REINFORCEMENT LEARNING CONTROL SCHEMES

Vilma Alves de Oliveira, Eduardo Fontoura Costa, Aluízio Fausto Ribeiro Araújo e Renato Tinós

Departamento de Engenharia Elétrica
Universidade de São Paulo
Caixa Postal 359, 13560-970, São Carlos, SP, Brazil

Abstract–In this paper, the use of Artificial Neural Network for the control of non-linear plants is explored. As the plant parameters or model is considered unknown, it is necessary to use plant input/output to train the controller and therefore reinforcement learning control schemes are devised to achieve desired results. The main features of the developed schemes are that few trials are required to train the controllers and a variety of control actions is taken rather than only two actions as in the standard reinforcement schemes. In addition, a supervised neural network controller, which is trained using the reinforcement control schemes, is proposed. An example of a magnetic suspension system is presented to illustrate the effectiveness of the control algorithms given. For comparison purposes, results of a linear optimal controller are included.

Key Words Intelligent controllers, Adaptive critics, Reinforcement learning, Neural networks.

1 INTRODUCTION

In general, conditions that assert the solution of non-linear systems control problems are not available, except in local terms, as in the neighborhood of a point. Also, design procedures work well only if a number of conditions are satisfied. The control problems become considerably more complex when the plant model or parameters are not completely known.

The use of reinforcement learning provides an interesting alternative to the control of systems (Milan, 1996; Dorigo and Colombetti, 1994; Costa *et alii*, 1997). This technique does not need plant parameters or plant model (Hoskins and. Himmelblau, 1992; Barto *et alii*, 1983) and facilitates the incorporation of practical constraints in the controller design. As a result, this technique may be applied to a large class of systems.

Reinforcement learning addresses the problem of an agent that must learn behavior through trial-and-error interactions in a dynamic environment (Kaelbling *et alii*, 1996). In control applications this involves determining a function through experience that maps current state of a plant into control actions and constructing a critic to judge which control actions are acceptable (success) and which are not (failure).

The objective of the presented approach is to construct a control system capable of driving the plant state to an equilibrium point and holding it at about that point. We present reinforcement control schemes based on the model-free Adaptive Critic strategy introduced by Barto *et alii* (1983) which has been applied to control systems since then (Millington and Baker, 1990; Shelton and Peterson, 1992; Sutton *et alii*, 1992; Brown and Chairs, 1994). The main features of the developed schemes are that few trials are required to train the controller and a variety of control actions is taken rather than only two actions. The schemes proposed led us to the development of two control algorithms with the incorporation of: (a) weight update direction sign and (b) temporal failure evaluation. A multilayer perceptron neural network is also proposed as a third control scheme.

This paper is organized as follows. The control problem is stated in Section 2. Two different reinforcement learning control schemes are presented in Section 3 and the multilayer perceptron neural network is presented in Section 4. In Section 5 we present results of the proposed control strategies and for comparison we include results of a linear optimal controller.

2 PROBLEM STATEMENT

This paper deals with the following class of problem: control of a dynamic nonlinear system, described by (1), such that the state and control input are maintained inside a region Δ established as acceptable for the system.

$$\dot{x}(t) = f[t, x(t), u(t)] \tag{1}$$

where $x \in R^n$ is the state, $u \in R$ is the control input, k denotes discrete time and f_k is a nonlinear function. The state is assumed to be available, the plant model is unknown and there are constraints on the state and amplitudes of the control input.

The region Δ is defined from the constraints on the state and control input and the desired stability and performance. The region Δ may be as complex as needed; for example, Δ may be simply the feasible region of the plant, or may be a function of time, as in the example of Section 5.1.2, in which Δ includes performance specification. In this way, control problems such as state regulation or tracking can be addressed by choosing an appropriate region Δ.

The control input u is given by adaptive elements (discussed later) which perform a mapping of the space state into control actions.

Throughout this paper we consider the following definitions. A failure state occurs when the system leaves the region Δ. A state x_0 is an equilibrium point of the system (1) if $f(t, x_0, u_0) = 0$ for a given u_0. A nonlinear system is autonomous if f_k does not depend on its first argument, k (Vidyasagar, 1993).

Artigo Submetido em 28/10/96
1a.Revisão em 28/02/97; 2a.revisão em 17/06/97; 3a. revisão em 25/09/97
Aceito sob recomendação do Ed.Cons. Prof.Dr. Fernando Gomide

3 REINFORCEMENT LEARNING CONTROL SCHEMES

There are two main classes of reinforcement learning algorithms: model-based algorithms and model-free algorithms. In the first one, the model-based algorithm learns a model and use it to derive a controller (Sutton, 1990a; Kumar and Varaya, 1986; Sutton, 1990b). In the second class, the model-free algorithms learn a controller but does not learn a model (Barto, 1983; Watkins, 1989; Dayan and Sejnowski, 1994; Sutton, 1988).

Model based algorithms demand great computational effort in order to obtain a reward function and a state transition function (Kaelbling *et alii*, 1996). Comparative studies suggest that among the model-free algorithms, the Barto, Sutton and Anderson algorithm (1983), BSA hereafter, has lower failure indices (Moriaty and Mükkulainen, 1996).

