Second-Order Training of Adaptive Critics for Online Process Control

James J. Govindhasamy, Seán F. McLoone, and George W. Irwin

***Abstract*—This paper deals with reinforcement lear ning for process modeling and control using a model-free, action- dependent adaptive critic (ADAC). A new modified recursive Levenberg Marquardt (RLM) training algorithm, called temporal difference RLM, is developed to improve the ADAC performance. Novel application results for a simulated continuously-stirred-tank-reactor process are included to show the superiority of the new algorithm to conventional temporal-difference stochastic backpropagation.**

***Index Terms*—Action-dependent adaptive critic, intelligent control, multilayer perceptrons, neural networks, nonlinear process control, process optimization, reinforcement learning.**

I. Introduction

The design of nonlinear optimal neurocontrollers based on the adaptive critic Ddesign (ACD) paradigm, introduced by Werbos [1], [2], is currently attracting much renewed interest, particularly for complex control applications.

However, closer examination of the current literature suggests that, a number of assumptions have been introduced which run counter to the original ADC concept. Thus, Wu [3], [4], Chan [5], Zeng *et al.* [6], and Riedmiller [7] all assumed *a priori* knowledge of the plant in selecting the control action. Ernst *et al.* [8], Park [9], [10], Venayagamoorty *et al.* [11]–[13], Iyer and Wunsch [14], Radhakant and Balakrishan [15], Sofge and White [16] trained their neurocontrollers offline using a model of the plant. While Hoskin and Himmelblau [17] successfully implemented an online model-free adaptive heuristic critic (AHC) [18], [19] architecture, the control was constrained to be bang-bang.

The fundamental solution to sequential optimization problems uses Bellman's Principle of Optimality [20]. This principle is applied to ACDs 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

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

where γ is a discount factor $(0 < \gamma < 1)$, which determines the importance of the present reward relative to future rewards. 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 maximize $J(k)$, also called the cost-to-go. Unfortunately, a direct solution to this optimization task is computationally infeasible due to the backward numerical solution process involved and the associated "curse of dimensionality". Thus, there is a need for more tractable approximation methods based on a recursive form of (1)

$$J(k) = r(k) + \gamma J(k+1). \tag{2}$$

Werbos proposed a variety of methods to estimate the function $J(k)$ in (2) using artificial neural networks as function approximators.

This paper investigates the use of a model-free, action-dependent adaptive critic (ADAC) developed by Si and Wang [21] for online modeling, control, and optimization of a nonlinear process. In [21], a "batch style" online training method was used, whereby the input and output data were stored during a successful run, and training of the networks was performed only when a failure was encountered. This, however, does not represent true online adaptive control. Their training algorithm is extended here so that it can be implemented online, with adaptation occurring at each sample instant. This eliminates the need to store the plant data and caters for immediate changes in the system dynamics. Further, Si and Wang's algorithm relied on temporal-difference [22] stochastic backpropagation (TD-SBP), which like its batch counterpart, has poor convergence properties and is susceptible to parameter shadowing when used on-line. The latter occurs when the output correctly tracks the desired output with the network parameters continuously adapting instead of converging [23]. An extended version of the recursive Levenberg-Marquardt (RLM) algorithm [24], called the temporal difference RLM (TD-RLM), is proposed to overcome such problems. Identification and control results from a simulated continuous-stirred-tank-reactor process, the first reported application of second order training methods to on-line reinforcement learning, confirm the advantages of our new approach.

The paper is organized as follows. Section II will give a brief description of the ADAC framework and its neural network implementation. The simulated process application is discussed in Section III, along with the control and identification strategy and simulation results. Conclusions and future work appear in Section IV.

II. Preliminaries

Si and Wang [21] formulated a modified version of (2), where instead of approximating $J(k)$, they proposed that the Critic Network be used to approximate the future accumulated reward-to-go, defined as

$$R(k) = r(k+1) + \gamma r(k+2) + \cdots \tag{3}$$

where $R(k) \triangleq J(k+1)$. This is illustrated in Fig. 1.

