A Direct Utility Adaptive Critic (DUAC) algorithm for power plant load management

Udhay Ravishankar, *Member IEEE*, and Milos Manic, *Member IEEE*
University of Idaho, Idaho Falls
ravi4736@vandals.uidaho.edu, misko@ieee.org

***Abstract*- This paper presents a Direct Utility Adaptive Critic (DUAC) algorithm applied to a power plant load management problem. The DUAC algorithm is an enhancement of the original Heuristic Dynamic Programming (HDP) Adaptive Critic Design (ACD) algorithm into a simpler and more robust controller. Typical ACD algorithms model dynamic systems with time-delayed states and action inputs and due to this the Action Network training procedure is a complex BackPropagation-Through-Time (BPTT) process. Also required in typical ACD algorithms is a dedicated Critic Network training process for different control sequences before the Action Network training procedure. The DUAC algorithm, presented in this paper, simplifies the Adaptive Critic algorithm by 1) eliminating the complex BPTT process for training the Action Network and 2) replacing the Critic Network with the user-defined utility function directly. Due to these changes the utility-action gradient typically required to train the Action Network is based on direct result of two utility values with respect to two action inputs. The replacement of the Critic Network with the user-defined utility function ensures better control accuracy since Critic Network modeling provides only approximations of the utility function. The DUAC algorithm was tested for time-varying consumer loads on an RMS voltage analogous s-domain model of the power plant created in Simulink using the SimPowerSystems toolbox. Test results indicated that the DUAC algorithm was able to derive an Action Network that controlled the power plant model to an output RMS voltage fluctuation variance of the order of no more than 10^{-3}. This result can prove to be an essential step in load balancing problems in Smart Grids.**

I. INTRODUCTION

Load Management is a generic goal of a Smart Grid to ensure power output stability for any unpredictable load variations in the consumer side. The benefit of maintaining a stable power output response is to minimize the occurrence of instability in the grid caused by such unpredictable load variations. The benefit also propagates into minimized utility bills for abrupt switching of high power loads which is specifically the case of industrial loads.

Load management problems are either tackled in a power grid topological scale [1] – [8] or in a generator scale [9] – [11]. Adaptive Critic Designs have also been used in such problems [7][8][12][13][14]. In this paper we consider the generator scale case because it is a smaller problem and an important step towards tackling bigger problems such as the grid topological case. In our setup, we have created a model power plant with a few consumer loads in Simulink using the SimPowerSystems toolbox. This model helps in visualizing the behavior of power plants for time-varying loads especially when there is a lack in data availability for such behaviors.

The SimPowerSystem model power plant is considered as the actual power plant that provides data regarding power plant behaviors for time-varying loads. Before applying the proposed algorithm, i.e. the DUAC, on the SimPowerSystem model, an s-domain analogous model of a specific behavior of the plant was created. In this paper we consider the RMS voltage response of the SimPowerSystem power plant in developing the s-domain analogous model.

The Direct Utility Adaptive Critic (DUAC) algorithm is designed based on Adaptive Critic Designs (ACD) introduced by Werbos in 1974 [15]. The DUAC is implemented using the Backpropagated Critic Architecture (BCA) shown in Fig. 1 in section II. This architecture is one of the architectures introduced by Werbos in [16] and [17]. Prokhorov and Wunsch have discussed the implementation of adaptive critic algorithms for the BCA and other architectures in [18]. Typical adaptive critic algorithms implemented on ACDs are the Heuristic Dynamic Programming (HDP), the Dual Heuristic Dynamic Programming (DHP) and the Global Dual Heuristic Dynamic Programming (GDHP) listed in ascending order of advancement. The algorithm used for the DUAC is the HDP based.

The rest of this paper proceeds as follows: Section II will discuss a background review on Adaptive Critic Designs followed by the introduction of the DUAC algorithm in section III. The power plant s-domain analogous model will be developed in section IV followed by the discussion of the neural network modeling of the power plant system in section V, which is an important part of the DUAC algorithm. Section VI will show test results of the DUAC algorithm and the paper finally concludes in section VII with future work.

II. THE ADAPTIVE CRITIC DESIGN

Adaptive Critic Design (ACD) belongs to the class of discounted discrete-time Markovian Decision Processes (MDP) [19]. MDPs are typically used in finding the optimal control policy for industrial and commercial processes. In MDP, the optimal control policy is determined based on the Bellman Equation of state transition matrices. The optimal policy in ACDs, on the other hand, is a function known as the Action Network based on the same Bellman Equation, but

using a user defined utility function U as given in Eq. (1) [19].

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

The Bellman function *J(•)* approximates the total utility value of future trials in the process by a discount factor γ. The advantage of using the discount factor is when $k \rightarrow \infty$ the Bellman function reaches a steady state value independent of *k*, meaning that it allows a quick approximation of the utility of future trials without iterating over *k*.