The main features of the BSA algorithm may be summarized as follows. The control signal is given by an adaptive search element (ASE) which is in charge of mapping the plant state in such a way that one out of two control actions happens, one for better and one for worse. This mapping is done by associating weights to partitions of the state space. The weights are changed by a rule which considers both a control action and a reinforcement signal in such a manner that decreases the probability of a control action that has happened at instants close to a failure occurrence. One can conclude that a failure inhibits the associated behavior. The reinforcement signal may be given by a measure of the plant performance.

The training strategy above is not suitable for the incorporation of a variety of control actions. Actually, a failure occurrence still indicates that the weights must be changed, however it does not indicate how, since more than two actions would be taken. For instance, a failure may indicate that a behavior must be stimulated rather than inhibited. We will show an illustrative example of this in Section 5.1.

In this paper fast learning schemes are devised because we use an on line learning controller with failures occurring during the training procedure, since the controller learns from experience. We consider two solutions to the learning problem. In the first one, the designer analyses the plant in order to provide a suitable reinforcement criterion. In the second one, the controller incorporates a temporal failure evaluation criterion.

Following, in Section 3.1 we describe the scheme obtained by using the first solution, which we call reinforcement learning control scheme with designer oriented reinforcement criterion (RSDO), pointing out where it differs from the BSA algorithm. In Section 3.2 we present the scheme obtained by using the second solution, which we call reinforcement learning control scheme with temporal failure evaluation (RSTFE).

3.1 Reinforcement Learning Control Scheme with Designer Oriented Reinforcement Criterion (RSDO)

The strategy of the control with designer oriented reinforcement criterion is as follows. The control problem is split into sub-problems by dividing the space state Ψ in a number of partitions. Each partition is associated with a specific control signal by adaptive elements. These elements have their weights adjusted by taking into consideration a reinforcement signal which indicates whether the system remains inside the region Δ or not and, in the RSDO, gives the correct signal of the weight adjustment. The adaptive elements are called adaptive search element (ASE) and associative critic element (ACE). A decoder is used to partition Ψ. When the system is in partition i the decoder output is equal to one and zero elsewhere, for i=1,...,i_T, with i_T being the total of partitions.

The control input $u(k)$ is given by the ASE output which is of the form

$$y(t) = f\left[\sum_{i=1}^{n} w_i(t)x_i(t) + noise(t)\right] \tag{2}$$

where k is the discrete time, w_i is the weight associated with the partition i, d_i is the decoder output, n_s is a real random variable with probability density function h, and f is either a threshold, sigmoid or identity function.

The weight update equation is written as

$$w_i(t+1) = w_i(t) + \alpha r(t) e_i(t) \tag{3}$$

where the positive constant α is the learning rate, $\hat{r}$ is the internal reinforcement signal, described below, and e_i is the eligibility term, which associates the responsibility of the weights of partition i with failure occurrence. In the BSA algorithm, the weight adjust depends on the eligibility which depends on the control action. As said above, such a rule works well when only two control actions are considered, otherwise it may be inappropriate. In this section we use a design oriented reinforcement criterion to set the weight adjustment rather than the control signal. Thus, the eligibility is given by (4).

$$e_i(k+1) = \delta e_i(t) + (1-\delta)x_i(t) \tag{4}$$

where δ, $0 \le \delta < 1$, is a decay rate.

The internal reinforcement signal, $\hat{r}$, is provided by the ACE, the objective of which is to predict a failure, allowing antecipative weight adjustment.

The ACE equations we use are the same as in the BSA algorithm and are presented here for easy reference.

$$p(k) = \sum_{i=1}^{i_T} v_i(k) d_i(k) \tag{5}$$

$$\hat{r}(k) = r(k) + \gamma p(k) - p(k-1) \tag{6}$$

$$v_i(k+1) = v_i(k) + \beta \hat{r}(k) \bar{e}_i(k) \tag{7}$$

$$\bar{e}_i(k+1) = \lambda \bar{e}_i(k) + (1-\lambda) d_i(k) \tag{8}$$

where p is the output of the ACE, v_i is the weight associated with partition i, $\bar{e}_i$ is analogous to the eligibility for the ACE, r is the reinforcement signal, γ and λ are real valued constants (where $0 < \gamma \le 1$ and $0 \le \lambda < 1$), and the positive constant β is the critic learning rate.

The scheme proposed here incorporates the sign (positive or negative) of weight update direction into the external reinforcement criterion. This means that the plant has to be analyzed in order to provide an appropriate criterion.

The steps of the learning algorithm are as follows, where t is defined as the trial number, t_{max} as the total number of trials and k_{max} as the total of sampling periods.

