Available online at www.uofk.edu
Proceedings Vol. 1 pp. 9-15
Annual Conference of Postgraduate Studies and Scientific Research
17-20 February 2012-10-17 Friendship
Hall, Khartoum

Optimum Operation and Control of a Batch Chemical Process using Reinforcement Learning

Mustafa, M.A.[1] **and Wilson, J.A.**[2]

[1]*Department of Chemical Engineering, Faculty of Engineering, University of Khartoum*

[2]*Chemical and Environmental Engineering, Faculty of Engineering, University of Nottingham*

Abstract: A Reinforcement learning (RL) approach is presented as a new automatic learning approach to the problem of optimal operation and control of batch chemical processes (e.g. batch reactors and batch distillation columns). The approach is particularly suited to batch process optimization problems, especially through not assuming prior detailed process knowledge or availability of a process model. The particular suitability of RL as a framework for optimising batch process operation has been recognised already (Martinez et al. (1998a,b)). In the implementation of RL, use is made of how the plant responds to control actions aggregated in generalised 'predictive' models, each linking adjacent intra-batch decision steps. The relative worth of a control action at a decision step is aggregated in the 'value function'. Both the predictive models and the value function are learned from the accumulating measurement data, batch-to-batch, starting from a small initial set of test batches. The methodology is exemplified using two simple batch process case studies which were used to test the MATLAB computer coding of the RL algorithm. In addition, issues regarding the structure of the initial training data set and the embedded Neural Network have been investigated.

Keywords:

1. Introduction

Rising importance of high-value-added, low-volume speciality chemicals has resulted in a renewed interest in batch processing technologies (Diewkar, 1995) and the drive for optimum operation is ever present. There is an important literature covering the optimal operation and control of batch chemical processes (Diwekar et al. (1995), Mujtaba and Hussian (1998) and Zhang and Smith (2004),). Although the classical approach to determining efficient time profiles still depends upon having an accurate process model (Aziz and Mujtaba (2002), Mujtaba et al. (2006) and Pommier et al. (2008)), in practice such models are never available partly because conditions and parameters vary from one batch to another. Furthermore, the classical open loop time profile can not react to measurements during the progress of a batch. This is due to the fact that with some of the models available, on-line measurements of the process state are usually scarce and seldom instantaneous (Luyben (1992)). Despite all those problems human operators have managed so far to incrementally drive those processes to near optimal operation. Hence batch process optimization problems fit nicely with RL applications since no knowledge of a process model is assumed by the approach. Furthermore, research in general has focused on the other problems in optimal operation and control besides addressing the central issue of the unavailability of an accurate process model in practice. Hence, the current work was targeted at producing practical solutions to this control problem.

2. Methodology

If an analysis of our learning during childhood is made, we find that (for example) we learn to walk without the help of an explicit teacher. Also learning how to talk, or even how to behave in society when we are growing up. We tend to learn according to trial and error interaction with our environment and then go on reinforcing those actions we took and resulted in better situations. Following this natural process provides us with wealth of knowledge and information about cause and effect, the results of different actions and hence what to do to achieve certain goals.
RL algorithms could be seen as a way of providing a computational approach focused on goal-directed learning and decision making from interaction. Following the book

on the subject by Sutton and Barto (1998) one could define RL, as simply being the mapping of situations to actions so as to maximise a numerical reward gauged through a Performance Index (PI). An important point to add is that during learning, the algorithm is not told which actions to take but must explore and exploit to discover actions that yield the most reward by trying those actions. The RL algorithm tends to learn an optimum control policy by gathering data from a series of batch runs

Batch chemical process problems fit nicely with a typical RL problem, characterised by setting of explicit goals, breaking of problem into decision steps, interaction with environment, sense of uncertainty, sense of cause and effect. The main elements of RL comprise of an agent (e.g. operator, software) and an environment (Sutton and Barto (1998)). The agent is simply the controller, which interacts with the environment by selecting certain actions. The environment then responds to those actions and presents new situations to the agent. The agent's decisions are based on signals from the environment, called the environment's state. Figure 1 shows the main framework of Reinforcement Learning.

