DYNAMIC CONTROL OF A ROBOT ARM USING CMAC NEURAL NETWORKS

G. Cembrano, G. Wells, J. Sardá and A. Ruggeri

Instituto de Robótica e Informática Industrial, Univ. Politécnica de Cataluña- Consejo Superior de Investigaciones Científicas, Gran Capitán 2-4, Planta 2, Barcelona 08034, Spain (gcembrano@iri.upc.es)

Abstract: Neural identification and control techniques are well-suited to the problem of controlling robot dynamics. This paper describes the use of CMAC networks for the adaptive dynamic control of an orange-harvesting robot. Among the various neural-network paradigms available, the CMAC model was chosen in this case because of its fast convergence and on-line adaptation capability. The solution of this dynamic control problem with CMAC is an encouraging demonstration of "experience-based", as opposed to model-based, control techniques and is a good example of the use of on-line learning in adaptive neural control

Keywords: Neural networks, identification, adaptive control, robotics, CMAC

1. INTRODUCTION

The problem of dynamic control of a robot arm consists of generating the appropriate motor commands at the joints so that the end-effector follows a desired trajectory as precisely as possible, even under extreme speed and payload conditions. The efficient solution of the dynamic control problem using conventional control schemes would require a thorough knowledge of the system behaviour, translated into a very accurate nonlinear mathematical model, which is typically very hard to obtain (Zomaya and Nabhan, 1993). The effects of friction and inertia, for example, depend on the system state, and the addition of payloads to the system may greatly affect the overall dynamic behaviour.

In this context, neural identification and control techniques are very well-suited to the problem of controlling robot dynamics. Firstly, the ability to learn nonlinear behaviours through the presentation of appropriate examples of inputs and outputs helps to overcome the modelling difficulties. Secondly, the on-line learning capabilities of neural networks make them very efficient in adaptive control schemes.

A general approach to neural adaptive control, proposed in (Narendra and Parthasarathy, 1990), consists of an indirect model-reference adaptive scheme with a series-parallel structure. Similarly, several concepts related to neural adaptive control are treated in (Hunt *et al.*, 1993) and (Cembrano and Wells, 1992). A number of more refined versions of backpropagation have been proposed in the literature for adaptive control and proved in a variety of applications e.g. (Tzirkel-Hancock and Fallside, 1992; Hoskins *et al.*, 1992; Sbarbaro-Hofer *et al.*, 1993; Loke and Cembrano, 1994). In addition, the CMAC neural network (Cerebellar Model Articulation Controller), originally developed by (Albus, 1975), has also been used in neural adaptive control schemes. It has the advantage of much faster convergence than backpropagation networks, and excellent model-tracking capabilities (Ananthraman and Garg, 1993). Other researchers (Miller *et al.*, 1987; Miller, 1994) studied

a CMAC-based learning control system for the dynamic control of robot manipulators with multiple feedback sensors and multiple command variables. In this scheme, a CMAC network is used to adaptively learn an approximate dynamic model of the controlled robot in appropriate regions of the system state space. The CMAC learns the unknown nonlinear mapping between the sensor outputs and the system command variables from on-line observations of each during system operation.

This paper describes the use of CMAC networks for the adaptive dynamic control of an orange harvesting robot. The problem derives from the ESPRIT III project "Robot Control based on Neural Network Systems", which aimed to demonstrate the advantages of neural networks in several areas of robot control. One of the industrial applications in this project was a custom-made prototype developed by the Italian research institute Consorzio per la Ricerca dell'Agricoltura del Mezzoggiorno (CRAM) for orange harvesting, which must perform very fast movements under strict requirements of positioning accuracy. The use of CMAC for this problem is described in the paper.

The modelling and controller design of the robot arm were carried out in two phases. In the first stage, the prototype robot was not yet operative and no field data were available for generating a model of the dynamics, so the development of the controller had to rely on a theoretical model supplied by the manufacturer. In the second stage, when measurements could be taken with the actual robot, a thorough set of experimental arm elongation data was provided by the manufacturer, and a new model of the dynamics was developed by the authors. A neural-network model was preferred, since it provided better approximation capabilities than the theoretical model.

