 and 8192 for joint 2, -5379 and 6144 for joint 3. 360 encoder counts roughly equal 1 degree.

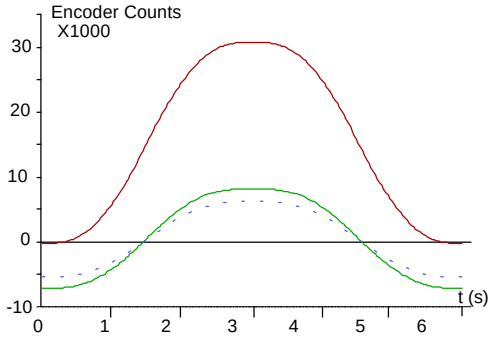

Figure 3: The Desired Trajectories (The solid line is for the Base Joint, the dashed line is for the Lower Arm Joint, and the dotted line is for the Upper Arm Joint)

Figure 4 shows the joint 1 trajectory tracking errors. Line (a) shows the errors without the neural network. The worst case error is 60 counts. Line (b) shows the errors of the neuro-controller in the first trial. The worst case is reduced to 46. Lines (c) and (d) show the errors in the third and fifth trials. Now the worst case errors are further reduced to 17 and 9 respectively. The tracking errors are finally reduced by more than six times. A similar improvement is also achieved to

joint 2 and joint 3. Their tracking errors are shown in figure 5 and figure 6 respectively.

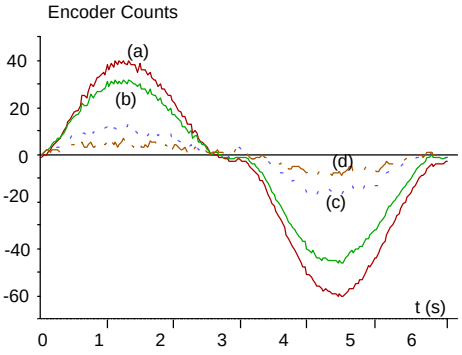

Figure 4: The Trajectory Following Errors of Joint 1. (a) Using the original controller; (b) Using the neuro-controller in the first trial; (c) Using the neuro-controller in the third trial; (d) Using the neuro-controller in the fifth trial

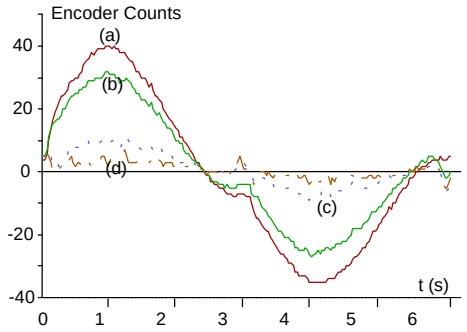

Figure 5: The Trajectory Following Errors of Joint 2. (a) Using the original controller, (b) Using the neuro-controller in the first trial, (c) Using the neuro-controller in the third trial, (d) Using the neuro-controller in the fifth trial

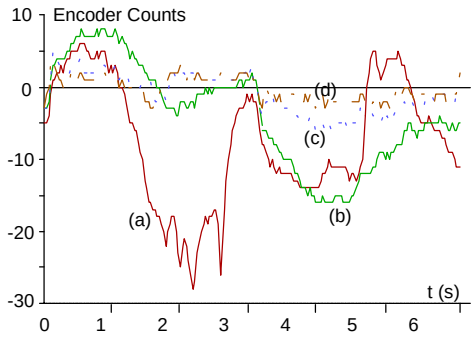

Figure 6: The Trajectory Following Errors of Joint 3. (a) Using the original controller; (b) Using the neuro-controller in the first trial; (c) Using the neuro-controller in the third trial; (d) Using the neuro-controller in the fifth trial

Neuro-Control of a Hexapod Leg Joint

We describe our initial results from the first phase of an investigation into on-line adaptive neuro-control of a hexapod walking robot's leg trajectories. In the intended application area of rugged terrain walking, very accurate trajectories and foot placement are required - even in the presence of disturbances. Our first experiments are concerned with the neuro-controller's performance in the face of intermittent end-effector force disturbances.

Below we identify a simplified neuro-control structure (relative to the Yaskawa experiments described above) which implements control of the coxa joint of one of the hexapod's front legs; in future work this will be extended to control of the other two joints of each leg, and then to all six legs. A Radial Basis Function (RBF) neural network is used as the controller [14].

In the Yaskawa robot experiments, the neural network contributed directly to control current supplied to each joint's motor. In the case here each motor is packaged as a servo-system, with on-board linear position control. A desired position is identified by Pulse-Position-Modulation (PPM). To generate a trajectory, desired positions are produced in sequence for each of the three servo-motors of a leg by the on-board computer (described below). In these experiments the RBF network is placed between this computer output and the servo-motor input, as detailed in the control diagram of figure 7.

