

CONTINUOUS LEARNING AUTOMATA AND ADAPTIVE DIGITAL FILTER DESIGN

M. N. Howell and T. J. Gordon

Dept. of Aeronautical and Automotive Engineering, Loughborough University, Leics., LE11 3TU, U.K.

Keywords: reinforcement learning, adaptive signal processing, system identification

ABSTRACT - In the design of adaptive IIR filters, the multi-modal nature of the error surfaces can limit the use of gradient-based and other iterative search methods. Stochastic learning automata have previously been shown to have global optimisation properties making them suitable for the optimisation of filter coefficients. Continuous action reinforcement learning automata are presented as an extension to the standard automata which operate over discrete parameter sets. Global convergence is claimed, and demonstrations are carried out via a number of computer simulations.

1 INTRODUCTION

Adaptive digital filters have found applications in a large number of areas such as signal processing, adaptive control, communication systems and noise cancellation. The objective of the adaptation is to alter the filter coefficients of a digital filter to match an unknown system transfer function. This is also a specific case of the more general problem of system identification, where model parameters of an unknown dynamic system are estimated from input/output data. The minimisation of a performance criterion, typically the mean squared error between filter output and desired response, is often attempted using iterative search techniques. However, when the optimisation surface is multi-modal, least squared and gradient based methods often fail to converge to the global minimum. Learning systems with improved global optimisation properties, such as genetic algorithms and learning automata, are to be preferred in these cases. Here we develop an approach based on learning automata, as this is more applicable to adaptive systems which require learning in real-time.

Stochastic learning automata [1,2,3] can operate in random and unknown environments, and have been successfully applied in a number of areas, including practical engineering problems such as suspension control [4] and power system regulation [5]. They operate by selecting actions via a stochastic process; these actions operate on some environment and are assessed according to a measure of system performance. Figure 1 shows the typical architecture; the automaton selects actions (X) probabilistically, these are applied to the environment, and the performance evaluation function provides a reinforcement signal β . This is used to update the automaton's internal probability distribution, whereby actions that achieve desirable

performance are reinforced via an increased probability, while those that don't are penalised or left unchanged depending on the particular learning rule employed. The result is that, over time, the average performance of the system will improve until some limit is reached. In terms of optimisation problem, the action with the highest probability corresponds to the global minimum, and rigorous proofs of convergence are available for this [1, 2].

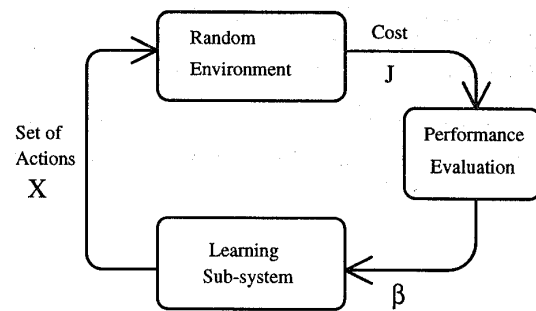


Fig.1. Reinforcement Learning System.

A wide variety of different learning rules have been reported in the literature, and one of the most widely used algorithms is the linear reward/inaction (LRI) scheme, which has been shown to have guaranteed and satisfactory convergence properties [2]. In response to action x_i being selected at time step n , the probabilities are updated as follows:

$$\begin{aligned} p_i(n+1) &= p_i(n) + \theta\beta(n)(1 - p_i(n)) \\ p_j(n+1) &= p_j(n) - \theta\beta(n)p_j(n) \quad \text{if } i \neq j \end{aligned} \quad (1)$$

θ is a learning rate parameter, $0 < \theta < 1$ and $\beta \in [0,1]$ is the reinforcement signal; $\beta = 1$ indicates the maximum reward and $\beta = 0$ is a null reward. Eventually, the probability of successful actions will increase to become close to unity, and if a single, most successful action predominates in this way, the automaton is deemed to have converged.

With a large number of discrete actions the probability of selecting any action becomes low and the convergence time can become excessive. To avoid this, learning automata can be connected in parallel as shown in Figure 2. Each automaton operates on a smaller number of actions and the 'team' works together in a co-

operative manner. This scheme can also be used to apply multivariable actions.