The first stage of the controller design provided some important results. It was useful for demonstrating the ability of CMAC to control a very complex dynamic behaviour similar to the robot dynamics. Despite the differences between the theoretical model and the real behaviour of the robot, the results of this stage showed that CMAC can efficiently approximate the implicit relationship between the end-effector trajectory and the motor commands, adapting the neural representation on-line. This result had the reasonable implication that CMAC will later be able to learn the real robot dynamics, since the neural structure is at all times independent of the robot model that generated the training data; it merely emulates an input-output relationship. Furthermore, at this stage, the characteristics of the CMAC learning and on-line adaptation processes were studied in depth, several control structures were designed and tested, and a preliminary tuning of the CMAC parameters for the problem was performed.

In the second stage, a neural identifier of the robot was used in place of the system model, within the same control scheme as in the first stage. A PID controller was used to provide nominal system control, and thus generate initial training data for the CMAC network. It was shown that the CMAC control scheme can achieve significantly improved control performance over the linear controller alone, by virtue of using CMAC network learning, to relate the observations of the tracking error to the controls being applied.

The development of the CMAC controller using data generated by the theoretical model are described in Sections 4 and 5. Section 6 deals with the application of the CMAC controller in a simulated scheme incorporating the experimentally obtained neural model of the robot arm. The conclusions of the work, and future extensions, are presented in Section 7.

2. THE CMAC NEURAL NETWORK

The CMAC network, based on the cerebellar model of neuromuscular control, is basically a nonlinear table-look-up technique which maps each N-dimensional input state-space vector to a corresponding output vector of the same or a different dimension. Each input vector activates exactly C overlapping input receptive fields, where C is a variable parameter representing the extent of generalization desired within the state space. The potentially very large virtual state space is mapped to a smaller physical weight table using a fixed random hashing table. Output values are computed simply as the sum of the C weights addressed by a given set of inputs. A supervised training method, resembling the Widrow-Hoff rule, is used to adjust the CMAC memory values, based on observations of the output error.

In the CMAC network, it is the input weight-addressing scheme which is nonlinear, whereas the output is a linear sum of weights. This characteristic provides a smooth error surface, thus improving convergence to a global minimum. Figure 1 shows a single point learned for an arbitrary bidimensional function, with a generalization factor of $C = 8$. The pyramidal shape indicates that the surrounding inputs produce output values which decrease linearly with distance. All sampled (learned) points of a function define an output surface formed by overlapping pyramids (a hypersurface for more than 2 inputs). For higher generalization factors, the pyramid spreads over a wider area, and it becomes narrower for smaller

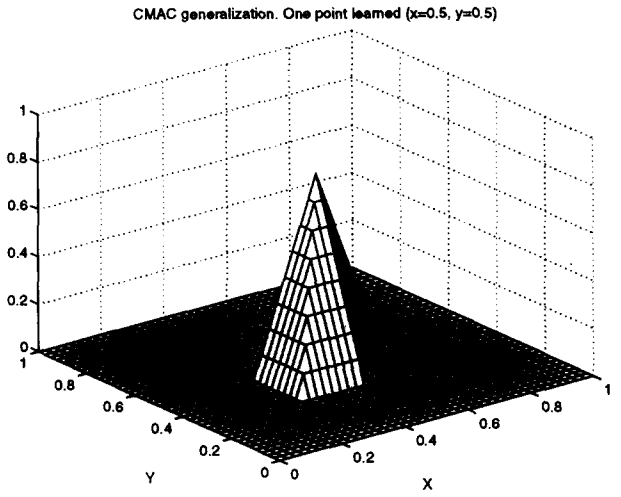

Fig. 1. Output surface after learning a single point (x= 0.5 and y= 0.5) of an arbitrary two-dimensional function. The scale was chosen so as to aid visualization of the generalization effect. $C = 8$

values. For $C = 0$, the CMAC memory acts as a linear look-up table. Conversely, a very large generalization factor would produce a behaviour resembling that of a feedforward backpropagation network.

