SEQUENTIAL LEARNING FOR ADAPTIVE CRITIC DESIGN: AN INDUSTRIAL CONTROL APPLICATION

James J. Govindhasamy†, *Member, IEEE*, Seán F. McLoone††, *Senior Member, IEEE*, and George W. Irwin†, *Fellow, IEEE*

†*Intelligent Systems and Control Research Group, Queen's University Belfast, Belfast BT9 5AH, N. Ireland, UK.*

††*Dept. of Electronic Engineering National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland.*

E-mail: *j.govindasamy@ee.qub.ac.uk*

ABSTRACT

This paper investigates the feasibility of applying reinforcement learning (RL) concepts to industrial process optimisation. A model-free action-dependent adaptive critic design (ADAC), coupled with sequential learning neural network training, is proposed as an online RL strategy suitable for both modelling and controller optimisation. The proposed strategy is evaluated on data from an industrial grinding process used in the manufacture of disk drives. Comparison with a proprietary control system shows that the proposed RL technique is able to achieve comparable performance without any manual intervention.

1. INTRODUCTION

A learning system that identifies a model of a process automatically and reconfigures the controller when necessary, has been a long-standing vision of control engineers [1]. A key requirement of such systems is to be able to *learn* from previous experience and/or examples of appropriate behaviour how to achieve long term goals, and not just how to perform in the immediate future. Reinforcement learning (RL) within an adaptive critic framework is capable of meeting this requirement, but computational and convergence issues make it difficult to implement in practice.

Sequential learning neural networks employ a procedure that involves growing and/or pruning networks iteratively as the training data is presented. Learning is achieved through a combination of new neuron allocation and parameter adjustment of existing neurons. New neurons are added if presented training patterns fall outside the range of existing network neurons. Otherwise the network parameters are adapted to better fit the patterns. This procedure is usually combined with pruning where neurons which contribute little to the overall network response over an extended period of time are removed. The seminal paper by Platt [2], proved that sequential learning using a constructive technique, called resource allocation networks (RAN) is suitable for online modelling. Since then there have been many publications on research into application of this concept to supervised learning problems [3,4,5].

To date there have only been a handful of publications which explore the use of sequential learning neural networks with RL algorithms [6,7], none of which have evaluated their applicability to industrial processes.

In this paper a novel sequential learning model-free action dependent adaptive critic (ADAC) design for RL is investigated for modelling and control of an industrial grinding process used in the manufacture of disk drive media. The proposed sequential learning methodology overcomes the *a priori* fixed network architecture limitation normally associated with ADACs by extending the search to the entire weight space of the neural network topology. It also searches for a near minimal network size which suits the complexity of the learning task, thereby increasing the speed and efficiency of computation.

The remainder of the paper is organised as follows. Section 2 provides a brief description of the ADAC framework. Section 3 gives details of the neural network implementation and sequential learning algorithm used. The industrial application is described in section 4 followed by details of the ADAC simulations performed and the results obtained in section 5. Finally conclusions are presented in section 6.

2. PRELIMINARIES

The fundamental solution to sequential optimisation or dynamic programming problems uses Bellman's Principle of Optimality [8]:... *an optimal trajectory has the property that no matter how the intermediate point is reached, the rest of the trajectory must coincide with an optimal trajectory as calculated with the intermediate point as the starting point*. This principle is applied by devising a "primary" reinforcement function or reward, $r(k)$, that incorporates a control objective for a particular scenario in one or more measurable variables. A secondary utility is then formed, which incorporates the desired control objective through time, the so-called Bellman equation, expressed as