1. Initialize the ASE and ACE weights, set t=1;
2. While $t < t_{max}$, do:

2.1. $t=t+1$;
2.2. $k=0$, $r(k)=0$, initialize $x(k)$;
2.3. While $r(k)=0$ and $k < k_{max}$, do:
2.3.1. $k=k+1$;
2.3.2. Generate $d_i(k)$, $i=1,...,i_T$, by decodifying the actual system state vector $x(k)$;
2.3.3. Calculate the element outputs $u(k)$ (2) and $p(k)$ (5);
2.3.4. Present $u(k)$ to the system and obtain $x(k+1)$(1);
2.3.5. Calculate $e_i(k+1)$ (4) and $\bar{e}_i(k+1)$ (8), $i=1,...,i_T$;
2.3.6. Determine $r(k+1)$ using a defined criterion and $\hat{r}(k+1)$ (6);
2.3.7. Adjust the ASE weights $w_i(k+1)$ (3) and the ACE weights $v_i(k+1)$ (7);
2.4 If $k = k_{max}$, stop the algorithm.

Remark: There is no distinction between training and operation phases in the sense that in normal operation the controller keeps learning with the same algorithm used for training.

The control design may thus be summarized as follows. First, establish the region Δ in which the state must remain; this region may be simply the set of the feasible state or may incorporate performance specifications (as illustrated in the example). Then, choose the reinforcement criterion in order to provide the correct weight adjustment; the value of r must be zero when the state is inside Δ. After this, define the partitioning of the state space (this includes choosing i_T) and choose suitable functions h and f. Then, set the controller parameters $\alpha,\delta,\gamma,\beta$ and λ. Finally, set the parameters t_{max} and k_{max} and execute the learning algorithm.

The following comments are useful in the choice of the design parameters. A high number of partitions may lead to a high number of trials, on the other hand a low number may be insufficient. The parameter γ is related to how the difference between the current and previous reinforcement prediction affects the internal reinforcement (if γ approximates to one, $\hat{r}$ assumes rewarding values and if γ approximates zero, $\hat{r}$ assumes penalizing values). The learning rates α and β have similar characteristics: if they are too high, the weight adjustments are too fast and they never reach the correct value and if they are too low, the learning procedure becomes too slow. The eligibility decay rates δ and λ are also similar: higher values lead to weight adjustment relative to partitions in which the system stayed in a more remote past. The parameter t_{max} is chosen in order to guarantee enough trials for the controller to learn properly. Further details are found in Barto *et alii* (1983).

3.2 Reinforcement Learning Control Scheme with Temporal Failure Evaluation (RSTFE)

The previous control scheme may be modified to yield a more generic algorithm in which it is not necessary to analyze the plant in order to choose an appropriate reinforcement criterion. Thus, the issue that arises is how to determine the sign of the weight adjustment by simply observing its effects on the system performance.

Let us consider that if the failure time instant in the current trial is higher than that of the previous one the sign is correct, otherwise the sign must change. Hence the new weight update equation is:

$$w_i(k+1) = w_i(k) + \alpha\hat{r}(k)e_i(k)sg(t, k_f(t) - k_f(t-1)) \quad (9)$$

where

$$sg(t,v) = \begin{cases} sg(t-1,v), & \text{if } v \geq 0 \\ -sg(t-1,v), & \text{if } v < 0 \end{cases} \quad (10)$$

with $k_f(t)$ the instant k in which the failure occurs at trial t, and $sg(0,v)=1$.

Now, once the reinforcement signal does not need to give the correct weight adjustment signal, the reinforcement criterion may be of the form

$$r(k) = \begin{cases} -1, & \text{failure state} \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

The design procedure and the learning algorithm are basically the same as in the RSDO. The differences are that here the reinforcement criterion does not need to provide the correct weight adjustment and (3) in step 2.3.7 is replaced by (9).

4 CONTROL WITH SUPERVISED NEURAL CONTROLLER (SNC)

One of the main advantages of the proposed control schemes, as a result of the space state partitioning, is their fast learning. However, the use of partitioning leads to a control input which is discontinuous and varies abruptly. Also, as a specific control signal is associated with a entire region of Ψ, it is not possible to stabilize the system asymptotically, except for particular classes of systems. In this section, we take advantage of the interpolation capability of multilayer perceptron neural networks (MPNN) in order to overcome the problems described above.

There is a vast literature in Neural Network models and training (Hertz *et alii*, 1991; Muller *et alii*; Heskes and Wiegerinck, 1996) and they will not be presented here. The MPNN training may be on line or off line. In the former, the MPNN is trained simultaneously with the RSDO or the RSTFE according to Figure 1. In the latter, training and test sets are the state chosen from the feasible region and the corresponding control actions given by the RSDO or the RSTFE.

The operation of the SNC is summarized in the following algorithm where W is the weight matrix obtained in the training process, k_{max} is the same as for the RSDO or the RSTFE and the SNC output $u(k)$ is given by

$$u(k) = N(x(k)) \quad (12)$$

where the neural network is represented by the operator N.

1. Set the SNC weights equal to W.
2. $k=0$, initialize $x(k)$;

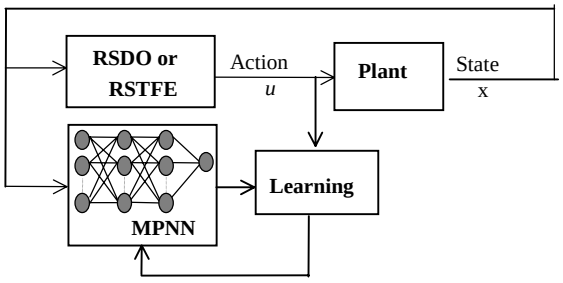

Figure 1. SNC supervised on line training.