In the CMAC network, the learning process and the neuron activations are local, whereas in a feedforward network (FFN), all the neurons are involved in the computation of an output. This fact makes CMAC much faster than FFNs, and capable of adaptation in specific regions of the input-output space, without requiring complete retraining. However, CMAC may, in some cases, fail to achieve the good generalization properties of FFNs, especially when very small C factors are used. In general, the C factor must be chosen as a compromise between the fast, local activation of a very few neurons with little capabilitiy of generalization, and the slower activation of a larger number of neurons with a good generalization behaviour.

3. PROBLEM DESCRIPTION

The orange-harvesting robot consists of a large 3-dof hydraulic arm which holds two smaller 3-dof telescopic harvesting arms with a cutting end-effector for the harvesting operation. A camera is mounted on each harvesting arm. The larger arm is heavy and slow and was conceived mainly for approximating the position of the harvesting arms to the correct height on the tree. The harvesting arms are light-weight structures which must perform very fast movements under strict precision requirements, even with high accelerations. The control problem dealt with in this work concerns the telescopic movement of the light-weight harvesting arm (a diagram of the arm is shown in Fig. 2).

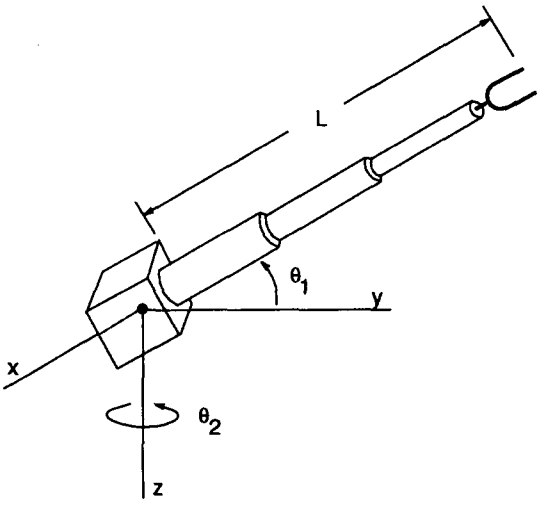

Fig. 2. Schematic diagram of the 3-dof telescopic harvesting arm.

3.1 *Theoretical model of the robot arm*

This model was a variable-structure, piecewise-linear approximation with four second-order linear transfer functions which represent the dynamic behaviour of the arm under different conditions of elongation and speed. This scheme is derived from the fact that no single linear model could accurately represent the whole range of dynamic system responses, so the system output was obtained from the most appropriate model in the current region of the state space.

The variable-structure model introduces a complex nonlinear switching behaviour, which is a very difficult problem for classical control techniques. The model takes account of variations in the inertia moments and friction coefficients depending on the elongation and speed of the arm. The prototype incorporates a programmable PID controller which provides the torque signals to the harvesting-arm actuators in a closed-loop control system. A scheme of the robot model is shown in Fig. 3.

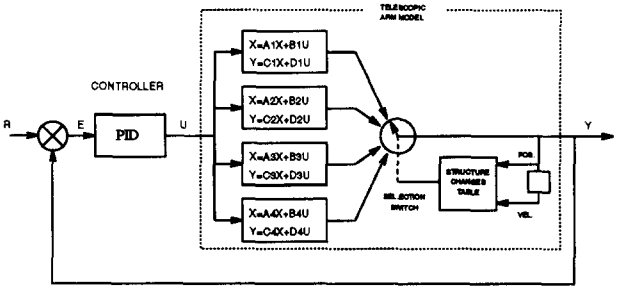

Fig. 3. Variable-structure telescopic robot model.

4. CONTROLLER DESIGN

4.1 *Linear control and reference signal*

Since the system would otherwise produce unstable responses, a PID controller is connected in

parallel with the CMAC controller, thus providing nominal system control in order to generate the initial training data for the CMAC. The controller was designed as a classical PID, tuned in order to ensure the best possible response for all of the four linear models, and it produces a good closed-loop tracking response for a complex reference trajectory.

The shape of the reference trajectory was designed in order to test the controller performance under the worst conditions. The robot's response is good, but shows some lag for step inputs, and instability in regions where the system structure changes rapidly. The graphs in Fig. 4 show the reference trajectory, the dynamics of the structural changes, and their influence on the control and output signals. This highly nonlinear behaviour must be learned by the CMAC adaptive controller. Figure 5 shows a detail of the switching behaviour.