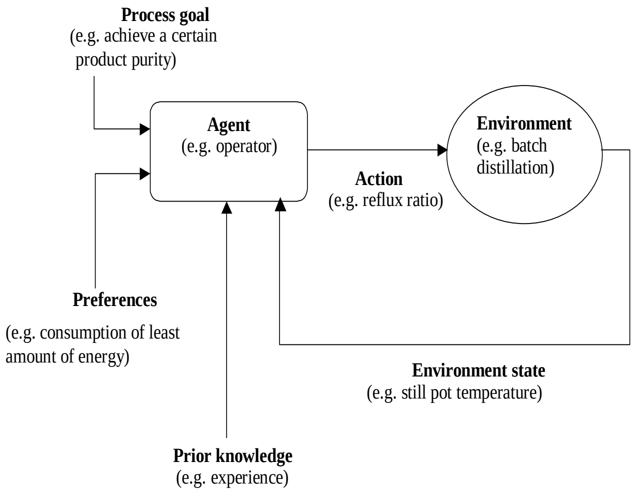
Process goal
(e.g. achieve a certain product purity)
Agent
(e.g. operator)
Environment
(e.g. batch distillation)
Action
(e.g. reflux ratio)
Preferences
(e.g. consumption of least amount of energy)
Environment state
(e.g. still pot temperature)
Prior knowledge
(e.g. experience)

Figure (1). Main framework of Reinforcement Learning.

The RL approach developed in this study is composed of a combination of integrated techniques such as Neural Networks (Carling (1992), Dynamic Programming (Bellman (1957)) and Wire Fitting (Baird and Klopf (1993)). Furthermore, predictive models are used to mimic the forward dynamics of the process.

$$Q(s_t,a_t) = \begin{cases} \leftarrow +PI, \text{ if } a_t \text{ is a final action and the goal has been achieved,} \\ \leftarrow -1., \text{if } a_t \text{ is a final action and the goal has not been achieved,} \\ \leftarrow \max\limits_{a_{t+1}\in\Omega} Q\ (s_{t+1},a_{t+1}), \text{ otherwise.} \end{cases} \tag{1}$$

where PI is the Performance Index. Penalty of -1 is a nominal value. Q (st,at) is a value function for a state action pair.

The main aim of the RL algorithm is to optimize the operation of the process through the following control law:

$$a^* = \arg\left(\max_{a\in\Omega} Q(s,a)\right) \tag{2}$$

where Ω represents the set of feasible control actions.

The RL approach could be seen (i.e. with reference to Wire Fitting approximations) as a means of learning to identify the optimal wire, or wires for the different states. Learning the optimal wire is achieved by learning the weights and biases in the Neural Network. The change in weights (Δ weights) is then calculated as follows:

$$\Delta \text{ weights} = -\alpha \left(\frac{\delta \text{ Mean squared Bellman Error}}{\delta \text{ weights}} \right) \tag{3}$$

where α is referred to as the learning rate.

$$Q^*(s_t,a_t^*) = \begin{cases} \leftarrow PI^*, \text{ if } a_t \text{ is a final action} \\ \leftarrow \max\limits_{a_{t+1}\in\Omega} Q^*(s_{t+1},a_{t+1}), \text{otherwise.} \end{cases} \tag{4}$$

Equation 4 is true only when the RL algorithm converges to the actual optimal value function. During incremental learning of the optimal value function, differences occur which define the error: Bellman error. The mean squared Bellman error (Bellman, 1957), E_B, is then used in the approach to drive the learning process to the true optimal value function (Equation 5 defines E_B for a given state-action pair (s_t,a_t)).

$$E_B = \begin{cases} \leftarrow \frac{1}{2} E\left[\left\{PI^* - Q^*(s_t,\ a_t^*)\right\}^2 \right], \text{ if } a_t \text{ is a final action.} \\ \leftarrow \frac{1}{2} E\left[\left\{ \max\limits_{a_{t+1}\in\Omega} Q^*(s_{t+1}, a_{t+1}) - Q^*(s_t, a_t^*) \right\}^2 \right], \text{otherwise.} \end{cases} \tag{5}$$

3. Case Study

To test the implementation of the RL approach, two case studies were used. The first consisted of non-overlap of target states while the second provided a higher level of complexity with overlap of target states. Thus the approach has to learn how to respond with two different actions given the same state but at different stages.