$$J(k) = \sum_{q=0}^{\infty} \gamma^q r(k+q) \tag{1}$$

where γ is a discount factor ($0 < \gamma < 1$), which determines the importance of the present reward as opposed to future ones. The reinforcement, $r(k)$, takes a binary form with $r(k) = 0$ when the event is successful (objective is met) and $r(k) = -1$ when it fails (when the objective is not met). Hence, the purpose of dynamic programming is to choose a sequence of control actions to maximise $J(k)$, the cost-to-go. Unfortunately, this optimisation problem is computationally intractable due to the complexity of the backward numerical solution process required, i.e. as a result of the "curse of dimensionality" for real problems. Thus, there is a need for more tractable approximation methods. The basis for such methods is a useful identity derived from Eq. 1, called the Bellman Recursion equation,

$$J(k) = r(k) + \gamma J(k+1) \qquad (2)$$

Si and Wang [9] formulated a modified version of Eq. 2, where instead of approximating $J(k)$, they proposed that a Critic Network be used to approximate the future accumulated reward-to-go, defined as

$$R(k) = r(k+1) + \gamma r(k+2) + \ldots , \qquad (3)$$

where $R(k) \triangleq J(k+1)$. The resulting RL scheme, known as an Action Dependent Adaptive Critic (ADAC), is illustrated in Figure 1.

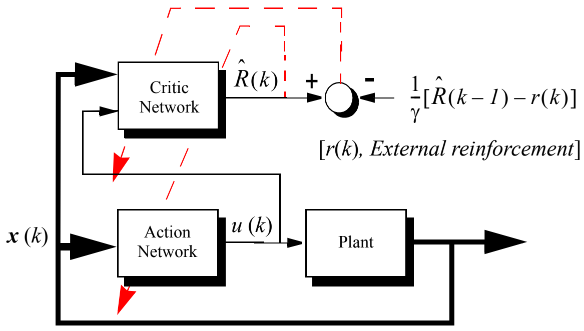

Fig. 1. Schematic of the Action Dependent Adaptive Critic scheme

The Critic Network is trained by using $\hat{R}(k-1)$, the previous estimate of the cost, and the current reward, $r(k)$, to provide a target value for the current cost estimate $\hat{R}(k)$. Thus,

$$[R(k)]_{\text{target}} = \frac{1}{\gamma}[\hat{R}(k-1) - r(k)] \qquad (4)$$

The instantaneous error, $e_c(k) = \hat{R}(k) - [R(k)]_{\text{target}}$, is then a function of two successive values of $\hat{R}$:

$$e_c(k) = \hat{R}(k) - \frac{1}{\gamma}[\hat{R}(k-1) - r(k)] , \qquad (5)$$

and is usually referred to as the temporal difference error.

The objective when training the Action Network is to maximise the future accumulated reward-to-go $\hat{R}(k)$. This has a maximum value of 0 for all k with the result that the instantaneous error estimate for the network is

$$e_a(k) = \hat{R}(k) . \qquad (6)$$

3. NEURAL NETWORKS IMPLEMENTATION

The sequential learning neural network architecture used here to provide the Critic and Action mappings is a Radial Basis Function (RBF) topology defined as

$$y_k = f(\boldsymbol{x}_k, \boldsymbol{w}) = \sum_{i=1}^{m} h_i \phi_i[\boldsymbol{x}_k(\boldsymbol{c}_i, \sigma_i)] , \qquad (7)$$

where $\phi_i(.)$ are localised Gaussian functions given by:

$$\phi[\boldsymbol{x}_k(\boldsymbol{c}_i, \sigma_i)] = \exp\left(-\frac{\|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2}{\sigma_i^2}\right) . \qquad (8)$$

Here, $\boldsymbol{c}_i$ and σ_i are the centre and width of the i^{th} basis function (hidden neuron) in the network and h_i is the linear output weight that connects the i^{th} basis function, to the output summer. The various centre, width and height parameters constitute the overall network weights vector, $\boldsymbol{w}$.