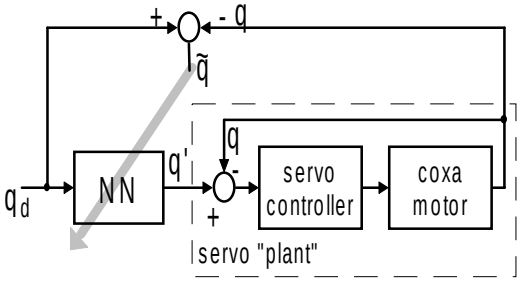

Figure 7: Leg joint neuro-control architecture

An error signal is derived from the position potentiometer already contained in the servo as part of the existing position control loop. This feedback signal is read into the on-board computer via an A/D converter and then used to derive an integer position error signal which is in turn used to train the neural network between one control action and the next. This generates a modified trajectory demand signal which should reduce trajectory tracking errors over time and adapt to control disturbances. In the Yaskawa experiments velocity feedback was used in addition to position (see equation 4). This will be required here also when the experimental platform is extended. For

the initial results presented here, however, only one desired trajectory, a curved forward movement of the leg, is considered and so velocity variations can be ignored.

Gaussian basis functions were used for the neural network, evenly spaced at intervals of 10 units across an integer desired position ($\mathbf{q_d}$) input range of 0 to 1000. Each basis function had a variance of 169.

The on-board computer is based around the Motorola 68332 microcontroller, clocked at a rate of 20MHz. The controller possesses an on-chip Time Processor Unit (TPU) which allows for conversion between integer values representing joint positions and a PPM signal to be sent to a servo-motor input. The controller also possess a Queued Serial Module (QSM) which provides for synchronous (SPI) and asynchronous (RS232) communications. The former is used for communication with some on-board peripheral devices, whilst the latter is used for transmitting the results of experiments to a host computer. The board is equipped with A/D converters, which are connected over the SPI line, for measuring the servo-motor position feedback signals. Software has been developed in the C language, using a PC-resident Microtec Research "XRAY" cross-development suite.

The RBF network is trained initially with an identity function so that, without on-line learning, it contributes no extra control effort. There are three plots on each graph; showing the desired position for the coxa joint, the servo-motor position feedback signal, and finally the RBF neural network output.

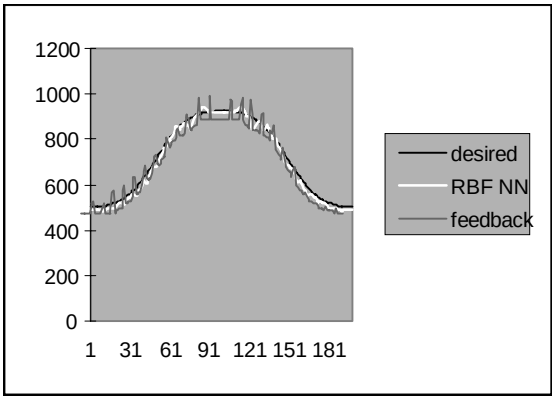

Figure 8: Initial off-load trajectory. Vertical axis: position in the integer units of the trajectory planning algorithm. Horizontal axis: time steps in units of approximately 30 ms.

Figure 8 shows the initial off-load case on the first run through the trajectory. It can be seen that all three signals are quite similar under these conditions, meaning that the servo-controller is performing quite well under these circumstances and the RBF network is unnecessary.

A spring was attached to the leg end-effector after the first run, which applied a variable tangential load disturbance to the system. Figure 9 shows the situation at the fourth cycle of the trajectory. One can clearly see that the RBF neural network has learned to compensate for tracking inaccuracies over the preceding three cycles, producing a modified trajectory demand which is substantially different from the original so as to give the required position feedback signal.

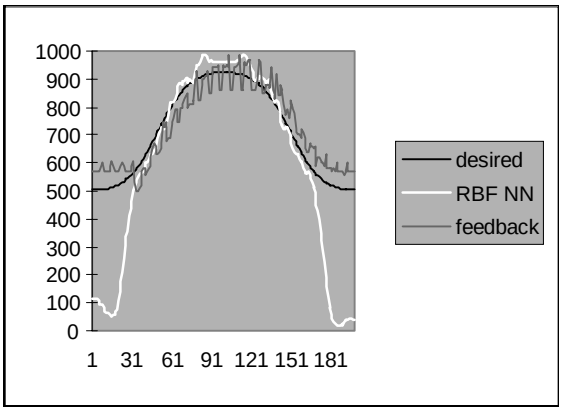

Figure 9: Trajectory three cycles after load applied. Vertical axis: position in the integer units of the trajectory planning algorithm. Horizontal axis: time steps in units of approximately 30 ms.

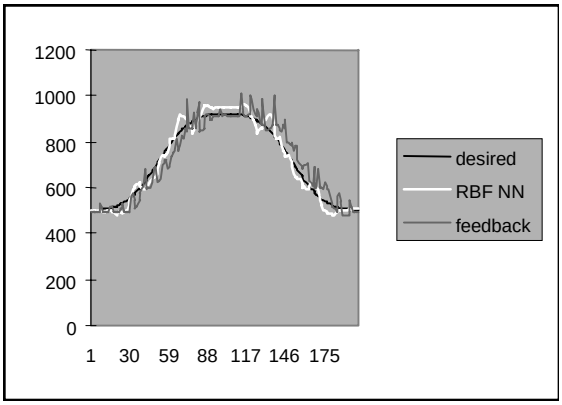

Figure 10: Recovered off-load trajectory. Vertical axis: position in the integer units of the trajectory planning algorithm. Horizontal axis: time steps in units of approximately 30 ms.