3. While $k < k_{max}$, do:
3.1. $k=k+1$;
3.2. Calculate the SNC output $u(k)$ (12);
3.3. Present $u(k)$ to the system and obtain $x(k+1)$ (1);
3.4 If $k = k_{max}$, stop the algorithm.

5 MAGNETIC SUSPENSION SYSTEM EXAMPLE

The control problem is to maintain a magnetic suspension system, which is nonlinear, autonomous and intrinsically unstable, around an operational point. This system consists of a steel sphere kept in suspension by a magnetic field. This field is generated by a current circulating in a coil (Figure 2). The state variables are the sphere position (x_1), speed (x_2) and the coil current (x_3). The control action is the voltage on the coil terminals (u). Practical constraints together with the system discrete dynamic equations, used here to simulate the plant, are found in the Appendix.

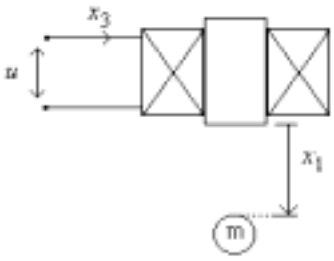

Figure 2. Magnetic suspension system with u denoting the input voltage.

The magnetic suspension system can be controlled using linear system techniques. The linear quadratic regulator (LQR) can be used with success to find the control law (u = -Kx) which minimizes a performance function with respect to the system linear dynamic equations. This controller achieves the optimal performance around the equilibrium point. However, the control system may become unstable when initialized in positions distant from the operational point.

The system dynamic performance is close to the predicted by the linearized system if it obeys the constraints inequality:

$$1-\varphi \le \frac{\mathbf{x}_j(k+1)-\mathbf{x}_j(k)}{\mathbf{x}_{jl}(k+1)-\mathbf{x}_{jl}(k)} \le 1+\varphi \qquad (13)$$

where x_{jl} is the state in the linearized model, and φ is a real valued constant. If inequality (13) is not satisfied, the difference between the behavior of the real system and the linearized one is significant, and the system may become unstable. Figure 3 shows a region θ_L with φ=0.2, for the magnetic suspension system.

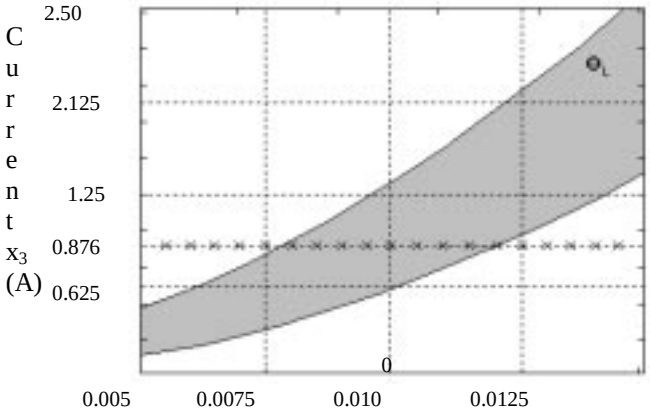

Figure 3. Region of state space with linear behavior for φ=0,2. The points marked by "x" correspond to the initial state of Figure 5.

5.1 Learning with the BSA algorithm

Before presenting the magnetic suspension system results obtained, we illustrate the learning difficulties associated with the reinforcement criterion mentioned in Section 3.

Figure 4 shows a portion of a learning phase in which the system response gets worse. The algorithm is the BSA with f being a saturation non-linearity. The control signal is scaled by $u_s=u$-17.4 such that its value is zero at the equilibrium point. When the failure occurs in trial 6, the control signal u_s is negative. Then the weights are adjusted in such a manner that decreases the probability of a negative control signal happening in partitions associated with this failure. In this way the control signal tends to be less negative and the system response tends to worsen. Here the failure indicates that the associated behavior must be stimulated rather than inhibited.

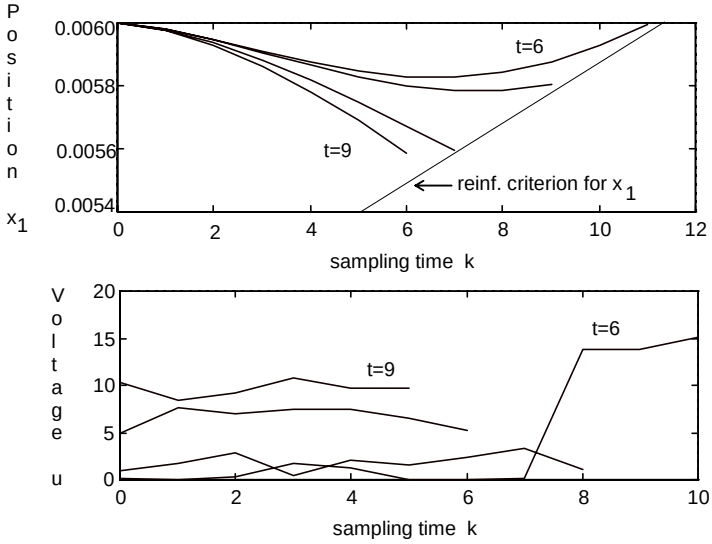

Figure 4. A portion of an execution of the BSA algorithm, showing that the system response gets worse from trial t=6 to trial t=9.