Defining the Mean Squared Error (MSE) cost functions for the Critic and Action network as $E[e_c^2(k)]$ and $E[e_a^2(k)]$ respectively, stochastic gradient descent weight-update rules can be derived on the basis of the instantaneous cost function estimates

$$E_c(k) = e_c^2(k) \qquad (9)$$

and

$$E_a(k) = e_a^2(k) . \qquad (10)$$

Here a recursive Levenberg Marquardt algorithm proposed by Ngia and Sjöberg [10] is used. This is essentially a regularised implementation of the Recursive Prediction Error algorithm and is defined as follows:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + 2\boldsymbol{P}_k \nabla\psi_k e(k) \qquad (11)$$

$$\overline{\boldsymbol{P}}_k = \frac{1}{\alpha_k}[\boldsymbol{P}_{k-1} - \boldsymbol{P}_{k-1}\Omega_k \boldsymbol{S}_k^{-1}\Omega_k^T \boldsymbol{P}_{k-1}] \qquad (12)$$

$$\boldsymbol{P}_k = \frac{1}{trace[\overline{P}_k]}\overline{\boldsymbol{P}}_k \qquad (13)$$

$$\boldsymbol{S}_k = \alpha_k \Lambda_k + \Omega_k^T \boldsymbol{P}_{k-1} \Omega_k \qquad (14)$$

$$\nabla\psi_k = \frac{\partial}{\partial \boldsymbol{w}_k} f[\boldsymbol{w}_k, \boldsymbol{x}_k] = \frac{\partial}{\partial \boldsymbol{w}_k}\hat{R}(k) \qquad (15)$$

$$\Lambda_k = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix}^{-1} , \quad \Omega_k^T = \begin{bmatrix} & & \nabla\psi_k & & \\ 0 & \ldots & 1 & \ldots & 0 \end{bmatrix} \qquad (16)$$

where Ω_k is a $N_w \times 2$ matrix with the first column containing the gradient vector $\nabla\psi_k$ and the second column consisting of a $N_w \times 1$ zero vector with a different element set to 1 at each iteration according to $k \bmod(N_w) + 1$, ρ is a positive scalar that controls the amount of regularisation, matrix $\boldsymbol{P}(k)$ is the inverse of the Gauss-Newton Hessian (the covariance matrix of weight estimate $\boldsymbol{w}_k$) and α_k is a scalar forgetting factor that controls the memory of the algorithm.

The prediction error $e(k)$ is as defined in Eq. 5 for the Critic Network weights, $\boldsymbol{w}_k^c$, and Eq. 6 for the Action Network weights, $\boldsymbol{w}_k^a$. The corresponding gradient vectors are computed by applying the chain rule and back-propagation, that is:

$$\nabla\psi_k^c = \frac{\partial\hat{R}(k)}{\partial \boldsymbol{w}_k^c}, \quad \nabla\psi_k^a = \frac{\partial\hat{R}(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial \boldsymbol{w}_k^a} \tag{17}$$

The sequential learning growth criterion considered here is an extension of the On-line Adaptive Centre Allocation (OLACA) algorithm [11] and is as follows:

$$d_{nc} = \min_i \|\boldsymbol{x}_k - \boldsymbol{c}_i\| > 2\underline{\alpha}\sigma_{nc}\,,\ i = 1, 2, \ldots m\,. \tag{18}$$

Here ($\boldsymbol{x}_k$, y_k) is a new data point to be fitted by the RBF network, d_{nc} is the distance between the input vector, $\boldsymbol{x}_k$, and the centre of the nearest hidden neuron, $\boldsymbol{c}_i$, while σ_{nc} is the width of the nearest neuron. The term $2\underline{\alpha}\sigma_{nc}$ defines the maximum neuron separation allowed and is a function of scalar $\underline{\alpha}$ which controls the degree of overlap between neurons (usually set equal to 1).

If the growth criterion is not satisfied network parameters are adapted using the RLM training algorithm. Otherwise a new Gaussian basis function hidden neuron is assigned as shown in Figure 2 with

$$\boldsymbol{c}_{N+1} = \boldsymbol{x}_k\,,\ h_{N+1} = e_k \text{ and } \sigma_{N+1} = \beta\frac{d_{nc}}{2}\,. \tag{19}$$

The deviation from the desired goal, e_k, is problem dependent for the Action Network and is defined as Eq. 5 for the Critic Network. The scalar β, is a user-defined parameter (usually unity) which determines the degree of overlap between neurons.