The Critic Network can be 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

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

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

$$e_c(k) = \hat{R}(k) - \frac{1}{\gamma}\left[\hat{R}(k-1) - r(k)\right] \tag{5}$$

and is usually referred to as the temporal difference error.

Manuscript received February 18, 2004; revised July 29, 2004 and October 28, 2004. The work of J. J. Govindhasamy was supported by the Seagate Technology Media (Ireland) Ltd. and by Queen's University Belfast. This paper was recommended by Associate Editor C. W. Tao.

J. J. Govindhasamy and G. W. Irwin are with the Intelligent Systems and Control, Research Group, Queen's University Belfast, Belfast BT9 5AH, N. Ireland, U.K. (e-mail: j. govindasamy@ee.qub.ac.uk).

S. F. McLoone is with the Department of Electronic Engineering, National University of Ireland Maynooth, Maynooth, Kildare, Ireland.

Digital Object Identifier 10.1109/TSMCB.2004.843276

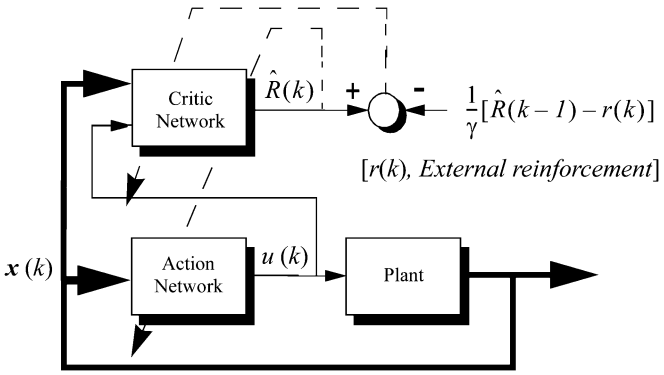

Fig. 1. Schematic of the ADAC scheme.

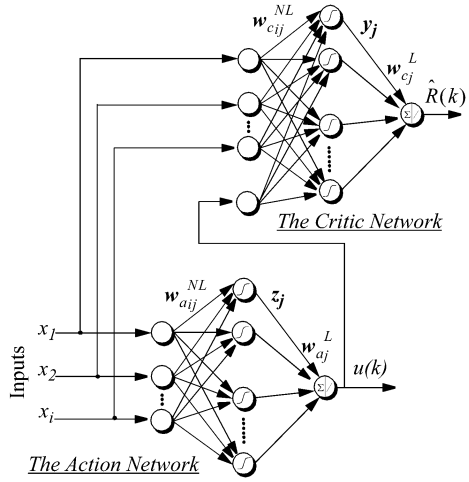

Fig. 2. Action and critic network with the process states, as well as the additional input of the control action, $u(t)$ from the action network to the critic network.

A. Training Method

Training of the Critic Network on the basis of the temporal difference error is equivalent to minimizing the mean-squared error objective function

$$E_c(k) = \frac{1}{2}e_c^2(k). \tag{6}$$

The output of the Critic Network, shown in Fig. 2, is calculated in a feedforward manner and is expressed as follows:

$$g_j(k) = \sum_{i=1}^{N_i+1} w_{cij}^{NL}(k)x_i(k), \quad y_j(k) = \frac{1-e^{-g_j(k)}}{1+e^{-g_j(k)}} \tag{7}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}_c^{NL}$ is the input to hidden layer (or nonlinear) weights, g is the input to the hidden layer nodes, and y is the output of the hidden layer nodes. Note the index N_i+1 is to include $u(k)$ (i.e., $x_{N_i+1} = u(k)$), the output of the Action Network as shown in Fig. 2 below. For this MLP network, the output $\hat{R}(k)$ is calculated as

$$\hat{R}(k) = \sum_{i=1}^{N_h} w_{cj}^{L}(k)y_j(k) \tag{8}$$

where $\boldsymbol{w}_c^L$ is the linear (or hidden to output layer) weight vector. The Action Network update is based on the prediction error given as