Numerous simulations were performed for different parameters, different output functions f, and also considering a different plant with no consistent learning success. Similar results to those displayed in Figure 4 were frequent in all these simulations.

5.2 Results

In this section we present simulation results for the developed control schemes.

5.2.1 LQR Results

Figure 5 shows simulation results for LQR design. As already mentioned, the system becomes unstable when initialized at positions distant from the operational point. The LQR parameters (weight matrices *Q* and *R*) are found in the Appendix.

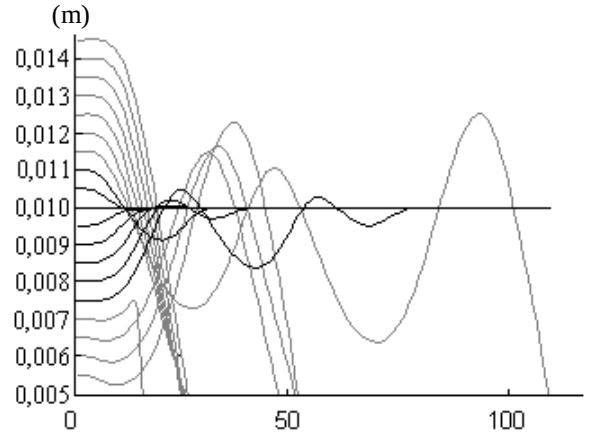

Figure 5. Sphere position for the optimal controller. Gray curves show initial state that result in instability. Black curves display stable dynamics.

5.2.2 RSDO and RSTFE Results

The adopted number of partitions of the state space Ψ is 288 for the first two schemes: 12 intervals for x_1, 6 for x_2, and 4 for x_3. The learning algorithm is executed for different initial state, as follows: we define a set of initial state $X_{ini} = \{x^1, \ldots, x^\mu\}$ where μ is the total number of initial state. In step 2 the system state are initialized as $x(0)=X_{ini}(n_t)$ and the learning algorithm is run for $n_t = 1,\ldots,\mu$. This procedure is repeated until $r(k)$ remains at zero for every learning algorithm execution with initial state $x(0) \in X_{\text{ini}}$. The typical number of trials for each initial state was 29 and 37 for the RSDO and RSTFE, respectively. The controller parameters are found in the Appendix.

For the RSDO and RSTFE the function *f* which gives the value of the controller output is given by (14).

$$f(h)=\begin{cases} u_{\max}, h \geq u_{\max} \\ h, u_{\min} < h < u_{\max} \\ u_{\min}, h \leq u_{\min} \end{cases} \tag{14}$$

The reinforcement criterion for the RSDO is developed by analyzing the magnetic suspension system and imposing a minimal acceptable performance. Equation (15) shows the reinforcement criterion. See also Figure 6.

$$r(k)=\begin{cases} 0 \text{ if } x \in \Delta \\ +1 \text{ if } x_1(k) > x_{1\sup}(k) \text{ or } x_2(k) > x_{2\sup}(k) \\ -1 \text{ if } x_1(k) < x_{1\inf}(k) \text{ or } x_2(k) < x_{2\inf}(k) \end{cases} \tag{15}$$

where

$$\Delta = \{x \in R^3 : x_{1\inf}(k) < x_1(k) < x_{1\sup}(k), x_{2\inf}(k) < x_2(k) < x_{2\sup}(k)\}$$

with

$$x_{j\inf}(k)=\begin{cases} [(x_{jc\min} - x_{ji\min})k / k_c] + x_{ji\min}, k < k_c \\ x_{jc\min}, k \geq k_c \end{cases}$$

$$x_{j\sup}(k)=\begin{cases} [(x_{jc\max} - x_{ji\max})k / k_c] + x_{ji\max}, k < k_c \\ x_{jc\max}, k \geq k_c \end{cases}$$

k_c is a sampling time chosen by the designer,

x_{jcmax} is the maximum x_j at sampling time k_c,

x_{jcmin} is the minimum x_j at sampling time k_c,

x_{jimax} is the maximum initial x_j,

x_{jimin} is the minimum initial x_j,

$j=1,\ldots,n_s$.

Figure 6 shows simulation results for the RSDO for different initial positions of the sphere. Note that this criterion is satisfied by the system.