3.1 Case Study 1: Non-overlap of States in Target Profile Data

A simple batch process was used in order to test the MATLAB implementation of the RL algorithm already developed and to study different aspects of the methodology. The case study involves learning a predefined profile shown in Figure 1, assuming that both sample period and time constant is 1 hour. The relationship between the current state S_t and the one step ahead future state S_{t+1} is according to the first order relationship:

$$s_{t+1} = \alpha s_t + (1-\alpha) a_t \qquad (6)$$

where $\alpha = e^{-1/\tau}$, τ=1hour

Hence, the process is represented by

$$s_{t+1} = 0.36788 s_t + 0.63212 a_t \qquad (7)$$

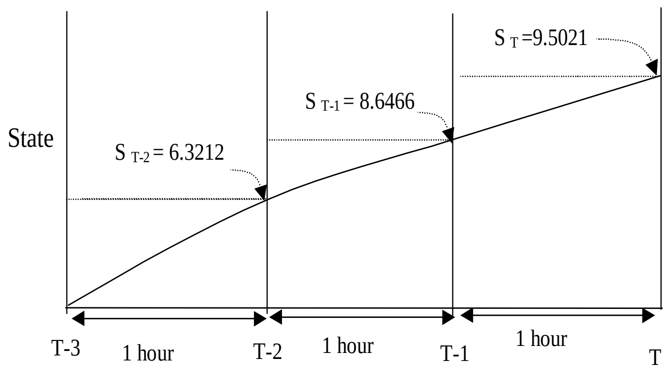

Figure (1). Pre-defined optimum profile (Optimal action = 10 at each decision step)

The Value Function was then defined as

$$PI = 400 - \sum_{t=T-3}^{t=T-1} \left(s_t - s_t^{predefined}\right)^2 \qquad (8)$$

where st is the state at the time interval t and stpredefined is the corresponding state following the predefined profile. Hence, the Performance Index penalises deviations from the profile of predefined states.

3.2 Case Study 2: Overlap of States in Target Profile Data

The second approach towards testing robustness of methodology, when faced with two similar states that require different actions was through Case Study 2. The case study involves the use of identical mid-range states in the target profile. Figure 2 shows how defining similar states at T-2 and T-1 modifies the Case Study 1.

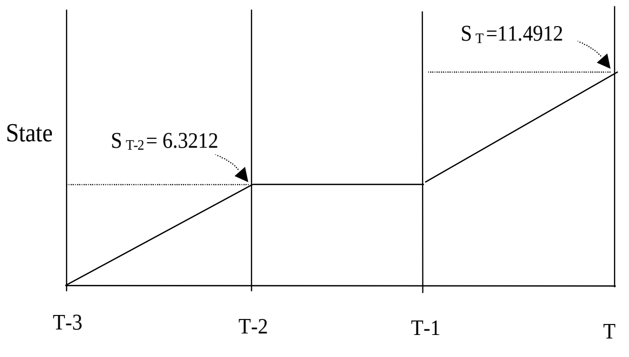

Figure (2). Case study 2 with identical mid-range states in the target profile (optimal action is as follows: 10, 6.3212 and 14.5 at decision step T-3,T-2 and T-1 respectively)

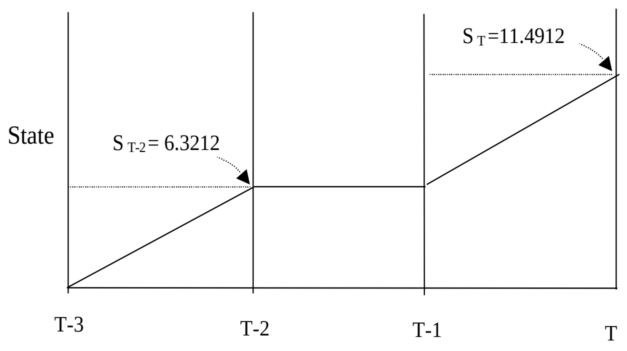

Figure (2). Case study 2 with identical mid-range states in the target profile (optimal action is as follows: 10, 6.3212 and 14.5 at decision step T-3,T-2 and T-1 respectively)