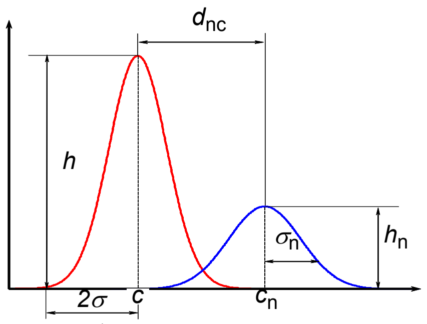

Fig. 2. The OLACA scheme

The dimensions of the weight vector and $\boldsymbol{P}_k$, are also increased accordingly, i.e.

$$\boldsymbol{w}_k = \begin{bmatrix} \boldsymbol{w}_k \\ [\boldsymbol{c}_{N+1}, \sigma_{N+1}, h_{N+1}]^T \end{bmatrix} \tag{20}$$

$$\boldsymbol{P}_k = \begin{bmatrix} \boldsymbol{P}_{k-1} & O^T \\ O & \boldsymbol{I}_{n_w \times n_w} \end{bmatrix} \tag{21}$$

where the dimension n_w is equal to the number of new parameters associated with the new Gaussian basis function and O is an appropriately dimensioned null matrix.

The pruning procedure, which is based on Yingwei *et al.* [5], involves eliminating the Gaussian kernels (*GK*) that show the least contribution to the model output for the past *M* sample instants, and can be summarised as follows;

- Compute the output of all the Gaussian kernel functions, $i = 1, 2, \ldots, m$.

$$GK(k)_i = \exp\left(-\frac{\|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2}{\sigma_i^2}\right) \tag{22}$$

- Find the largest absolute Gaussian basis function output value

$$GK_{max} = max(|GK(k)_i|) \text{ and } i = 1, 2, \ldots, m \tag{23}$$

- Determine the normalised contribution factor for each basis function:

$$\omega_i = \left|\frac{GK(k)_i}{GK_{max}}\right| \text{ and } i = 1, 2, \ldots, m \tag{24}$$

If $\omega_j < \delta$ ($\delta \ll 1$) for *M* consecutive sample instants, then prune the jth hidden neuron and reduce the dimensionality of $\boldsymbol{w}_k$ and $\boldsymbol{P}_k$.

The window size, *M,* and threshold δ are problem dependent parameters which are particularly sensitive to the level of system excitation and have to determined by trial-and-error. In the industrial application of the ADAC scheme described next $\delta = 10^{-5}$ and $M = 100$ were found to give good results.

4. INDUSTRIAL APPLICATION

The industrial application considered is the identification and control of a ring grinding process for aluminium substrate disks. The disks are ground in batches of twelve between two grindstones, as shown in Figure 3. The grindstones can be moved apart. A pick and place unit is used to place the disks between the grindstones and to remove them after grinding is completed. The grindstones move in opposite directions, causing the disks between them to move as well, which is when the grinding takes place. The rate at which the disks are ground, called the removal rate, is the critical variable. It varies depending on a number of parameters.

The initial thickness of the disks varies also, although the disks in any one batch are sorted to be approximately the same thickness. Currently, the thickness of one disk from each batch is measured before the batch is ground. The system controller calculates the actual removal rate from the previous batch and estimates the current value of removal rate. It predicts how much material has to be removed by subtracting the target thickness from the input thickness and then calculates the necessary grinding duration for the current batch. When the grinding is completed, the disk selected is measured again. If it is within specification, then the whole batch is passed. If the disk is too thick (above the upper specification limit), the disks are ground again (i.e. reworked) but if the disk is too thin (below the lower specification limit), the batch is rejected. When a grindstone is newly installed (i.e. replaced due to wear), the pressure is initially set to a low value and then gradually increased to counteract the stone deterioration, which in turn increases the removal rate. Subsequently the removal rate appears to fall until the stage is reached where it is so low that the grindstone has to be resurfaced. This is done by slicing off the worn part of the grindstone. Once re-installed the whole process is repeated.