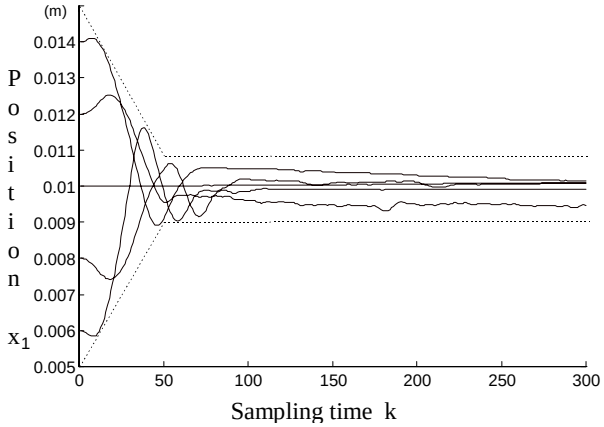

Figure 6. RSDO results for different initial positions. The reinforcement criterion is also shown (dashed line).

The reinforcement criterion for the RSTFE was developed by imposing a minimal acceptable performance. Equation (16) shows the reinforcement criterion.

$$r(k)=\begin{cases} 0 \text{ if } x \in \Delta \\ -1 \text{ otherwise} \end{cases} \tag{16}$$

where

$$\Delta = \{x \in R^3 : x_{1\inf}(k) < x_1(k) < x_{1\sup}(k), \; x_{2\inf}(k) < x_2(k) < x_{2\sup}(k)\}$$

with x_{1inf}, x_{1sup}, x_{2inf}, x_{2sup} are the same as in (15).

Figure 7 shows simulation results for the RSTFE for different initial positions of the sphere. Note that this criterion is satisfied by the system.

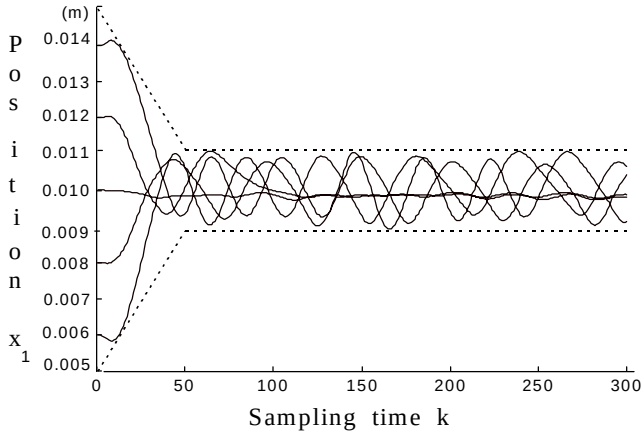

Figure 7. RSTFE results for different initial positions. The reinforcement criterion is also shown (dashed line).

In Hoskins and Himmelblau (1992), Moriarty and Mükkulainen (1996) and Anderson (1989) we can find control examples which make use of reinforcement learning to train neural networks. In these examples the number of trials to train the controller is more than five times the number of trials obtained here. Actually, in Hoskins and Himmelblau (1992) and Anderson (1989) this number increases to one hundred times. This may be explained by the fact that the decodification process we used here was not implemented there.

5.2.3 SNC Results

Here an off-line training for the SNC is used. The training algorithm is the Backpropagation with the Levenberg Marquardt method, which yields quadratic convergence (Bazaraa *et alii*, 1993). The training data set is chosen as follows. First, we mark the partitions which the system state has reached during reinforcement training and we obtain t_p, the total number of these partitions. Then, we generate n_t equally spaced state values per partition and we obtain (by using RSTFE or RSDO) the corresponding control actions. Finally, we form a training data set $X_a = \{x^1, \ldots, x^{n_t t_p}\}$ and $u_a = \{u(x^1), \ldots, u(x^{n_t t_p})\}$. Similarly, for the testing data set with n_s equally spaced state values per partition, we have $X_t = \{x^1, \ldots, x^{n_s t_p}\}$ and $u_t = \{u(x^1), \ldots, u(x^{n_s t_p})\}$. There are other ways to choose training and test data sets, as for example using a probability density function to generate state values, which leads to similar results.

The multilayer perceptron neural network used in the SNC scheme has three layer with 12, 6, and 1 units, respectively from the first to the last layer. The activation functions are hyperbolic tangent in the hidden layers and linear in the output layer. For the RSTFE we have $t_p = 118$; choosing $n_t = 6$ and $n_s = 4$ we have a training set with 1416 patterns and a test set with 944 patterns. The typical number of epochs is 20.

Figure 8 shows the results of the SNC trained by the RSTFE for different initial positions. Finally, Figure 9 shows the control signal for both RSTFE and SNC for initial position at 0.014 m. The RSDO results are also displayed for comparison. Notice that the SNC control signal is continuous and smooth, whereas the RSDO and RSTFE signal vary abruptly.

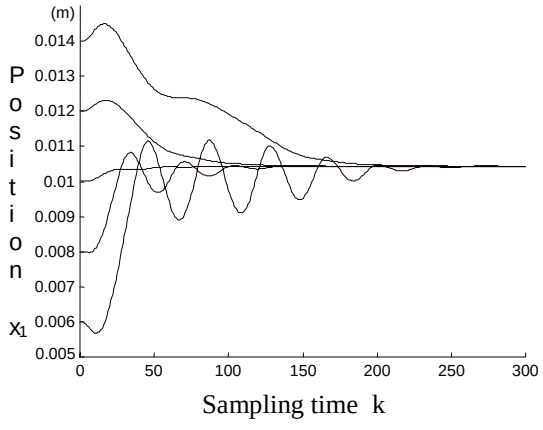

Figure 8. SNC trained by the RSTFE.