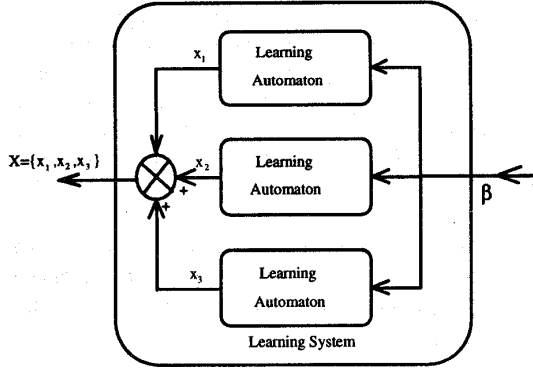


Fig.2. Interconnected Automata

Tang and Mars [6,7,8] have shown that discrete stochastic learning automata can be used to determine global optima for digital filters which have multi-modal mean square error surfaces. However, the discrete nature of the automata requires the discretisation of a continuous parameter space, and the level of quantisation tends to reduce the convergence rate. To help overcome this, a sequential approach may be adopted [4] where an initial coarse quantisation is later refined with re-quantisation around the most successful action. Here, an inherently continuous form of automaton is used, to speed the learning process and avoid this additional complexity.

2 CARLA ALGORITHM

The Continuous Action Reinforcement Learning Automata (CARLA) was developed as an extension of the stochastic discrete learning automata for applications involving searching a continuous action space in a random environment [9]. Several CARLA can be connected in parallel in order to search multidimensional spaces. The automaton's discrete probability distribution is replaced with a continuous probability density function, which is again used as the basis for action selection. It operates an LRI learning rule similar to the discrete learning automata. Successful actions are updated via a Gaussian neighbourhood function, that increases the probability density in the vicinity of the successful applied action. Figure 3 summarises the learning cycle for a single CARLA. The initial probability distribution can be chosen as being uniform over a desired range and over many iterations this converges to a Gaussian distribution around the best action value (in cases where the environment is uni-modal).

Initialise the probability density function to a uniform distribution
Repeat
 - *Select an action using its probability density function*
 - *Execute action on the environment*
 - *Receive cost/reward for previous action*
 - *Update performance evaluation function β*
 - *Update probability density function*
Until stopping condition

Fig. 3. Generic Pseudo-code for a single CARLA

If action x is defined over the range (x_{\min}, x_{\max}) , the probability density function $f(x, n)$ at iteration n is updated according to the following rule:

$$f(x, n+1) = \begin{cases} \alpha [f(x, n) + \beta(n)H(x, r)] & \text{if } x \in (x_{\min}, x_{\max}) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where the parameter α is chosen to renormalize the distribution according to the condition

$$\int_{x_{\min}}^{x_{\max}} f(x, n+1) dx = 1 \quad (3)$$

$\beta(n)$ is again the reinforcement signal from the performance evaluation and $H(x, r)$ is a symmetric Gaussian neighbourhood function centred on $r = x(n)$

$$H(x, r) = \lambda \exp\left(-\frac{(x-r)^2}{2\sigma^2}\right) \quad (4)$$

λ and σ are parameters that determine the height and width of the neighbourhood function and are defined as in terms of the range of actions

$$\sigma = g_w \cdot (x_{\max} - x_{\min}) \quad (5)$$

$$\lambda = \frac{g_h}{(x_{\max} - x_{\min})} \quad (6)$$

The speed and resolution of learning are then controlled by the free parameters g_w and g_h . Let action $x(n)$ be applied to the environment at iteration n , and return a cost or performance index $J(n)$. Current and previous costs are stored as a reference set $R(n)$, from which the median and minimum values J_{med} and J_{min} are found.

Then $\beta(n)$ is defined as follows:

$$\beta(n) = \max\left\{0, \frac{J_{med} - J(n)}{J_{med} - J_{min}}\right\} \quad (7)$$

To avoid problems with infinite storage, and to allow the system to adapt to changing environments, only the last m values of costs are stored in $R(n)$.

For multivariable actions, each action component is associated with a separate probability density function. These are all updated using a single cost function (based on the error between filter output and desired response). The approach also lends itself naturally to problems of multi-objective optimisation, by assigning different actions to different objective functions; such cases will not be considered in this paper however.