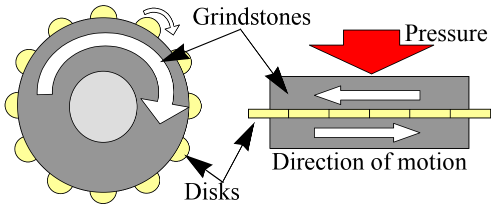

Fig. 3. The ring grinding process

5. SIMULATION RESULTS

An ADAC based modelling and control strategy will be considered. The aim is to demonstrate that the action dependent adaptive critic can be used both as a direct controller and as a framework for process identification, an aspect less well reported in the literature. The implementations will now be discussed. The main aim here, is to achieve accurate thickness control in order to minimise the number of out-of-specification disks produced by the grinding process. This process optimisation can be achieved through manipulation of the grind cycle time as illustrated in Figure 4. Here rr_k, is the removal rate (an internal variable), p_k, is the current pressure, and c_k is the current cycle time, and T_k is the unloading thickness. The unknown disturbances, d_k, include factors such as machine vibration, coolant fluctuations and operator error.

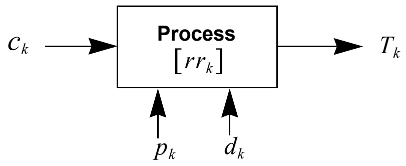

Fig. 4. The ring grinding process block diagram

Neural network based direct inverse control has been shown, in [12], to provide an effective solution to this problem and was therefore chosen as the basis for the ADAC investigation. The ADAC framework was considered for two elements of the controller design, namely developing a process model and optimising the final controller. A process model is needed as this forms the basis for the direct inverse controller implementation. The recommended model is one which predicts the removal rate, for each grind cycle, on the basis of the current state of the process.

The existing proprietary controller was used as a reference for evaluating the performance of the new ADAC controller. The accuracy of the prediction was measured in terms of the percentage normalised mean prediction error (MPE), defined as

$$MPE = \frac{1}{n}\sum_{k=1}^{n}\frac{|y_k - \hat{y}_k|}{\sigma_y} \times 100\% \tag{25}$$

where y_k is the actual response while $\hat{y}_k$ and σ_y are the prediction and standard deviation of variable y_k respectively and n is the number of data samples. In this case the MPE over the life of the test grindstone was 6.8%.

5.1 ADAC Model

The process model is required to predict the grindstone removal rate which is used to calculate the cycle time of the grinding machine. Accurate prediction of removal rate will therefore produce an improved cycle time estimate for the grind process. In the ADAC framework, the Action Network is trained to form the required process model and is connected as shown in Figure 5.

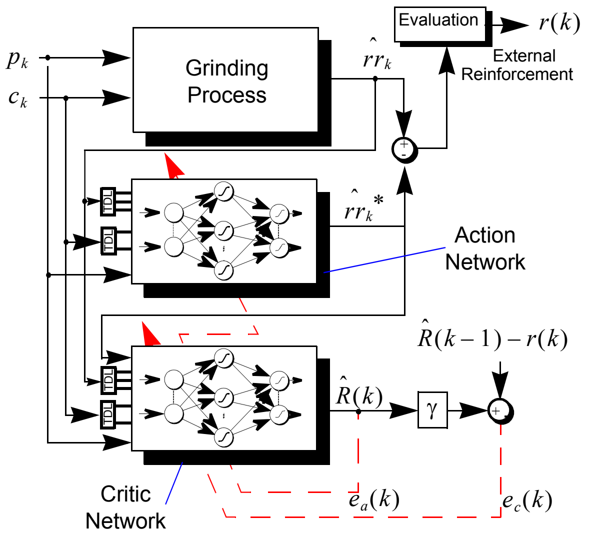

Fig. 5. Schematic of modelling strategy using the ADAC for the grinding process

For this study a nonlinear ARX modelling strategy was employed, where the current removal rate was estimated as a function of previous removal rates, rr_{k-1}, rr_{k-3} and rr_{k-5}, current pressure, p_k, and the current and past cumulative cycle times, cct_k and cct_{k-1}. The Action Network was trained online to learn the unknown mapping