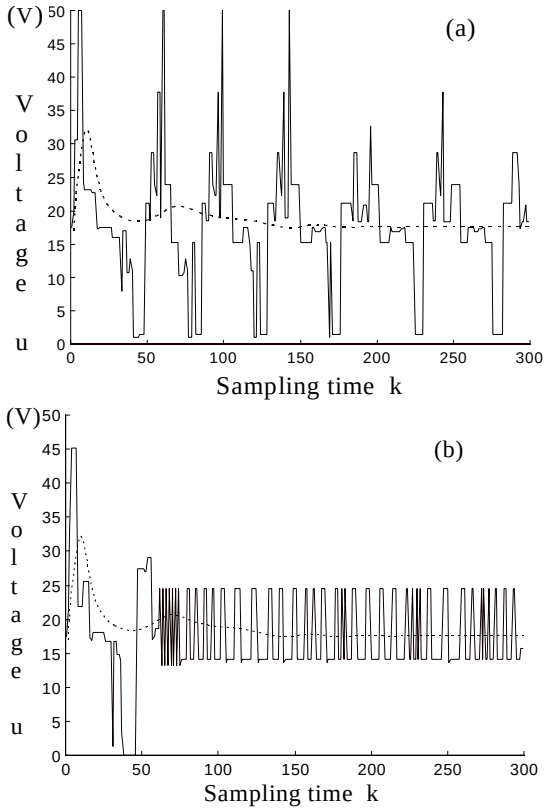

Figure 9. (a) Control signal generated by the RSTFE (solid line) and by the SNC (dotted line) for initial position at 0,014 m. (b) Control signal generated by the RSDO (solid line) and by the SNC (dotted line) for initial position at 0,014 m.

We remark that despite of the fact that the SNC does not learn in real time, a hybrid SNC/adaptive critic scheme may be proposed to again yield an adaptive control system.

6 DISCUSSIONS AND CONCLUSIONS

In this paper, reinforcement learning schemes with adaptive elements are used in the design of different controllers, providing a solution to the control problem described. In addition, an MPNN trained by one of the developed schemes is used as the controller.

Each controller developed has its own characteristics. The RSDO can be used on-line, is adaptive and presents fast learning. The RSTFE can also be used on-line, is adaptive, presents fast learning and needs no knowledge of the plant parameters or models. The main features of the developed schemes are that few trials are required to train the controllers and a variety of control actions is taken rather than only two actions as in the standard reinforcement schemes. Finally, the SNC presents a continuous control signal, removing the oscillation that may occur when using the other controllers. Hence, each control scheme can be used with success to solve different nonlinear control problems.

The example presented showed that all developed control schemes can cope with a wider operating range for the plant than that obtained with the quadratic linear optimal controller presented. Well developed control methods can be successfully applied to suspension systems, but they are more dependent on the knowledge of the plant dynamics and uncertainties.

REFERENCES

Anderson, C. H. (1989). Learning to Control an Inverted Pendulum Using Neural Networks. *IEEE Control Systems Magazine*, Vol. 9, pp. 31-36.

Barto, A. G., R. S. Sutton and C. W. Anderson (1983). Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13, pp. 834-846.

Bazaraa, M. S., H. D. Sheral and C. M. Shetty (1993). *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons Inc., New York.

Brown, M. and C. Harris (1995). *Neurofuzzy Adaptive Modeling and Control.* Prentice Hall, Englewood Cliffs, NJ.

Costa, E. F., R. Tinós, V. A. Oliveira and A. F. R. Araújo (1997). Reinforcement Learning Schemes for Control Design. *Proc. of the 1997 American Control Conference*, Albuquerque, NM, USA, pp. 2414-2418.

Dayan, P. and T. J. Sejnowski (1994). TD(λ) Converges with Probability 1. *Machine Learning*, 14 (3), pp. 295-301.

Dorigo, M. and M. Colombetti (1994). Robot Shaping: Developing Autonomous Agents through Learning. *Artificial Intelligence*, 71 (2), pp. 321-370.

Hoskins, J. C. and D. M. Himmelblau (1992). Process Control via Artificial Neural Networks and Reinforcement Learning. *Computers Chem. Engng*., 16 (4), pp. 241-251.

Kumar, P.R. and P.P. Varaiya (1986). *Stochastic Systems: Estimation, Identification and Adaptive Control*, Prentice-Hall, Englewood Cliffs, NJ.

Kaelbling, L. P., M. L. Littman and A. W. Moore (1996). *Reinforcement Learning: A Survey*. *Journal of Artificial Intelligence Research* 4, pp. 237-285.

Milan, J. A. (1996). Rapid, Safe and Incremental Learning of Navigation Strategies. *IEEE Transactions on Systems, Man and Cybernetics*, 26 (3), pp. 408-420.