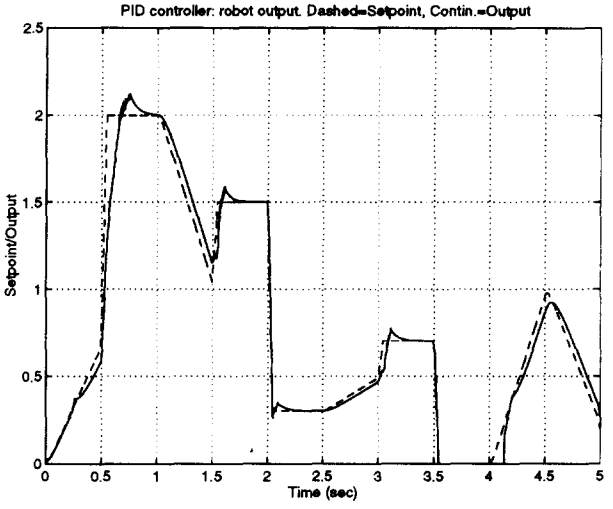

Fig. 4. PID tracking

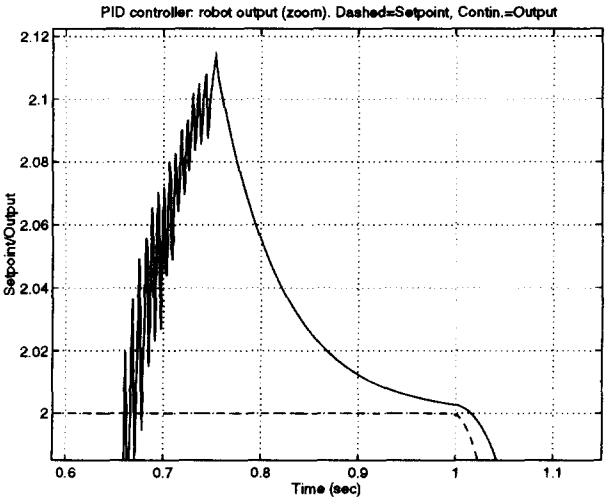

Fig. 5. PID tracking (detail showing complex switching behaviour). The ripple is not due to added noise, but is the effect of dynamic structural changes, which must be learned for good tracking performance.

4.2 *CMAC control scheme*

The CMAC network is used in a closed-loop control system to adaptively learn the inverse dynamics of the robot, thus predicting the actuator torques required to make the robot follow a desired trajectory. These torques are a function of the current errors in the joint position and velocity, and are used as feedforward terms in parallel with a fixed-gain PID linear feedback controller. The control signal input to the robot is the sum of the terms from the CMAC module and the feedback controller. The control scheme, based on that used by Miller (Miller *et al.*, 1987), is shown in Fig. 6.

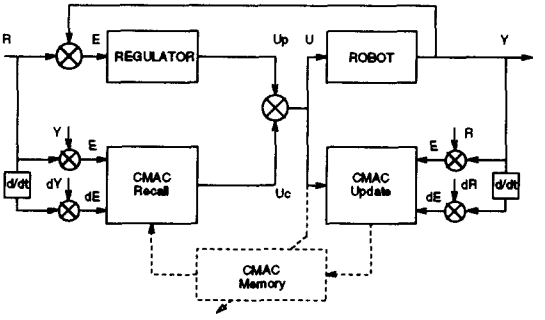

Fig. 6. CMAC control scheme.

Training is performed on-line in real time, which is possible due to the localized nature of the CMAC receptive fields and the speed of the CMAC algorithm. At the beginning, the CMAC memory is initialized to zero values. The PID controller therefore provides all of the initial robot control signals, producing the robot movements which generate the initial training data for the CMAC. After each control cycle, the CMAC weights are updated, based on the current errors in position and velocity. The error between the actual control signal input to the robot (the sum of the PID and CMAC signals), and the control signal the CMAC produces when given the current errors as inputs, is used in the Widrow-Hoff rule to adapt the weights. Thus, the CMAC is trained on the inverse dynamic response of the robot. As training progresses, the CMAC output signal improves, and the resulting control error decreases, diminishing the control effort required by the PID controller and increasing that of the CMAC. The CMAC therefore progressively takes over control from the PID. As will be shown later, the CMAC controller eventually provides a better control signal than the PID, by learning directly from the measurements of applied controls and the trajectory-tracking performance.