$$e_a(k) = \hat{R}(k) \tag{9}$$

where the objective function to be minimized is

$$E_a(k) = \frac{1}{2}e_a^2(k). \tag{10}$$

The output of the Action Network shown in Fig. 2 is calculated in a feedforward manner as follows:

$$h_j(k) = \sum_{i=1}^{N_i} w_{aij}^{NL}(k)x_i(k), \quad z_j(k) = \frac{1-e^{-h_j(k)}}{1+e^{-h_j(k)}} \tag{11}$$

$$u(k) = \sum_{j=1}^{N_h} w_{aj}^{L}(k)z_j(k). \tag{12}$$

Here, h_j and z_j are the input and output of the jth hidden layer neuron activation function $\boldsymbol{w}_a^{NL}$ and $\boldsymbol{w}_a^L$ are the hidden layer and output layer weight vectors and $\boldsymbol{x}$ is the input vector.

The proposed weight-update rule for the Critic and Action Networks is based on a modified version of the RLM formulation [24], with temporal-difference (TD) learning [22] incorporated. The RLM formulation is as follows:

$$S(k) = \alpha_k\Lambda(k) + \Omega^T(k)P(k-1)\Omega(k) \tag{13}$$

$$P(k) = \frac{1}{\alpha_k}\left[P(k-1) - P(k-1)\Omega(k)S^{-1}(k) \times \Omega^T(k)P(k-1)\right] \text{with}$$

$$P(k) = \frac{1}{trace[P(k)]}P(k) \tag{14}$$

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + P(k)(\nabla\psi[\boldsymbol{w}(k)])e(k) \tag{15}$$

$$\nabla\psi[\boldsymbol{w}(k)] = \frac{\partial}{\partial \boldsymbol{w}}f[\boldsymbol{w}(k), \boldsymbol{x}(k)] \tag{16}$$

$$\Omega^T = \begin{bmatrix} & \nabla\psi[\boldsymbol{w}(k)] & \\ 0\ldots & 1 & \ldots 0 \end{bmatrix}$$

$$\Uparrow$$

$$position = k \bmod (N_w) + 1 \tag{17}$$

and

$$\Lambda(k) = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix}^{-1}. \tag{18}$$

Here, $P(k)$, the inverse of the Gauss–Newton Hessian, can be interpreted as the covariance matrix of the weight estimate $\boldsymbol{w}(k)$. The scalar ρ in the 2×2 regularization matrix $\Lambda(k)$ controls the amount of regularization, while $\Omega(k)$ is a $N_w \times 2$ matrix that determines which weight is regularised at a given iteration. Its first column is the gradient vector $\nabla\psi[\boldsymbol{w}(k)]$ while its second column is a vector with one nonzero element that changes position at each iteration as illustrated.

The prediction error is defined as (5) for the Critic Network and (9) for the Action Network. The gradient vectors for both the networks are as follows:

$$\nabla\psi\left[w_{cj}^{L}(k)\right] = \frac{\partial E_c(k)}{\partial \hat{J}(k)}\frac{\partial \hat{J}(k)}{\partial w_{cj}^{L}(k)}$$

$$\nabla\psi\left[w_{cj}^{NL}(k)\right] = \frac{\partial E_c(k)}{\partial \hat{J}(k)}\frac{\partial \hat{J}(k)}{\partial y_j(k)} \times \frac{\partial y_j(k)}{\partial g_j(k)}\frac{\partial g_j(k)}{\partial w_{cij}^{NL}(k)}$$

$$\nabla\psi\left[w_{aj}^{L}(k)\right] = \frac{\partial E_a(k)}{\partial \hat{J}(k)}\frac{\partial \hat{J}(k)}{\partial u(k)}\frac{\partial u(k)}{\partial w_{aj}^{L}(k)}$$

$$\nabla\psi\left[w_{aij}^{NL}(k)\right] = \frac{\partial E_c(k)}{\partial \hat{J}(k)}\frac{\partial \hat{J}(k)}{\partial u(k)}\frac{\partial u(k)}{\partial z_j(k)} \times \frac{\partial z_j(k)}{\partial h_j(k)}\frac{\partial h_j(k)}{\partial w_{aij}^{NL}(k)}. \tag{19}$$