A piecewise linear function is used to implement the probability density $f(x, n)$. Dividing the action range into $N-1$ intervals, namely

$$x_{\min} = x_1 < x_2 < \dots < x_{N-1} < x_N = x_{\max} \quad (8)$$

, we write

$$f(x, n) = m_i x + c_i \quad (9)$$

$$(x_i \leq x \leq x_{i+1}, \quad i = 1, 2, \dots, N-1)$$

Interval spacing on the x -axis is conveniently chosen according to an equal area criterion; this provides increased resolution around actions of high probability in an automatic fashion. To define an action value $x(n)$ associated with this probability density function, a uniformly distributed pseudo-random number $z(n)$ is generated in the range $[0, 1]$. Simple interpolation is then used to equate this value to the cumulative distribution function:

$$\int_{x_{\min}}^{x(n)} f(x, n) dx = z(n) \quad (10)$$

3 IMPLEMENTATION

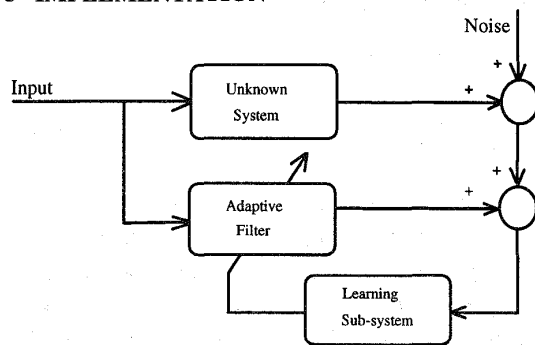


Fig. 4. Filter Adaptation by CARLA

We now apply the new techniques to some specific examples taken from the work of Tang and Mars [4,5,6]. Figure 4 represents the structure of these examples, which are implemented as stochastic optimisation problems, the input signal and the corrupting noise

being generated by pseudo-random signals. For simplicity, the corrupting noise has been set to zero in the examples presented here. The examples contain only a small number of unknown filter parameters, but are well suited to this study, since the error surfaces (mean squared errors in the sense of an ensemble average) can be represented as known functions of the model parameters and as such are seen to contain local minima.

In the following, the performance index to be minimised is the sum of squared errors between plant and model outputs, $y_p(t)$ and $y_m(t)$, over a period of T discrete-time intervals:

$$J = \sum_{t=1}^T (y_p(t) - y_m(t))^2 \quad (11)$$

We have set $T=100$ throughout the following examples. (The precise value of T is not especially significant here, and no investigation has been carried out regarding the effect of larger or smaller values.) The 'unknown' plant is operated as a continuous process, while the internal states of the model filter are initialised to zero at each iteration, before a new set of parameters are tested. For performance evaluation, storage of costs within the set R was limited to 500 values, and in each example 3000 iterations of the CARLA learning were applied. To verify the consistency of the learning process, ten independent examples were simulated for each case. The CARLA parameters were fixed at $g_w = 0.02$ and $g_h = 0.3$ for all simulations.

3.1 Second-Order Plant with First-Order Model

This problem involves the modelling of a second-order plant by a first order transfer function. This problem was first proposed by Johnson and Larimore [10] and has been used to demonstrate that global convergence could not be obtained using recursive LMS [11] nor a hyperstability based algorithm. The plant transfer function is given by

$$H_p(z^{-1}) = \frac{y_p(z^{-1})}{u(z^{-1})} = \frac{0.05 - 0.4z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}} \quad (12)$$

and is modelled by

$$H_m(z^{-1}) = \frac{y_m(z^{-1})}{u(z^{-1})} = \frac{b}{1 - az^{-1}} \quad (13)$$

Here the input $u(t)$ is a white noise signal, which is applied simultaneously to both the plant and the model - see Figure 4. The learning system is to search the two-dimensional parameter space of a and b with the aim of reducing the values for J in equ (13). Figure 5 shows a contour plot of the error surface, which has a

local minimum at $(a, b) = (-0.519, 0.114)$ and a global minimum at $(0.906, -0.311)$.