The Value Function was then redefined as

$$PI = 400 - \sum_{t=T-3}^{t=T-1} 8 * \left(s_t - s_t^{predefined}\right)^2 \qquad (9)$$

where st is the state at the time interval t and $s_t^{predefined}$ is the corresponding state following the predefined profile. It can be noticed that a factor of eight is introduced into the definition of the Performance Index. Although the optimization is not affected, the factor was introduced so as to magnify the changes that occur in PI

4. Results and Discussion

Once the case studies were defined, the computer coding of the RL algorithm was applied to test for bugs in the program. Furthermore, investigations were carried out in the following areas of the methodology:

1. Structure of initial training data set.
2. Neural Network Topology.
3. Incremental learning of the Value Function.

4.1 Investigation into Structure of Initial Training Data Set

One of the most important issues in RL applications is the selection of the initial training data set (Neglecting uncertainty in the process at this stage). For computer code debugging (and testing) purposes, 50 non-overlapping batch runs were used as the initial training data set. The purpose of this was to provide enough training data, and to simplify the problem for the RL algorithm. Simplification of the problem lies in the use of non-overlapping data, which would mean that there does not exist similar states at different stages.

Using Case Study 1, learning of the Value Function was performed for states at T-1, then progressed to include states at T-2, and finally to include the initial state (assuming all batches start from the same initial state). This procedure was repeated four separate times using the same training data set but starting from different initial weights in the Neural Network. The results confirmed convergence of the Reinforcement Learning algorithm as shown in Table 1. The third run, shown in Table 1, provided the best result with the lowest sum of squared error of the Neural Network outputs equal to 21.8338, and a value function of 399.978 which is the closest to the optimal value of 400.

Table (1). Performance of methodology while using 50 non-overlapping batch runs as the training data set (best run in bold)

Run number	SSE	Proposed action to take at stage			Value Function
	(Bellman error E_B)	T-3 to T-2	T-2 to T-1	T-1 to T	
1	407.527	11.7672	9.9531	9.9531	398.5958
2	49.7399	10.0676	10.3848	9.9052	399.9298
3	21.8338	10.0122	9.8691	10.2436	399.978
4	61.6385	10.1695	10.2375	10.1412	399.9273

The next step was to try a smaller data set and hence a 20 non-overlapping training data set was used. Again learning of the Value Function was repeated four times using the same 20 batch run training data set, but starting from different initial weights in the Neural Network. Convergence was achieved with the least value of SSE of the Neural Network of 10.23 and a value function of 399.9. The values of SSE for the other 3 runs were 10.9, 139.2 and 159.

Following the convergence of the algorithm so far, the complexity of the case study was increased to test cases where overlapping states occur. The issue of overlapping data deals with the existence of similar states in different stages, and how robust the methodology is in providing the appropriate action for the same state at different stages. The following two approaches were used:

1. Use of an overlapping training data set
2. Use of a different case study (Case Study 2)where overlapping states occur in the target profile data

Starting with the first approach, an overlapping training data set of 50 batch runs was used and the algorithm was executed four separate times, starting from different initial random weights in the Neural Network. The results produced are shown in Table 2.

Table (2). Performance of methodology while using an overlapping data set of 50 batch runs (the best case run is in bold font)

Run number	SSE (Bellman error E_B)	Proposed action to take at stage			Value Function
		T-3 to T-2	T-2 to T-1	T-1 to T	
1	90.7925	9.2535	10.8504	9.7951	399.6448
2	16.7022	10.2081	9.3909	10.1042	399.866
3	62.8841	9.8576	10.7945	9.7459	399.7716
4	85.7716	9.8926	10.6302	9.963	399.843

Since, the optimal value function is 400, it is clear from the results that the algorithm managed to learn the Value Function (up to this point) using both overlapping and non-overlapping data. The second run provided the lowest sum of squared error of the Neural Network equal to 16.70. Furthermore, an overlapping training data set of 30 batch runs was used and applied to Case Study 2 with overlap of states in target profile data. Convergence was still achieved with the lowest SSE value of 21.19. The results prove that the RL algorithm is able to differentiate between different actions to take regarding similar states in different stages.