$$\hat{rr}_k = f(rr_{k-1}, rr_{k-3}, rr_{k-5}, p_k, cct_k, cct_{k-1}) \tag{26}$$

The RL learning goal was to minimise the absolute prediction error between the desired removal rate, rr_k, and the predicted one, rr_k^* with the external reinforcement signal, $r(k)$, defined as

$$r(k) = \begin{cases} -1, & |\text{Prediction Error}| \le 10^{-3} \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

Both networks were trained for 2000 samples using sequential reinforcement learning, after which their weights were stored. Figure 6 shows the removal rate prediction performance of the Action Network during and after training. Although the network has clearly improved its performance during training the overall prediction capability after 2000 iterations is still quite poor. The resulting model yields a MPE of 34% over the complete stone life.

Noting that much of the error was due to low-frequency offsets in the Action Network prediction, a 1st order predict-correct, as described in [12], was introduced to enhance the accuracy of the model. This final ADAC trained model yielded a MPE of 3.1% (Figure 7), which represents a factor of two reduction in MPE compared to the proprietary control scheme.

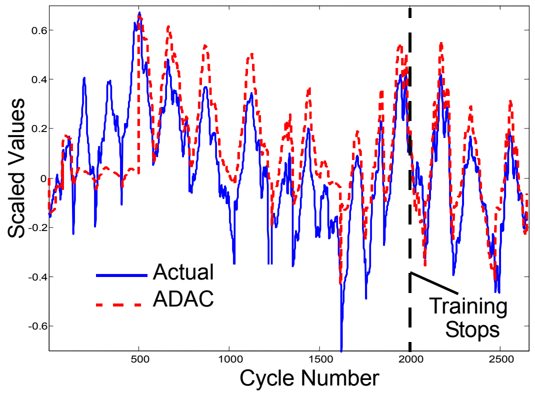

Fig. 6. The removal rate prediction from the ADAC scheme during training

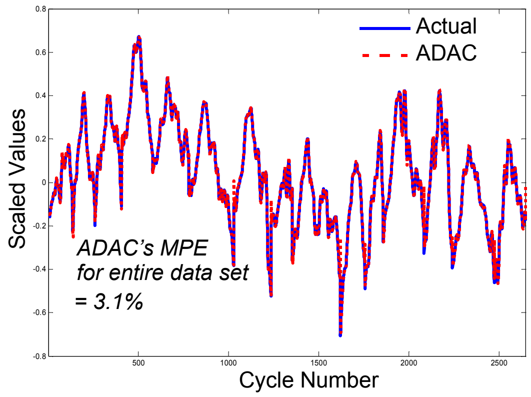

Fig. 7. The removal rate prediction from the ADAC scheme with PC

The growth of the Action and Critic Networks during training is illustrated in Figure 8. While the size of the Critic network had stabilised it is clear that the Action network was still adapting significantly at the end of the training period, an indication that further training would be beneficial.