TABLE I
CSTR PROCESS PARAMETERS[25], [26]

Variables	Description	Nominal Values
q_{if}	Product flow rate	100 l/min
C_{if}	Input product concentration	1 mol/l
T_{if}	Input temperature	350K
T_{cf}	Coolant Temperature	350K
V	Container volume	100 l
E/R	Activation energy term	10^4 K
K_0	Reaction rate constant	$7.2\text{x}10^{10}$/min
K_1	Plant constant	1.44 x 10^{13} K l/min/mol
K_2	Plant constant	0.01/l
K_3	Plant constant	700 l/min

III. CASE STUDY

ADAC modeling and control of the highly nonlinear, continuous stirred tank reactor (CSTR) process ([25], [26]) is used to compare the performance of the new TD-RLM algorithm with the conventional TD- SBP training algorithm. The CSTR consists of an irreversible, exothermic reaction, in a constant volume reactor cooled by a single coolant stream. This inherently nonlinear process requires adaptive controllers to cope with its time-varying plant dynamics. The CSTR has been used as a benchmark for numerous nonlinear control strategies for process control over the past 30 years. The plant considered here is single- input-single-output (SISO), where the output is the product concentration C, the input being the coolant flow rate q_c. The chemical reaction which produces the compound takes place inside the insulated tank. This is an exothermic process, which raises the temperature inside the tank, and reduces the reaction rate. The objective is to control the measured concentration of the product C by manipulating the coolant flow rate q_c.

The physical equations of the simulated process are as follows:

$$\frac{dT}{dt} = \frac{q_{if}}{V}(T_{if} - T) + K_1 C \cdot e^{(-E/R\cdot T)} + K_2 q_c \cdot \left[1 - e^{(-K_3/q_c)}\right] \cdot (T_{cf} - T) \tag{20}$$

$$\frac{dC}{dt} = \frac{q_{if}}{V}(C_{if} - C) - K_a C \cdot e^{(-E/R\cdot T)}. \tag{21}$$

The plant constants, which include the heat of reaction, specific heats, liquid densities, and heat transfer terms in simplified forms are summarized in Table I.

This model was simulated at a nominal operating point of $q_c = 100\ \text{l/min}$, $T = 441.22$ K, and $C = 0.0882\ \text{mol/l}$. The simulated input and output are the deviations from these nominal values.

A. Modeling the CSTR

The identified CSTR model was based on a second-order NARX representation using a (4,10,1) MLP network with the output defined as

$$\hat{C}(k) = f\{C(k-1), C(k-2), q_c(k-1), q_c(k-2)\}. \tag{22}$$

Training data in the form of sampled input and output was generated by using a sequence of random steps with superimposed Gaussian white noise. The ADAC scheme is trained to identify the dynamics of the CSTR process as depicted in Fig. 3.

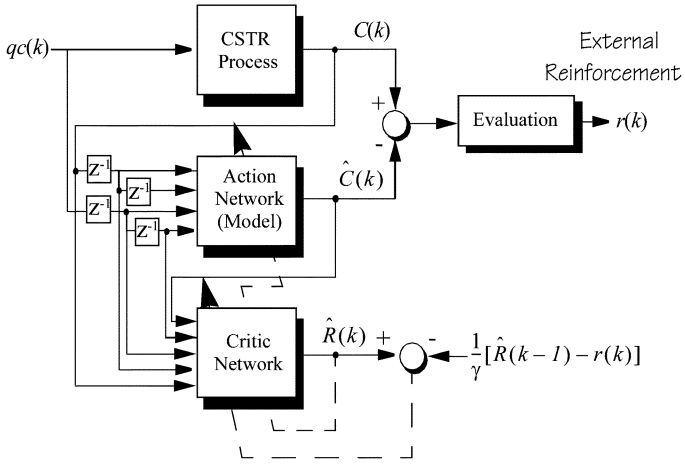

Fig. 3. Schematic of ADAC modeling strategy of the CSTR.