The next step into the investigation of the initial training data then revolved around determining how the number of batch runs in the initial training data set affects the learning of the Performance Index. Again, using Case Study 2, the algorithm was executed using different initial training data sets ranging from 3 to 30 batch runs. The algorithm was repeated four separate times (for each data set) starting from different initial random weights in the Neural Network. Figure 3 shows the average value of the performance (averaged over the four runs) as a function of the number of batch runs in each initial data set. All cases achieve near optimal values with the exception of using three batch runs, where the algorithm fails to converge. Hence, the use of three initial batch runs could be suggested as an initial starting point for later implementation of the full incremental RL algorithm to Case Study 2.

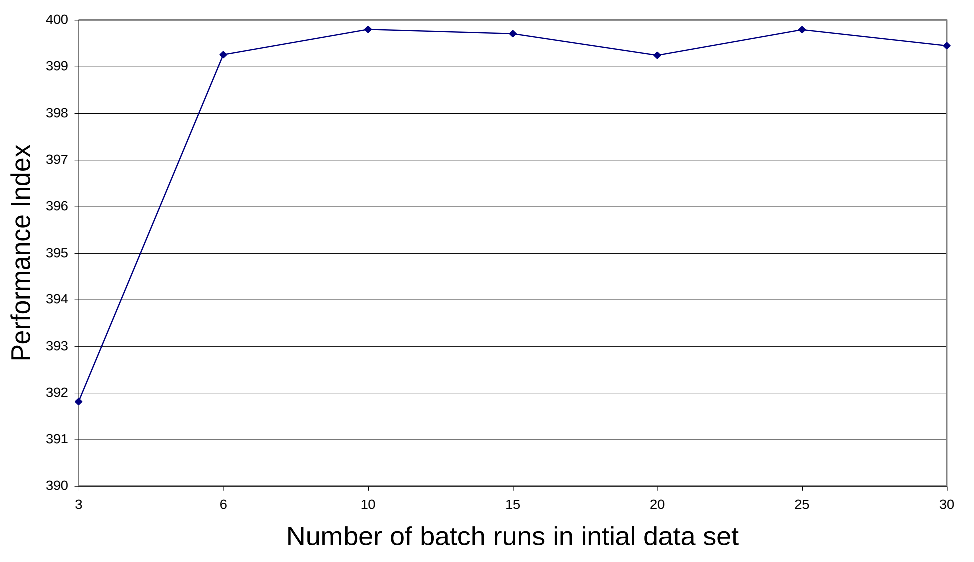

Figure (3). Average Performance Index Vs number of batch runs in initial data set for Case Study 4.2

4.2 Investigation into Neural Network Topology

Case Study 2 was used to investigate the effect of the Neural Network Topology on the learning of the Value Function. The use of three wires for the Value Function approximation fixes the number of output nodes in the Neural Network to three nodes. As for the number of nodes and hidden layers, a simple Neural Network with one node in the hidden layer was used followed by a gradual increase in the complexity of the NN. Using a training data set of 50 random batches, the algorithm was repeated three separate times (starting from different initial training weights in the Neural Network) for different Neural Network architectures. The number of nodes used ranged from 1 to 8 where any increase in the complexity of the Neural Network was terminated due to the sum of squared error of the NN reaching extremely high values. The results in Figure 4 show the log of the sum of squared errors of the Neural Network for the best case when different numbers of nodes in the hidden layer is used. The results show that the use of four nodes in the hidden layer gives the best performance (lowest SSE) for Case Study 2.