5.2 ADAC Control

Using the removal rate model identified in the previous section as the basis for a direct inverse model control (DIMC) strategy [12], the ADAC framework was explored as a means of fine tuning the controller performance. The reinforcement signal for this more traditional ADAC role was defined in terms of the upper and lower control limits (i.e. the UCL and LCL) for the unloading thicknesses of the disks as follows.

$$r(k) = \begin{cases} -1, \mathrm{UCL} < \mathrm{ULT} < \mathrm{LCL} \\ 0,\ otherwise \end{cases} \tag{28}$$

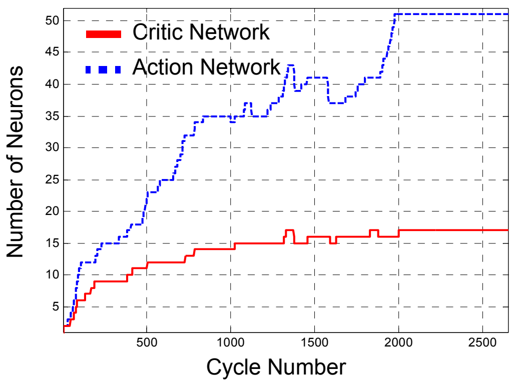

Fig. 8. The growth of the ADAC networks for the grinding process identification

Figure 9 shows the variation in disk unloading thickness obtained with the ADAC tuned controller, while Table I summarises its performance compared with the controller prior to tuning, an offline trained MLP direct inverse controller developed in [12], and the proprietary control scheme benchmark for the same grindstone data. In each case the target unloading thickness was 3075 μm .

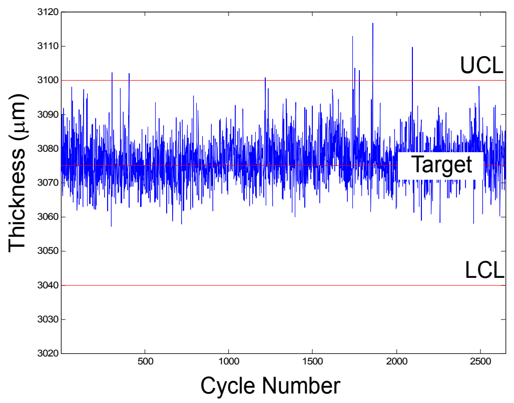

Fig. 9. Unloading thickness performance of the ADAC tuned controller

It can be seen that the sequential learning ADAC controller out performs the proprietary controller by 33% in terms of the number of rejects (out-of-spec. disks) generated. It also matches the performance of the offline trained MLP DIMC reported in [12]. Furthermore, a comparison of the proprietary and ADAC tuned controller unloading thickness distributions (Figures 10) shows that the latter achieves tighter thickness control (i.e. there is less variability in unloading thickness).

TABLE I: PERFORMANCE COMPARISON BETWEEN THE ACTUAL AND THE ADAC SCHEME

Controller	Unloading Thickness Mean (μm)	Unloading Thickness Variance (μm)	Number of Rejects (Batches)
Proprietary	3075.85	7.10	12
ADAC-model DIMC	3075.12	7.20	12
Fixed MLP DIMC	3076.32	6.41	8
ADAC-tuned DIMC	3076.32	6.31	8

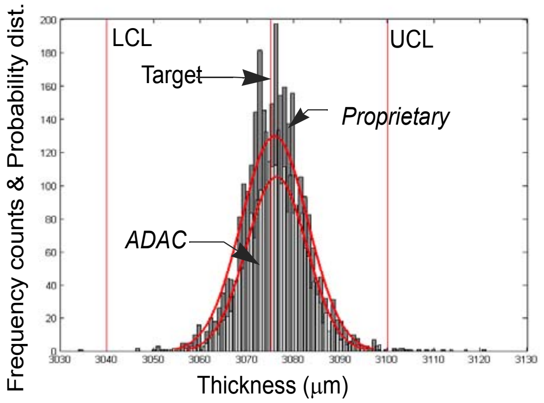

Fig. 10. Unloading thickness distribution plot of the ADAC method compared to the proprietary scheme

6. DISCUSSION AND CONCLUSIONS

A sequential learning ADAC reinforcement methodology has been presented. This is characterised by a learning procedure that combines on-line parameter adaptation and network construction in one integrated routine, thereby automatically determining Action and Critic network sizes. In addition to increasing the autonomy of the overall learning system, the approach also has the advantage of reducing complexity and increasing the computational efficiency of ADAC training, compared to fixed network architecture implementations.