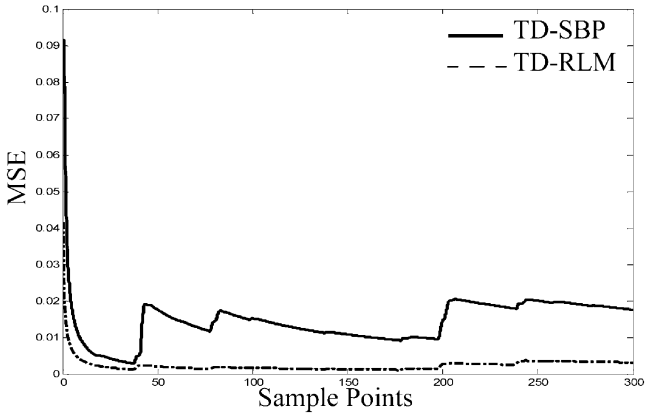

Fig. 4. Training error comparison between TD-SBP and TD-RLM.

The external reinforcement signal for the ADAC model was chosen as

$$r(k) = \begin{cases} -1, & |\text{Prediction Error}| \geq 1 \times 10^{-3} \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

The training mode was terminated once the model was able to maintain this performance for at least 100 sample instances (i.e., to reach the goal after 5 min of a 10-min step time interval). The weights were then stored for the remainder of the training data sequence. Fig. 4 compares the mean squared error (mse) learning curves of the TD-RLM and TD-SBP training algorithms and clearly demonstrates the superiority of the former.

The accuracy of the predicted product concentration was measured in terms of the percentage normalized mean prediction error (MPE), defined as

$$MPE = \frac{1}{n}\sum_{i=1}^{n} \frac{|C_i - \hat{C}_i|}{\sigma_C} \times 100\% \tag{24}$$

where σ_C is the standard deviation of target output, C, and n is the number of data samples. The TD-RLM model achieved an MPE of 8.3% after the training was stopped at the 495th sample instant compared to the TD-SBP of 13.4%, as shown in Figs. 5 and 6. However, the computation time for the TD-RLM was twice that needed for TD-SBP.

In order to further enhance the accuracy of the model prediction, error feedback can be used to compensate for the low frequency offset in the ADAC model prediction.This "predict-correct" technique uses

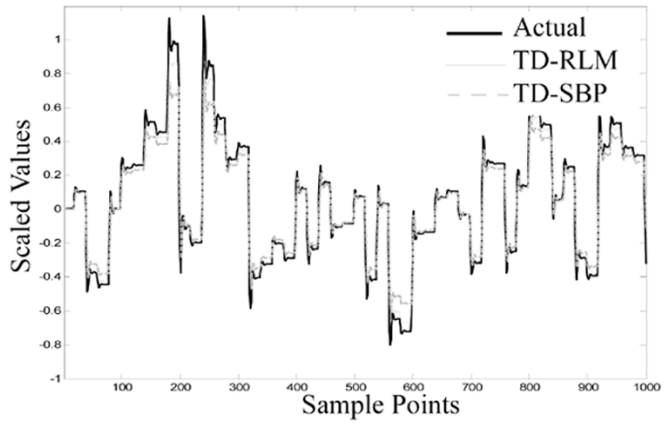

Fig. 5. Modeling prediction comparison between TD-SBP and TD-RLM.

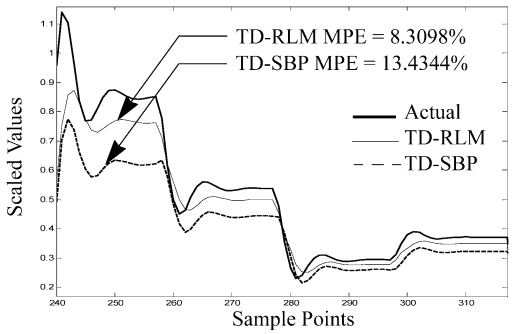

Fig. 6. Zoomed view of the modeling prediction performance between TD-SBP and TD-RLM.