5. CMAC LEARNING OF THE INVERSE DYNAMICS

5.1 *Internal mapping*

The internal mapping learned by the CMAC memory is very much dependent on the shape of the reference signal and the generalization factor. A representation of the CMAC output mapping for the first point learned at the beginning of the first training cycle has the shape of a cone, whose base has a width that depends on the magnitude of the generalization factor (QF), as was shown in Fig. 1. The subsequent points form a chain of overlapping cones which represents the inverse dynamics of the robot model as a series of ridges and valleys. Figure 7 illustrates how this represents the control signal for one complete training cycle, which is related to the evolution of the reference signal.

It must be pointed out that the initial topography resembles the shapes of the various portions of the reference signal (the initial error follows the setpoint because there is still no robot response for the first time step, and the PID is designed as dominantly proportional). In addition, the error surface is divided into a series of parallel ridges. Since structural changes depend on the position and velocity, it can be deduced that the separation between the different ridges in the topography is due to structural changes. The widths of the peaks and valleys depend on the generalization factor. For higher generalization values, the chain of cones forms a smoother topography, whose ridges and valleys fuse. Lower generalization produces sharper topographies, with more separation between the ridges. This characteristic is relevant to the variable structure problem. As may be observed in Fig. 7, the structure-changing behaviour is reflected in the number of ridges and valleys in the topography. With higher generalization factors, the generalization effect smoothes out the ridges and valleys, and with lower factors they become sharper and more separate, increasing the resolution of the control signal. In other words, part of the dynamics of the structural changes is lost with overgeneralization.

The effect on the memory topography of increasing the generalization factor is observed in Fig. 8. The robot output begins to diverge from the reference trajectory as the generalization factor is increased. The learning time also increases, due to the larger number of memory values which must be updated at each cycle. This is an undesirable effect for the real-time application. The value of the generalization parameter must therefore be chosen carefully.

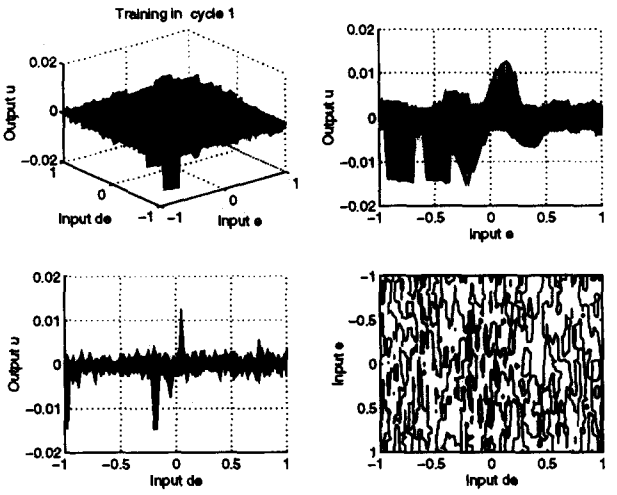

Fig. 7. CMAC memory mapping after one training cycle, with generalization factor $C = 8$. Labels e and de represent the error and error derivative. Label u is the control signal. The diagram shows a perspective view and projections on all 3 planes.