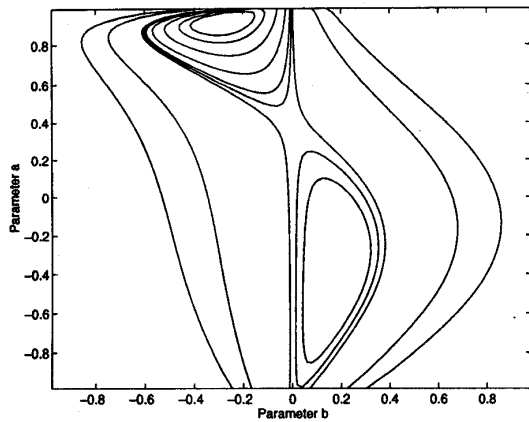


Fig. 5. Error Surface of Example 3.1

Two CARLA were used in this example (one for each parameter) with each initialised to a uniform distribution in the range $(-1, 1)$. All ten independent simulation runs converged very accurately to the optimal solution within 3000 iterations. The corresponding cost reductions in mean square error are shown in Figure 7. This shows the average cost over 100 consecutive iterations, averaged again over the ten runs. The evolution of the two probability density functions, obtained for a typical run, are shown in Figure 8. It can be seen that most of the convergence is achieved within the first 1000 iterations, with some gradual sharpening of the distribution occurring after that. The final probability densities are reproduced in Figure 9, and the modes (peaks) of these distributions can be taken as the final parameter estimates. These results also indicate that parameter a has converged to the sharper distribution, and this is consistent with the eccentricity of the contours around the global minimum in Figure 6.

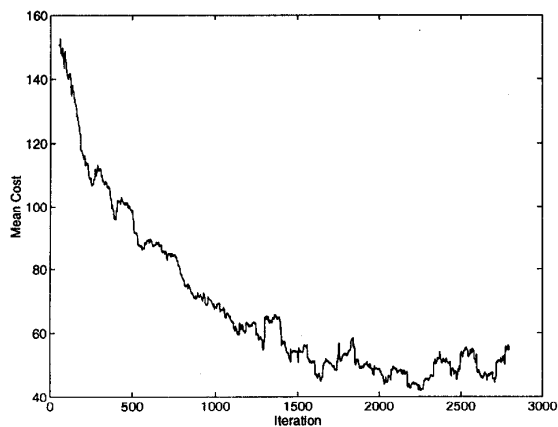


Fig. 6. Averaged Cost Reduction During Learning

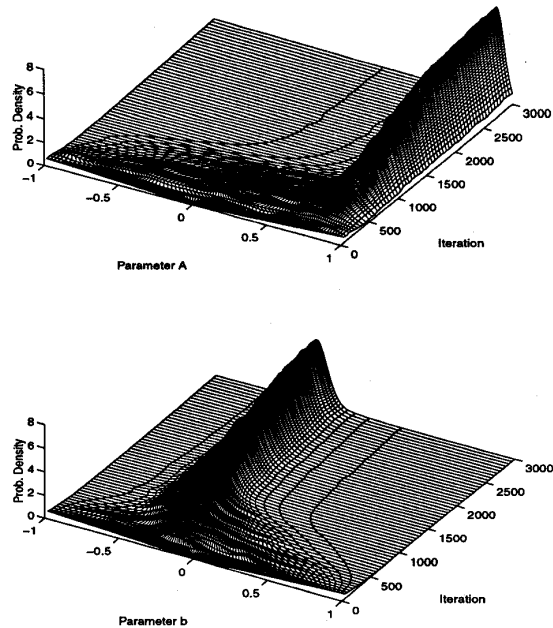


Fig. 7. Convergence of Probability Density Functions

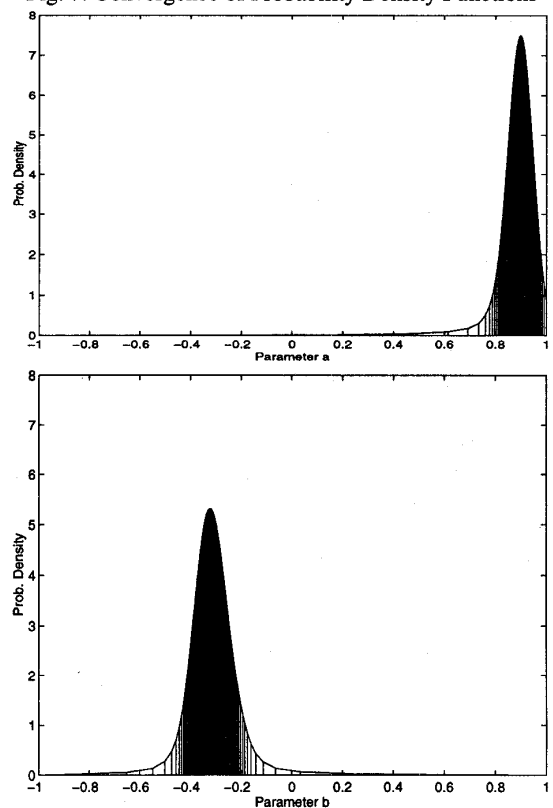


Fig. 8. Final Probability Densities

Informally, this suggest that measures of distribution dispersion (e.g. variance) could also be useful to indicate 'error bounds' on the parameter estimates; no

formal analysis of this effect has yet been carried out though.