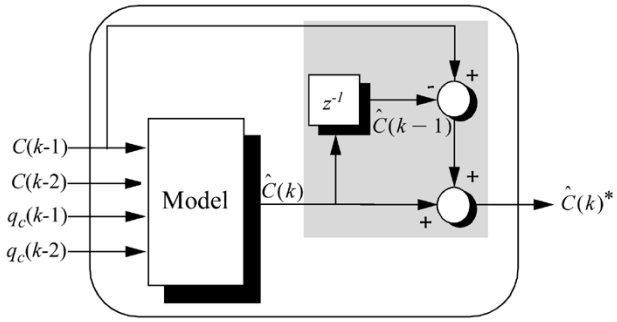

Fig. 7. Predict correct scheme.

past plant outputs and the corresponding model predictions to generate a correction to the current estimate $\hat{C}(k)^*$. Successful applications have been reported in Rovlak and Corlis [27] and Irwin *et al.* [28]. The scheme is usually implemented as follows:

$$\hat{C}(k)^* = \hat{C}(k) + \frac{1}{N}\sum_{j=1}^{N}\Big[C(k-j) - \hat{C}(k-j)\Big]. \tag{25}$$

A first-order predict-correct was incorporated into the ADAC predictor, as shown in Fig. 7. The performance of the ADAC model with predict correct is illustrated in Fig. 8. A clear improvement is observed with the MPE now reduced to 4.2%.

B. CSTR Control

It has been shown that an optimized PI controller of the form

$$q_c(k) = K_p e(k) + K_i T_s \sum_{i=1}^{k} e(i) \tag{26}$$

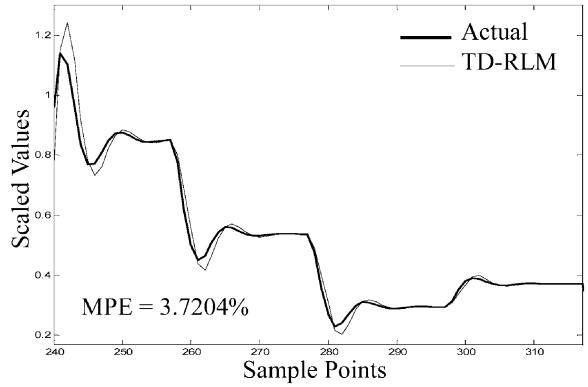

Fig. 8. Prediction performance of the TD-RLM based on the predict correct.

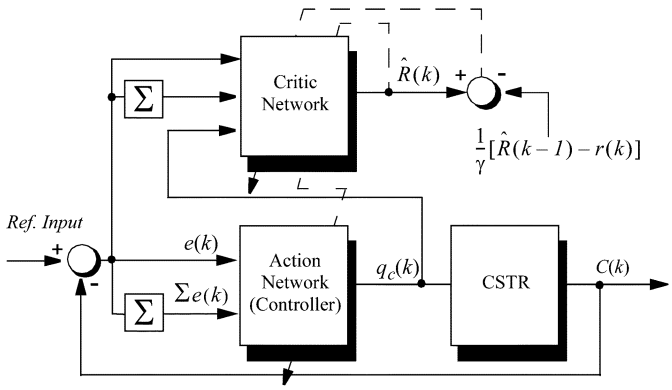

Fig. 9. Block diagram of the ADAC neurocontrol of the CSTR process.

is able to control the CSTR for certain operating regions [25]. Here, T_s is the sampling time, K_p and K_i are the controller gains, and $e(i)$ is the tracking error at the ith sample instant. To provide a direct comparison with the reinforcement learning methodology a nonlinear PI controller was developed based on the ADAC design. The inputs to the Action and Critic networks were chosen to be $e(k)$ and $\Sigma e(k)$, yielding the nonlinear PI controller

$$q_c(k) = f\left\{e(k), \sum_{i=1}^{n} e(k)\right\}. \tag{27}$$

The overall ADAC PI controller structure is thus as depicted in Fig. 9. In this investigation, the plant was fed with random step changes in the concentration set-point at 5-min intervals to simulate different concentration settings. The simulation studies were based on 50 runs, each consisting of 50 min, during which the Action Network had to control the concentration within the operating region using a (2,8,1) MLP network. For each run, the weights of the networks were initialized randomly. The external reinforcement signal for the ADAC controller was chosen as

$$r(k) = \begin{cases} -1, & |e(k)| \geq 5 \times 10^{-4} \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