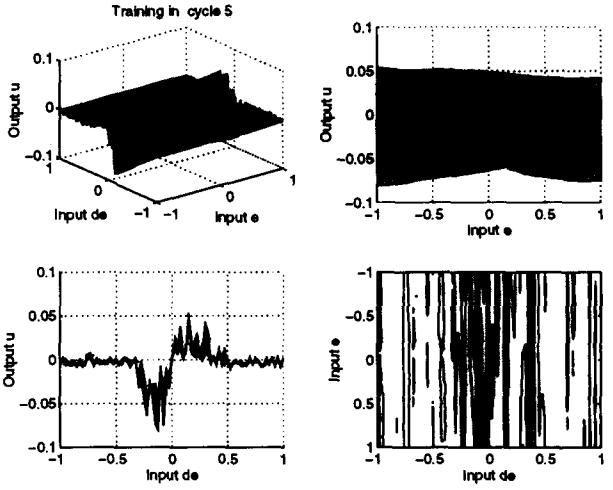

Fig. 8. CMAC memory mapping after the fifth training cycle with generalization factor $C = 128$. Labels e and de represent the error and error derivative. Label u is the control signal. The diagram shows a perspective view and projections on all 3 planes.

5.2 *On-line adaptation*

During the initial training cycles, the CMAC controller learns little more than the behaviour of the PID. In the CMAC memory topography, no further enlargement is observed in the peaks and valleys. Any deviation of the command that is not represented in the CMAC memory is balanced by the PID controller. Likewise, the inverse dynamic behaviour of the robot, that the PID could not adequately respond to, is learned and represented in the CMAC memory, thus supplementing the control signal.

As the number of on-line training cycles increases, the CMAC controller output contributes more and more to the overall control signal input to the robot, and the error surface changes accordingly. The process continues until a point of equilibrium

is reached, where both controllers contribute to the control signal. After this point, the CMAC controller's influence on the robot response is at least as strong as that of the PID, and it begins to learn the responses that result from applying its own control signal. The CMAC's behaviour becomes adaptive, directed by the feedback error signal, and improvement in the robot's response begins to be observed.

Figure 9 shows the results of CMAC control after 14 learning cycles. Despite abrupt changes in position imposed by the reference trajectory, the CMAC control scheme is able to track it very closely, even for the faster, almost instantaneous movements. The only significant deviations from the setpoint occur at the tops of the step portions of the trajectory, where many structural changes occur during a very short interval. Although this response was observed for both CMAC control and PID control, this problem is highly unlikely to arise with the real robot, since any structural variations it may display will occur in a more continuous manner. Even after the CMAC training has stabilized for this trajectory, some differences in performance may be observed from one cycle to the next. Nevertheless, tracking remains stable and accurate. The results of this test show that the CMAC can efficiently learn the inverse dynamics of this complex robot model and improve the results of applying classical PID control alone.

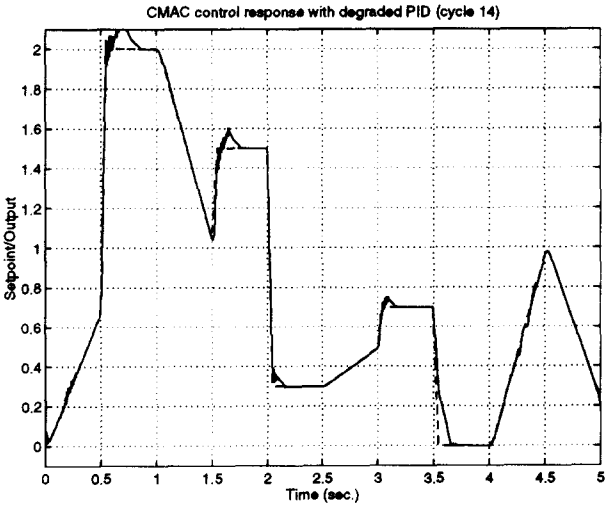

Fig. 9. CMAC Control vs. setpoint.

Due to the fact that the PID controller was optimally designed for the theoretical robot model, the PID control results are very good and the tracking improvement with the CMAC controller only slightly appreciable. However, it would be impossible to obtain such results in real conditions with the PID controller alone, where important deviations from the model would appear. In this case, the theoretically designed PID would not perform satisfactorily, and no explicit representation of the robot would be available, with which to redesign it. It is in those conditions that the ability of CMAC to learn from the experimental data of tracking error and applied control is especially useful, since it will learn to produce a better control signal than that of the model-dependent PID controller.