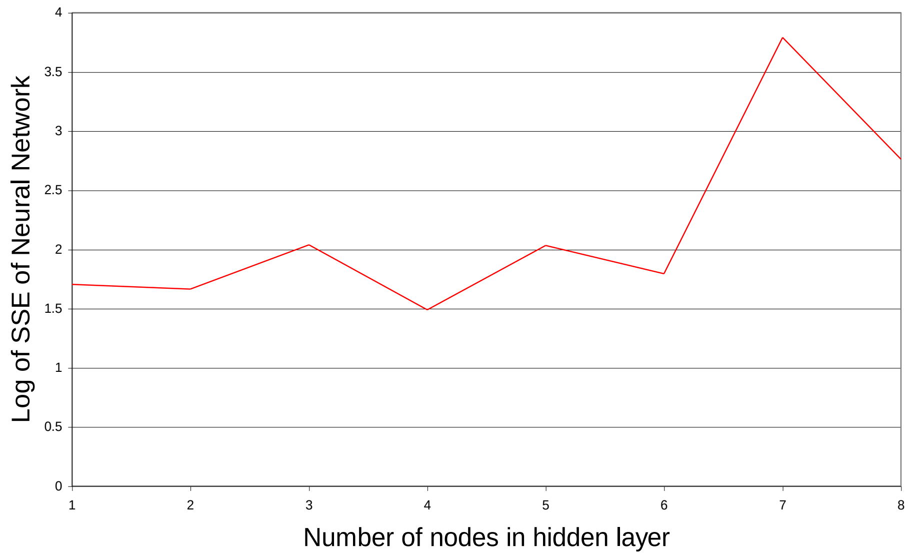

Figure (4). Effect of number of nodes in hidden layer on the performance of the Neural Network

4.3 Incremental Learning of the Value Function

Up to this point the Reinforcement Learning algorithm had only been partially implemented by restricting the algorithm to learn only from a pre-set number of initial batch runs. From this stage onwards, the full RL algorithm is implemented with incremental batch to batch learning. Incremental Learning refers to the execution of the following steps: Step 1 (Start with a small initial training data set), Step 2 (Learn the Value Function for the current data set), Step 3 (Calculate and implement new control profile of manipulated variables during course of next new batch run), Step 4 (Add data from the new batch run to the current training data set), Step 5 (Repeat Steps 2, 3, and 4 until a convergence criterion is met).

Previous results on the number of training data batches (Figure 3) show that learning was not complete for the case with only three initial batch runs. Hence, an initial training data set of three batch runs was used before implementing the full incremental RL algorithm to Case Study 2. Figure 4 shows how the RL algorithm manages to incrementally learn the Value Function, over subsequent batches, converging to a value of 399.98.

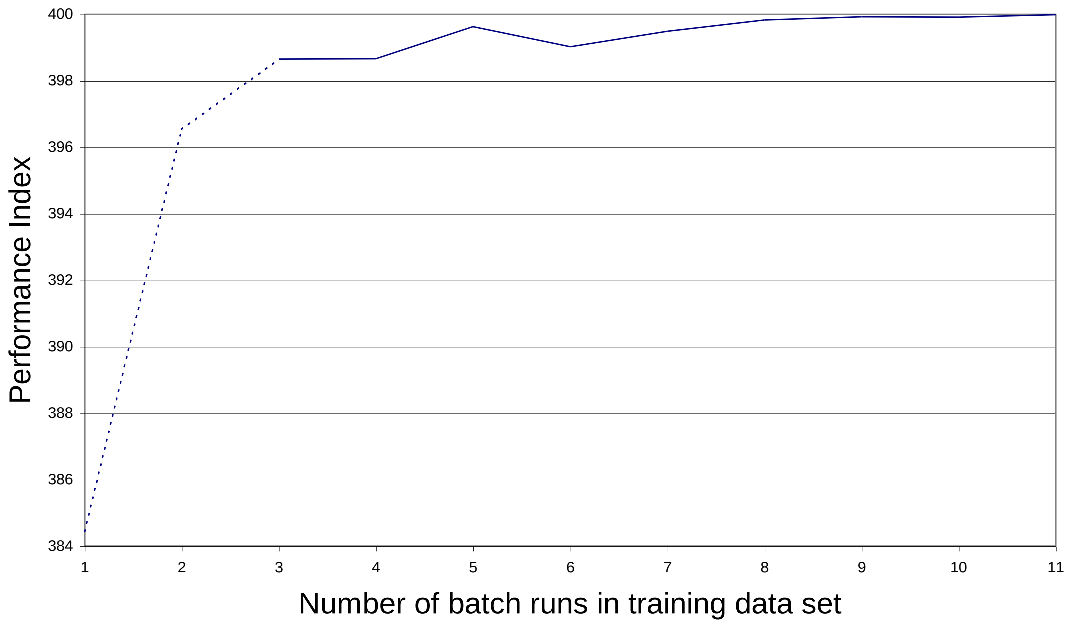

Figure (4). Incremental improvement in Performance Index with increase in number of batch runs for Case Study 2 (The dotted line refers to the Performance Index values of the 3 initial batch runs)