Training was performed continuously to allow adaptation across the different operating regions of the process. The performance of both training algorithms was measured in terms of the percentage of runs that were successful. The TD-RLM trained controller achieved a 100% success rate while the TD-SBP controller only managed 6%. As expected, the computation time for TD-RLM was 30% more than needed by TD-SBP. Fig. 10 shows a typical successful run, where the TD-RLM ADAC controller performs almost as well as the optimized conventional PI controller and significantly better than the TD-SBP trained

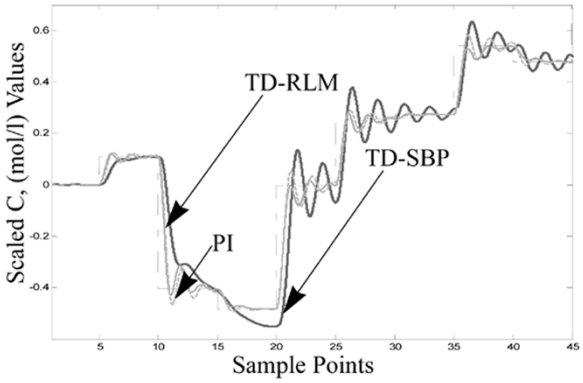

Fig. 10. TD-RLM and TD-SBP controllers tracking comparison with the conventional PI controller.

controller. A useful qualitative performance measure for the control effort is the sum of the square of the control actions $u(k)$

$$I = \sum_{i=1}^{k} [q_c(i)]^2 . \tag{29}$$

For the portion of the tracking response shown in Fig. 10, the TD-RLM trained ADAC controller yielded a control effort of 11.49 compared to 13.76 for the TD-SBP trained controller and 12.10 for the optimized PI controller.

IV. Conclusions

This paper extended the model-free ADAC design of Si and Wang to produce a fully online neurocontroller which avoids the necessity to store plant data during a successful run. In particular, the potential limitations of their stochastic backpropagation training, in terms of poor convergence and parameter shadowing were avoided by introducing an extension to the RLM algorithm called the TD-RLM.

The improved performance of the new ADAC scheme, for both identification and control, was confirmed using results from a simulated continuous-stirred-tank-reactor process. The goal of tracking the reference input to within an acceptable region was also realized. The adaptive critic was shown to be capable of tuning a nonlinear PI controller without any manual intervention, thereby producing comparable performance to an optimized PI controller. This constitutes the first reported application of second order training methods to on-line reinforcement learning and is a significant step forward in moving reinforcement learning toward industrial applications.

References

[1] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*, W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: MIT Press, 1990, pp. 67–95.

[2] ——, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York: Van-Nostrand Reinhold, 1992, pp. 493–525.

[3] Q. H. Wu and A. C. Pugh, "Reinforcement learning control of unknown dynamic systems," *Proc. Inst. Elect. Eng. D*, vol. 140, pp. 313–322, 1993.

[4] Q. H. Wu, "Reinforcement learning control using interconnected learning automata," *Int. J. Control*, vol. 62, no. 1, pp. 1–16, 1995.

[5] K. H. Chan, L. Jiang, P. Tiloston, and Q. H. Wu, "Reinforcement learning for the control of large-scale power systems," in *Proc. 2nd Int. Symp. Engineering of Intelligent Systems (EIS'2000)*, Paisley, U.K., 2000.

[6] X. Zeng, J. Zhou, and C. Vasseur, "A strategy for controlling nonlinear systems using a learning automaton," *Automatica*, vol. 36, pp. 1517–1524, 2000.