6. RESULTS OF CMAC CONTROL ON THE EXPERIMENTAL ARM MODEL

With a comprehensive set of measurements of the control, position and velocity signals in elongation and retraction movements provided by the manufacturer, a better simulator of the robot dynamics was developed. Due to the highly nonlinear nature of this system, a neural network model was preferred to the piecewise linear approximation used beforehand. The identification process was carried out according to the methods described in (Cembrano and Wells, 1992) and the best results were obtained with a 3-layer feedforward neural network with 5 inputs, 1 output and 8 hidden nodes. The inputs are the current state of the system (position and velocity) and the current control action, as well as delayed observations of these signals. The output is the next position of the system. Figure 10 shows an example of the identification results with this neural network for an arbitrary elongation movement of the arm. The plot represents the elongation of the end-effector vs. time, with the identifier and with the real system. A more detailed description of the identification process and its results on several robot movements may be found in (Cembrano and Wells, 1995)

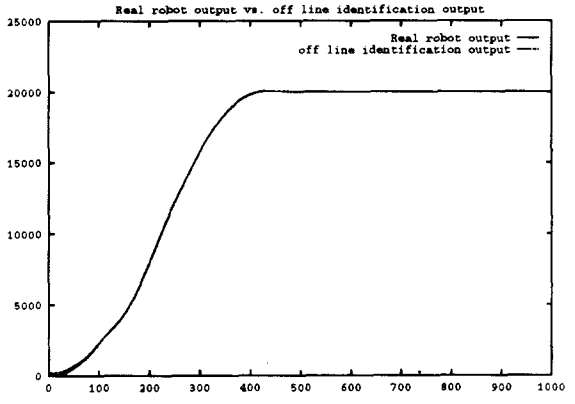

Fig. 10. Identification results for an elongation movement using the neural identifier.

The neural identifier was incorporated in the CMAC control scheme in place of the linear model. In these conditions, the PID controller was not optimal for the process, and any improvements on the PID parameters had to be performed on a trial-and-error basis. Figure 11 shows the control performance with the PID alone

for an elongation movement. The plot represents elongation vs. time for the observed and reference trajectories.

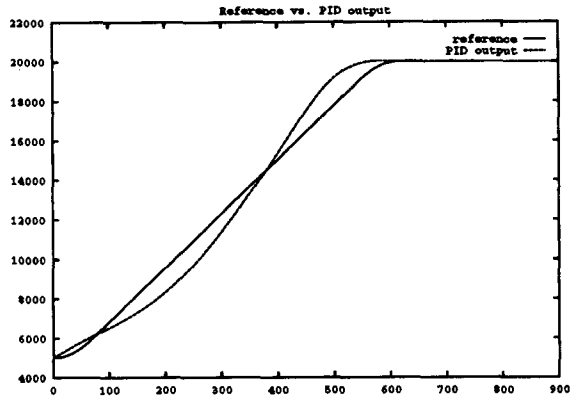

Fig. 11. PID control in an elongation movement. Elongation vs. time for reference and observed trajectories.

The CMAC controller was trained with the new robot model, as described in Section 6. After the 5th learning cycle, the CMAC controller had learned on-line to control the system, so that the combined control produced a significantly better tracking performance than the PID alone, as shown in Fig. 12.

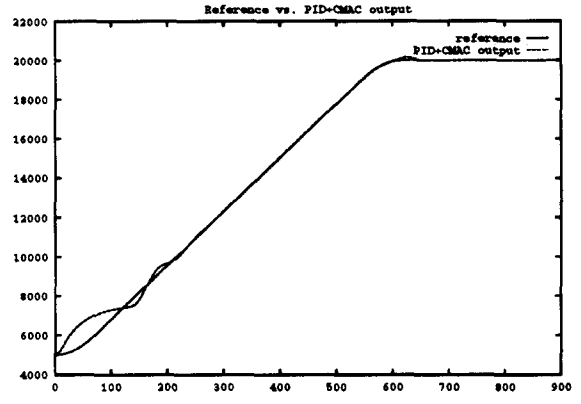

Fig. 12. CMAC Control for the same elongation movement as in Fig. 11.