After this the spring was removed. This produced a significant initial trajectory overshoot. However by the seventh cycle the situation was nearly recovered to the original situation, as shown in figure 10.

Discussion & Conclusions

For the work carried out on the Yaskawa manipulator it was possible to achieve a six- to ten-fold reduction in joint level trajectory tracking errors on each of the principal three axes of movement compared with using the Yaskawa's existing joint controllers on their own. Recently these experimental results have been extended by testing the composite controller in many more parts of the manipulator's operational envelope, enabling an empirical analysis of its learning and generalisation performance across many trajectories. The results of this work have been encouraging, including recommendations for enhancement of the controller implementation [2].

The work on the hexapod walking robot is at a much earlier stage. There are a number of problems to overcome. Perhaps most significant is the noise generated by the potentiometer based feedback signal from the servo-system, this can be seen clearly in all the diagrams above. This requires either complete replacement, or perhaps some filtering.

However, despite these difficulties, the two sets of experimental results described in this paper indicate that an adaptive on-line neuro-control approach is not only capable of yielding accurate control performance when applied to complex non-linear plant, it can also provide useful adaptive responses to time varying dynamic load conditions. Although there was not space to print them in full here, these algorithms are based on thorough analyses of system convergence and stability, that we have summarised here and published fully elsewhere [4, 5].

References

[1] Albus, J.S., 1975, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)", *Journal of Dynamics Systems, Measurement, & Control*, Vol.97, pp.220-227.

[2] Cherian, R. P., 1997, *On-Line Learning Neuro-Control of an Industrial Manipulator*, MSc Thesis, Engineering Faculty, *Univ. of the West of England.*.

[3] Iiguni, Y., Sakai, H., 1991, and Tokumaru, H., "A Nonlinear Regulator Design in the Presence of System Uncertainties Using Multilayered Neural Networks", *IEEE Trans. on Neural Networks*, Vol.2, pp.410-417.

[4] Jin Y., Pipe A. G., Winfield A., 1997, "Chapter 5: Stable Manipulator Trajectory Control Using Neural Networks", *Neural Systems for Robotics*, Eds. Omid Omidvar & Patrick van der Smagt, Academic Press, ISBN 0-12-526280-9.

[5] Jin Y., 1994, *Intelligent Neural Control and Its Applications to Robotics*, PhD Thesis, Engineering Faculty, *Univ. of the West of England.*

[6] Kawato, M., Uno, Y., Isobe, M., and Suzuki, R., 1988, "Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics", *IEEE Control Sys. Magazine*, Vol.8, No.2, pp.8-15.

[7] Kraft, L.G., Miller, W.T. and Dietz, D., 1992, "Development and Application of CMAC Neural Network-Based Control", *Handbook of Intelligent Control, Neural, Fuzzy, and Adaptive Approaches*, eds: D.A.White and D.A.Sofge, New York: Multiscience Press Inc., pp.215-232.

[8] Lewis, F. W. and Jagannathan, S., 1996, *Neural Network Control of Robot Manipulators and Non-linear Systems*, Taylor & Francis books.

[9] Lewis, F. W., Jagannathan, S. and Yesildirek A., 1999, *Neural Network Control of Robot Manipulators and Non-linear Systems*, Taylor & Francis books.

[10] Lyapunov, A. M., 1992, *The General Problem of the Stability of Motion*, Taylor & Francis books.

[11] Miller III, W.T., Glanz, F.H., Kraft III, L.G., 1987, "Application of a General Learning Algorithm to the Control of Robotic Manipulators", *Int.Journal of Robotics Research*, Vol.6, No.2 pp.84-98.

[12] Narendra, K. S. and Parthasarathy K., 1990, "Identification and control of dynamical systems using neural networks", *IEEE Trans. Neural Networks*, vol. 1, pp.4-27.

[13] Narendra, K. S., 1992, "Adaptive Control of dynamical systems using neural networks", in *Handbook of Intelligent Control*, ed. White, D. A. and Sofge, D. A., New York: Van Nostrand Reinhold, pp.141-183.

[14] Sanner, R.M. and Slotine, J.J.E., 1992, "Gaussian Networks for Direct Adaptive Control", *IEEE Trans. on Neural Networks*, Vol.3, pp.837-863.

[15] Suykens, Van de Wall & De Moor, 1995, *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*, Kluwer Academic Publishers.

[16] Wen J.T., Bayard D.S., 1988, "New Class of Control Laws for Robotic Manipulators, Part 1: Non-Adaptive Case", *International Journal of Control*, Vol.47, pp.1361-1385.