[7] M. Riedmiller, "Concepts and facilities of a neural reinforcement learning control architecture for technical process control," in *Neural Computing and Applications*. London, U.K.: Springer-Verlag, 1999, vol. 8, pp. 323–338.

[8] D. Ernst, M. Glavic, and L. Wehenkel, "Power system stability control: Reinforcement learning framework," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 427–436, Feb. 2004.

[9] J. W. Park, R. G. Harley, and G. K. Venayagamoorthy, "Adaptive critic designs and their implementations on different neural network architectures," in *Proc. Int. Joint Conf. Neural Networks*, vol. 3, 2003, pp. 1879–1884.

[10] ——, "Adaptive critic based optimal neurocontrol for synchronous generator in power system using MLP/RBF neural networks," in *Conf. Record—IAS Annu. Meeting*, vol. 2, 2002, pp. 1447–1454.

[11] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "A nonlinear voltage controller with derivative adaptive critics for multimachine power systems," in *Proc. IEEE Power Industry Computer Applications Conf.*, 2001, pp. 324–329.

[12] ——, "Excitation and turbine neurocontrol with derivative adaptive critics of multiple generators on the power grid," in *Proc. Int. Joint Conf. Neural Networks*, vol. 2, 2001, pp. 984–989.

[13] ——, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 764–773, May 2002.

[14] M. S. Iyer and D. C. Wunsch II, "Dynamic re-optimization of a fed-batch fermentor using adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1433–1444, Nov. 2001.

[15] P. Radhakant and S. N. Balakrishnan, "Proper orthogonal decomposition based optimal neurocontrol synthesis of a chemical reactor process using approximate dynamic programming," *Neural Netw.*, vol. 16, pp. 719–728, 2003.

[16] D. A. Sofge and D. A. White, "Neural network based process optimization and control," in *Proc. IEEE Conf. Decision and Control*, vol. 6, 1990, pp. 3270–3276.

[17] J. C. Hoskins and D. M. Himmelblau, "Process control via artificial neural networks and reinforcement learning," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 241–251, 1992.

[18] C. W. Anderson, "Learning and Problem Solving With Mulitlayer Connectionist Systems," Ph.D. dissertation, Dept. Comput. Inform. Sci., Univ. Massachusetts, Amherst, 1986.

[19] ——, "Strategy Learning With Multilayer Connectionist Representations," GTE Laboratories, Waltham, MA, Tech. Rep. TR87-507.3, 1987.

[20] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.

[21] J. Si and Y. T. Wang, "Online learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.

[22] R. S. Sutton, "Learning to predict by the method of temporal differences," in *Mach. Learn.*, 1988, vol. 3, pp. 9–44.

[23] S. McLoone, "Neural network identification: A survey of gradient based methods," in *IEE Colloq. Optimization in Control: Methods and Applications*, London, U.K., Nov. 1998, Dig. 98/521.

[24] L. S. H. Ngia and J. Sjöberg, "Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg- Marquardt algorithm," *IEEE Trans. Signal Process.*, vol. 48, no. 7, pp. 1915–1926, Jul. 2000.

[25] J. D. Morningred, B. E. Paden, D. E. Seborg, and D. A. Mellichamp, "An adaptive nonlinear predictive controller," *Chem. Eng. Sci.*, vol. 47, no. 4, pp. 755–762, 1992.

[26] ——, "An adaptive nonlinear predictive controller," in *Proc. American Control Conf. (ACC)*, vol. 2, 1990, pp. 1614–1619.

[27] J. A. Rovlak and R. Corlis, "Dynamic matrix based control of fossil power plants," *IEEE Trans. Energy Convers.*, vol. 6, no. 2, pp. 320–326, Jun. 1991.

[28] G. W. Irwin, P. O'Reilly, G. Ligthbody, M. Brown, and E. Swidenbank, "Electrical power and chemical process applications," in *Neural Network Applications in Control*. ser. IEE Control Engineering Series 53, G. W. Irwin, K. Warwick, and K. J. Hunt, Eds. London, U.K.: IEE, 1995.