3.2 Second-Order Plant with Second-Order Model

This system was considered by Fan and Jenkins [11]. The plant and model transfer functions are given by

$$H_p(z^{-1}) = \frac{1}{(1-0.7z^{-1})^2} = \frac{1}{1-1.4z^{-1}+0.49z^{-2}} \quad (14)$$

$$H_m(z^{-1}) = \frac{b}{1+a_1z^{-1}+a_2z^{-2}} \quad (15)$$

The parameter estimation is made more difficult by the coloration of the system input spectrum; $u(t)$ is generated from white noise via the following FIR filter

$$H_{in}(z^{-1}) = (1-0.7z^{-1})^2 \cdot (1+0.7z^{-1})^2 \quad (16)$$

Clearly, this filter tends to cancel the plant dynamics. The global minimum is located at $(a_1, a_2, b) = (-1.4, +0.49, 1)$, where the model exactly matches the plant. A contour plot of the cost surface, for $b = 1$, is shown in Figure 9. Stability is only guaranteed if the poles lie within the unit circle, which corresponds to the case where a_1 and a_2 lie within the triangle [12] shown in the figure:

$$1-a_1+a_2 > 0, 1+a_1+a_2 > 0, 1-a_2 > 0 \quad (17)$$

Filter adaptation was implemented in two different ways, both using the CARLA.

Method 1

This is the straightforward application of the above. Three CARLA are used - one for each of the parameters a_1 , a_2 and b - the search regions being $(-2,2)$, $(-1,1)$, $(-1,1)$ respectively. The existence of instability in the model filter does not present any major problems, provided the filter simulation does not lead to a fatal run-time error in the digital computer (potentially disastrous in a real-time application!). However, on the assumption that unstable model parameters will never be reinforced, there is little point 'wasting good learning time' on these selections. Thus, once the parameters have been chosen via their probability densities, a_1 and a_2 are tested against the stability condition; if necessary the selection of a_1 and a_2 is repeated until they lie within the stability triangle.

Method 2

Here a total of four CARLA were employed - one for b and three for the two denominator parameters a_1 and a_2 - essentially one aligned with each edge of the stability triangle. This does not exclude the possibility

of an unstable filter choice, but the alignment is more natural, and the example serves to demonstrate the flexibility of the CARLA approach. Defining the following transformed variables

$$\begin{aligned} \phi_1 &= a_1 \\ \phi_2 &= 2^{-0.5}(a_2 + a_1 + 1) \\ \phi_3 &= 2^{-0.5}(a_1 - a_2 - 1) \end{aligned} \quad (18)$$

the corresponding action ranges are $(-2, 2)$, $(0, 2\sqrt{2})$ and $(-2\sqrt{2}, 0)$. Only two of the ϕ_i are independent and so actions at any iteration are based on the two which have converged most in distribution. The selected values are used to determine a_1 and a_2 , and the stability condition is checked for the other variable. Once a stable action has been confirmed, the filter is tested, and all three distributions are updated according to the reinforcement rule (2). Figure 10 shows that there is actually a worthwhile improvement in convergence speed achieved by this method.

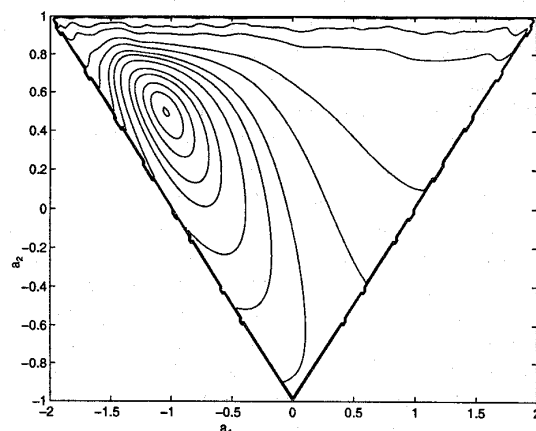


Fig. 9. Contour plot showing the stability region for Example 3.2.