5. Conclusion

The simple batch process case studies presented excellent candidates for testing the RL algorithm. In addition, different layers of complexity were easily added in order to gain insight into various issues regarding the RL algorithm. The simple case studies have allowed the computer coding in MATLAB, of the RL algorithm, to be thoroughly debugged and tested. As for the RL algorithm, it was shown to be able to identify the appropriate action required when faced with similar values of states at different stages. Finally, concerning the Neural Network topology, the use of four nodes in the hidden layer gave the best results for Case Study 2. Also a minimum of 3 batches for the training data set were required. Hence, a minimum of four nodes in the hidden layer and 3 batches for the training data set could be suggested for use in future RL applications.

Notation

a_t	control action at time t
E	squared error
E_B	mean squared Bellman error
$Q(s_t, a_t)$	Value Function for state action pair at time t
PI	Performance Index
RL	Reinforcement Learning
s_t	Process state at time t
α	learning rate
Ω	set of feasible control actions

Subscripts and Superscripts

T	Time
T	final time step
*	Optimum

REFERENCES

[1] Aziz, N., Mujtaba, M. 2002, 'Optimal operation polices in batch reactors', Chemical Engineering Journal, vol. 85, no. 2-3, pp. 313-325.

[2] Baird, L.C. and Klopf, A.H. 1993, Reinforcement Learning with High-dimensional Continuos Actions, Technical Report WL-TR-93-1147, Wright Laboratory, Wright Patterson Air Force Base.

[3] Bellman, R. 1957, Dynamic Programming, Princeton University, Press, Princeton, New Jersey.

[4] Carling A. 1992, Introducing Neural Networks, SIGMA Press, UK

[5] Diwekar, U.M. 1995, Batch Distillation: Simulation, Optimal Design and Control, Carneige Mellon University, Pittsburg, Pennsylvania.

[6] Luyben, W.L. 1992, Practical distillation control. Van Nostrand Reinhold, New York, USA.

[7] Martinez, E.C, Pulley, R.A., and Wilson, J.A. 1998a, 'Learning to Control the Performance of Batch Processes', Chemical Engineering Research & Design, vol. 76(A6), pp. 711-722.

[8] Martinez, E.C, and Wilson, J.A. 1998b, 'A Hybrid Neural Network First Principles Approach to Batch Unit Optimisation', Computer & Chemical Engineering, Suppl. 22:S893-S896.

[9] Mujtaba, I.M, Aziz, N., Hussain, M.A 2006, 'Neural Network Based Modelling and control in Batch Reactor', Chemical Engineering Research and Design, vol. 84, no. 8, pp. 635-644.

[10] Mujtaba, I.M. and Hussain, M.A. 1998, 'Optimal Operation of Dynamic Processes Under Process-Model Mismatches: Application to Batch Distillation', Computers & chemical. Engineering, vol. 22, Suppl., S621-S624.

[11] Pommier, S., Massebeuf, S., Kotai, B., Lang, P., Baudouin, P., Floquet, P., Gerbaud, V. 2008, 'Hetrogenous batch distillation processes: Real system optimisation', Chemical Engineering and Processing: Process Intensification, vol. 47, no. 3, pp. 408-419.

[12] Sutton, R.S. and Barto, A.G. 1998, Reinforcement Learning: An Introduction, The MIT Press, Cambridge, Massachusetts, London, England.

[13] Zhang, J., Smith, R. 2004, 'Design and optimisation of batch and semi-batch reactors', Chemical Engineering Science, vol. 59, no. 2, pp. 459-478.