These results illustrate the capabilities of the proposed CMAC control scheme to learn to control a very complex system, based on on-line observations of system performance. Even though a linear controller is initially required to provide the nominal control trajectories from which the CMAC can begin learning, the above results show that the trained CMAC controller can eventually outperform the PID controller. As long as the initial controller provides a stable output, its performance is not critical. This fact is important, since it implies that the initial controller design process is made simpler, and subsequent improvements can be achieved through learning.

7. CONCLUSIONS AND FUTURE WORK

The work described in this paper was carried out as part of a research project aimed at demonstrating the use of neural networks in several aspects of robot control. The control of robot dynamics under strict requirements of trajectory tracking of partially unknown or time-varying nonlinear systems has traditionally been a difficult problem in robotics. Therefore, this was considered a challeging problem for the use of neural networks.

Among the various neural-network paradigms available, the CMAC model was chosen for this dynamic control application because of its fast convergence and on-line adaptivity. The learning capabilities of the CMAC network for a variable-structure system have been analyzed and the results for an arbitrary tracking problem have been given. The results show that the CMAC control scheme used here can efficiently learn to control a complex nonlinear system and provide improved performance over that achieved with a PID optimally designed for this system.

The solution of this dynamic control problem with CMAC is an encouraging demonstration of "experience-based", as opposed to model-based, control techniques, and constitutes a good example of the use of on-line learning in adaptive neural control. Ongoing work in this research area is aimed at thoroughly validating the arm control scheme for subsequent hardware implementation.

ACKNOWLEDGEMENTS

This research work was partially funded by the ESPRIT III program of the EEC, under project No. 6715 "Robot Control based on Neural Network Systems". Research in neural control at the Instituto de Cibernética is partially supported by the research grant CICYT-TAP94-0552-C03-01 of the Spanish Science and Technology Council. Mr. Sardá was supported by CONICIT and BAUXILUM (Venezuela).

REFERENCES

Albus, J. (1975). A new approach to manipulator control: the Cerebellar Model Articulation Controller (CMAC). *Trans. of the ASME. Journal of Dynamic System, Measurement and Controls* **No. 97**, pp:220–227.

Ananthraman, S. and D. P. Garg (1993). Training backpropagation and CMAC neural networks for control of a SCARA robot. In: *Engineering Applications of Artificial Intelligence*. Pergamon Press.

Cembrano, G. and G. Wells (1992). Neural networks for control. In: *Application of Artificial Intelligence in Process Control* (R. A. Vingerhoeds L. Boullart, A. Krijgsman, Ed.). pp. 388–402. Pergamon Press.

Cembrano, G. and G. Wells (1995). Neural identification of the orange-harvesting robot. In: *Deliverable Report No. 4* (ESPRIT-III Project No. 6715, Ed.). Robot Control Based On Neural Network Systems.

Hoskins, D. A., J. N. Hwang and J. Vagners (1992). Iterative inversion of neural networks and its applications to adaptive control. *IEEE Trans. on Neural Networks*.

Hunt, K. J., D. Sbarbaro, R. Zbikowski and P. J. Gawthrop (1993). Neural networks for control systems - a survey. In: *Automatica*. Vol. 28(6). Pergamon Press.

Loke, R.E. and G. Cembrano (1994). Neural adaptive control of a bioreactor. In: *Intelligent Components and Instruments for Control Applications* (Cs. Bányász, Ed.). Vol. 2. pp. 182–187. John Wiley. Budapest, Hungary.

Miller, W.T., F.H. Glanz and L.G. Kraft (1987). Application of a general learning algorithm to the control of robotic manipulators. *Journal of Robotics Research* **6(2)**, 84–98.

Miller, W.T. (1994). Real-time neural network control of a biped walking robot. *IEEE Control Systems* **1(2)**, 41–48.

Narendra, K. S. and K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*.

Sbarbaro-Hofer, D., D. Neumerkel and K. Hunt (1993). Neural control of a steel rolling mill. *IEEE Control Systems*.

Tzirkel-Hancock, E. and F. Fallside (1992). Stable control of nonlinear systems using neural networks. In: *International Journal of Robust and Nonlinear Control*. Vol. 2. John Wiley.

Zomaya, A. Y. and T. M. Nabhan (1993). Centralized and decentralized neuro-adaptive robot controllers. *Neural Networks* **6**, pp:223–244.