The proposed scheme has been explored as a platform for modelling and control of an industrial grinding process. While the sequential learning ADAC succeeded in identifying a process model from scratch, its slow learning rate meant that even after 2000 iterations of training there were still significant low frequency prediction errors present (MPE of 34%). This was addressed by introducing a "predict correct" scheme. Incorporating the resulting model (MPE of 3.1%) into a DIMC strategy yielded a controller with comparable performance to the proprietary control scheme (12 rejects). The latter is essentially an integral controller whose gain varies as a function of removal rate. Further online optimisation of the controller performance was then undertaken by tailoring the ADAC reinforcement signal to the control objectives. This produced a final controller whose performance was comparable to an MLP based DIMC scheme reported in [12] (8 rejects).

Thus, it may be concluded that while the ADAC learning paradigm may not be appropriate for determining process models and control schemes from scratch, it has the potential of being an effective technique for fine-tuning system performance over an extended period of time, particularly when only limited performance feedback is available.

Ongoing research is aimed at assessing the limitations of ADACs for practical problems in relation to convergence rate, the quality of system excitation needed and the sensitively of performance to several manually-tuned problem specific learning parameters.

7. ACKNOWLEDGEMENTS

The first author wishes to acknowledge the financial support of the Seagate Technology Media (Ireland) Ltd. and Queen's University Belfast.

8. REFERENCES

[1] Åström, K.J., "The future of control", *Journal of Modeling, Identification, and Control*, vol. 15, pp. 127-138, 1994.

[2] Platt, J. C., "A resource-allocating network for function interpolation", Neural Computation, vol. 3, pp. 213-225, 1991.

[3] Kadirkamanathan, V. and Niranjan, M., "A function estimation approach to sequential learning with neural networks", Neural Computation, Vol. 5, no. 6, pp. 954-975, 1993.

[4] Molina, C., and Niranjan, M., "Pruning with replacement on limited resource allocating networks by F-projections", Neural Computation, Vol. 8, no. 4, pp. 855-868, 1996.

[5] Yingwei L., Sundararajan N. and Saratchandran P., "Identification of Time-Varying Nonlinear Systems Using Minimal Radial Basis Function Neural Networks.", IEE Proceedings Control Theory Application, vol. 144, no. 2, pp. 202-208, 1997.

[6] Shiraga, N., Ozawa, S., and Shigeo, A., "A Reinforcement Learning Algorithm for Neural Networks with Incremental Learning Ability", Proc. of the Int. Conf. on Neural Information Processing 2002 (ICONIPS' 2002), Vol. 5, pp. 2566 - 2570, 2002.

[7] Rivest, F., and Precup, D., "Combining TD-Learning with Cascade-correlation Networks", Proc. of the 20th Int. Conf. on Machine Learning (IMCL-2003), pp. 632-639, 2003.

[8] R. E. Bellman, "Dynamic Programming", Princeton, NJ, Princeton University Press, 1957.

[9] J. Si and Y. T. Wang, "Online Learning Control by Association and Reinforcement", IEEE Transactions on Neural Networks, vol. 12, no. 2, pp. 264-276, March 2001.

[10] Ngia L.S.H. and Sjöberg J., "Efficient Training of Neural Nets for Nonlinear Adaptive Filtering using a Recursive Levenberg- Marquardt Algorithm", IEEE Transactions on Signal Processing, vol. 48, no. 7, pp. 1915-1926, July 2000.

[11] McLoone S.F., "Neural Network Identification of AVR Loop Dynamics: A Comparison Between MLPs and RBFs.", CD-ROM Pre-Print 35th Universities' Power Engineering Conference (UPEC 2000), Belfast, U.K., Sept. 6-8, 2000.

[12] Govindhasamy, J. J., McLoone, S. F., Irwin, G. W., Doyle, R. P., and French, J. J., "Neural modelling, control And optimisation of an industrial grinding process", (2004) In press for publication in Control Engineering Practice.