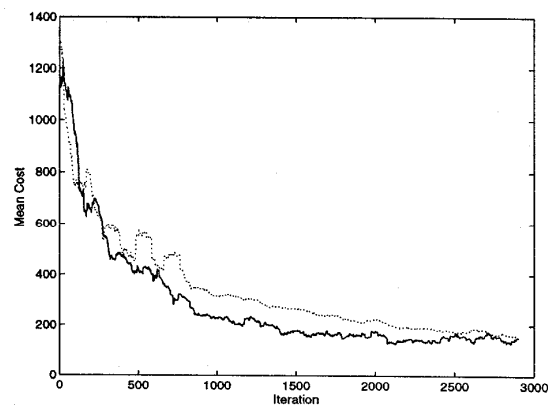


Fig. 10. Cost reduction for example 3.2. (solid line represents Method 2)

4 SUMMARY AND CONCLUSIONS

This paper has considered the application of a new learning method to the problem of selecting optimal parameters in digital IIR filters, to give minimum mean-square output error compared to a prescribed, but unknown plant. The work has focused attention on obtaining global minima in situations where multi-modal error surfaces easily lead to local minima. The known global convergence properties of Learning Automata has motivated their use in this problem class, and the implementation of a continuous version of the automata makes the approach both simple and effective, and capable of adaptation in a changing environment. Note however, that the extensions and enhancements incorporated in multi-CARLA learning mean that global convergence has not been proved; this will be the subject of a future paper.

It has been shown elsewhere [9] that the approach is also robust to added stochastic disturbances, and though convergence will be slowed, the methods presented here are applicable in cases where there is unmeasured input or output noise. Other examples, not presented here, also show that the methods work well with higher order filters.

The approach is well-suited to real-time application. Though it is clear that a wide range of filter parameters will be tested, especially in the early stages of learning, this need not be of such great concern, even if the *application* of the learnt filter coefficients is also in real-time. Suppose for example, the estimated filter coefficients are used in adaptive control, where the controller dynamics are modified according to filter estimates for the plant. In this case, it is natural to use some measure of the 'most-likely' filter parameters (e.g. the mode) rather than simply the parameters being tested by the CARLA.

REFERENCES

- 1 Najim, K. and Poznak, A.S., *Learning Automata - Theory and Applications*, 1994, (Pergamon Press, Oxford).
- 2 Narendra, K.S. and Thathachar, M.A.L. *Learning Automata: An Introduction*. 1989., (Prentice Hall, London).
- 3 Baba, N. *New Topics in Learning Automata Theory and Applications*. Lecture Notes in Control and Information Sciences - 1984, Thoma, M. ed. (Springer-Verlag, New York).
- 4 Frost, G.P., Gordon, T.J., Howell, M.N. and Q.H. Wu., *Reinforcement learning of active and semi-active vehicle suspension control laws*, Proc. Instn. Mech. Engrs. Part I, 1996, Vol. 210, 249-257.
- 5 Wu, Q.H., *Learning coordinated control of synchronous machines in multi-machine power systems*, Proc. IEEE Sys., Man, Cybern. Conf., Vol. 3., 728-733, 1993.
- 6 Tang, C.K.K and Mars, P., *Stochastic Learning Automata and adaptive IIR filters*, *IEE Proceedings-F*, Vol. 138, No. 4, August 1991.
- 7 Tang, C.K.K and Mars, P., *Intelligent Learning Algorithms For Adaptive Digital Filters*, Electronics Letters, Vol.25. No. 23. 1989,
- 8 Tang, C.K.K and Mars, P., *Games of Stochastic Learning Automata and Adaptive Signal processing*, IEEE Trans. Systems, Man and Cybernetics, pp851-856, Vol.23, No. 3, 1993.
- 9 Howell, M.N., Frost G.P., Gordon T.J., Wu Q.H., *Continuous action reinforcement learning applied to vehicle suspension control*, Mechatronics, Vol 7, No. 3, pp263-276, 1997.
- 10 Johnson, C.R., and Larimore, M.G., *Comments on and additions to "An Adaptive Recursive LMS filter"*, Proceeding of the IEEE, Vol. 65, pp1399-1402, 1977.
- 11 Fan, H. and Jenkins, W.K., *A New Adaptive IIR Filter*, *IEEE Trans. Circuits and Systems*, Vol. CAS-33, No. 10, 1996.
- 12 Shynk, J.J., *Adaptive IIR Filtering*, IEEE ASSP Magazine April 1989.