Millington, P. J. and W. L. Baker (1990). Associative Reinforcement Learning for Optimal Control. *AIAA Guid. Nav. and Contr.*, Vol. 2, pp. 1120-1128.

Moriarty, D. A. and R. Mükkulainen (1996). Efficient Reinforcement Learning through Symbiotic Evolution. *Machine Learning*, 22 (1), pp. 11-32.

Muller, W. T., R. S. Sutton and P. J. Werbos, Eds. (1990). *Neural Networks for Control*. MIT Press, Cambridge, MA.

Shelton, R.O. and J. K. Peterson (1992). Controlling a Truck with an Adaptive Critic CMAC Design. *Simulation*, 58 (5), pp. 319-326.

Sutton, R.S. (1988). Learning to Predict by the Method of Temporal Differences. *Machine Learning*, 3 (1), pp. 9-44.

Sutton, R.S. (1990). Integrated Architectures for Learning, Planning and Reacting Based on Approximating Dynamic Programming. *Proc. of the Seventh International Conference on Machine Learning*, Austin, TX, USA, pp. 216-224.

Sutton R. S. (1990). First Results with Dyna and Integrated Architecture for Learning, Planning and Reacting. In W. T. Muller, R. S. Sutton and P. J. Werbos (Eds.) *Neural Networks for Control*, MIT Press, pp. 179-189.

Sutton R. S., A. G. Barto and R. J. Williams (1992). Reinforcement Learning is Direct Adaptive Optimal Control. *IEEE Contr. System Magazine*, pp. 19-22.

Vidyasagar, M. (1993). *Nonlinear Systems Analysis*. Prentice-Hall, Englewood Cliffs, NJ.

APPENDIX

The system suspension dynamic equations:

$$x_1(k+1) = T_0{}^2 g + \frac{L_{b0}}{2am}\left(\frac{T_0 x_3(k)}{1+x_1(k-1)/a}\right)^2 + 2x_1(k-1) - x_1(k-2) \quad \text{(A-1)}$$

$$x_2(k+1) = (1/T_0)\left(x_1(k) - x_1(k-1)\right) \quad \text{(A-2)}$$

$$x_3(k+1) = \left(\frac{1}{T_0 R + L}\right)\left(L x_3(k-1) + T_0 u(k)\right) \quad \text{(A-3)}$$

where x_1 is the sphere position, x_2 the sphere speed, x_3 the coil current, T_0 the sampling period, u the coil voltage, g gravity constant, m the mass of the steel ball, R the coil resistance, L the coil inductance, L_{b0} the coil inductance when $x_1 = 0$ and a a constant.

Parameters of real system of magnetic suspension:

m=0.02255 kg, R=19.9 Ω, a=0.00607, L_b =0.47 H, L_{b0} =0.0245 H and T_0=0.001 s.

Equilibrium point: (x_{1o} , x_{2o} , x_{3o}) = (0.01; 0.0; 0.876).

Practical constraints on the state:

$$0.005 \le x_1 \le 0.015 \text{ and } 0 \le x_3 \le 2.51$$

Practical constraints on the control action:

0 ≤ u ≤ 50 V.

Intervals for x_1:
[0.0050 0.0065]; [0.0065 0.0078]; [0.0078 0.0088]; [0.0088 0.0094]; [0.0094 0.0099]; [0.0099 0.0100]; [0.0100 0.0101]; [0.0101 0.0106]; [0.0106 0.0112]; [0.0112 0.0122]; [0.0122 0.0135]; [0.0135 0.0150].

Intervals for x_2:
[-0.40 -0.20]; [-0.20 -0.05]; [-0.05 0.00]; [0.00 0.05]; [0.05 0.20]; [0.20 0.40].

Intervals for x_3:
[0.00 0 .94]; [0.94 1.26]; [1.26 1.57]; [1.57 2.51].

Parameters of the LQR:

$Q=5\times10^{6}\,\mathrm{I}_3$; $R=1\times10^{-4}$.

Parameters of the RSDO:

t_{max}=100; k_{max}=500; x_{1cmax}=0.0105; x_{1cmin}=0.0095; u_{max} = 50; u_{min} = 0; α=3; β=0.2; δ=0.85; γ=0.995, λ=0.95, f is the identity function and n_s is a real random variable with normal probability density function with zero mean and variance equal to one.

Parameters of the RSTFE:

t_{max}=100; k_{max}=500; k_c=50; x_{1cmax}=0.0108; x_{1cmin}=0.092; x_{1imax}=0.015; x_{1imin}=0.005; x_{2cmax}=0.1; x_{2cmin}=-0.1; x_{2imax}=0.3; x_{2imin}=-0.3

$x_{3cmax} = x_{3imax} = \infty$; $x_{3imax} = x_{3imin} = -\infty$; u_{max} = 50; u_{min} = 0; $\mu_{e\,max}$=9; for X_{ini}

we set x_2=0 and x_3=0.876, and we varied x_1=0.006 to x_1=0.014 in steps of 0.001; α=3; β=0.2; λ=0.7; γ=1, δ=0.9, again f is the identity function and n_s is a real random variable with normal probability density function with zero mean and variance equal to one.