Werbos introduced a couple of ACDs in [16] and [17]. The designs introduced in [16] and [17] are the Classical Critic Architecture and the Backpropagated Critic Architecture. In this section we will review the Backpropagated Critic Architecture as this is the architecture used for the DUAC algorithm.

A. The Backpropagated Critic Architecture (BCA)

The Backpropagated Critic Architecture (BCA) is shown in Fig. 1 [17]. It consists of a Critic Network, a Model Network and an Action Network. The Model Network makes the BCA a more advanced controller than the Classical Critic Architecture (which does not have a Model Network) as it provides the controller an estimate of the future system state R(t) it wishes to control from the current system state R(t-1) and control input u(t-1). The Critic Network is a neural network model of the user-defined utility function that inputs the estimated future system state R(t) through the Bellman Equation, described earlier in Eq. (1), which guides the Action Network. The Action Network is trained to store the optimal policy function using a HDP, DHP or GDHP approach. Since the HDP technique is used for the DUAC, the traditional HDP algorithm will be described in the following subsection.

B. Heuristic Dynamic Programming (HDP) for ACD

The HDP algorithm is the simplest ACD algorithm that requires the Critic Network to model the Bellman Equation of eq. (1). The Critic Network training is performed to derive the gradient $\partial J/\partial A$ (with A representing the set of possible control actions) by backpropagating a one from the output of the Critic Network through the control input of the Model Network, to train the Action Network as given in Eq. (2) [18]. In Eq. (2) the η_A is the learning constant.

$$\Delta W_A = -\eta_A \frac{\partial J}{\partial A} \frac{\partial}{\partial W_A} \left(\frac{\partial J}{\partial A} \right) \tag{2}$$

From the principle of backpropagation, the $\partial J/\partial A$ is minimized when backpropagated into the Action Network.

In the typical HDP algorithm the Critic Network training is performed using the error equation Eq. (3).

$$E = J(t) - \gamma J(t+1) - U(t) \tag{3}$$

The Critic Network training requires different set of control sequences to ensure that it is properly trained.

The DUAC algorithm is a modification of the HDP algorithm that simplifies the optimal Action Network derivation even when time-delayed inputs are present in the Model Network. This is discussed in the following section.

III. THE DIRECT UTILITY ADAPTIVE CRITIC (DUAC)

This section discusses in detail the modifications made on the traditional HDP algorithm into a simplified and yet a more robust control algorithm, the DUAC algorithm.

For the DUAC algorithm, the Critic Network in the traditional HDP algorithm is replaced by the direct use of the user-defined utility function. This provides better control accuracy than the traditional HDP since neural networks provide only approximations of the utility function. The notion of replacing the Critic Network is based on the understanding of the Bellman Equation described earlier in Eq. (1). The understanding is developed by z-transforming the Bellman Equation which is given in Eq. (4).

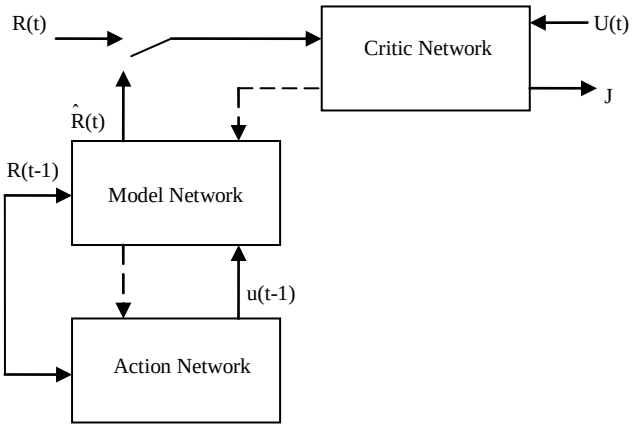
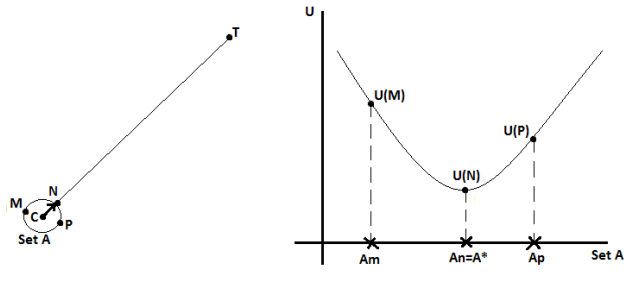


Fig. 1. The Backpropagated Critic Architecture [17].



(a) (b)

Fig. 2. (a) A 2D illustration of finding the optimal control action from a radial set A (the circle around C) at the current system state C. The point N is the optimal point to move to towards the target state T. (b) A graphical view of the utility function U, defined by the target state T, versus the radial set A. The action An in the radial set A is the optimal action A*.

$$J(z) = \frac{zU(z)}{z-\gamma} \tag{4}$$

It is clear from Eq. (4), if γ is set to zero, the Bellman function is equivalent to the utility function itself. For any other $0<\gamma<1$, the Bellman function is a converging total of the future utilities. The DUAC algorithm thus utilizes the property of zero discount factor.

Since the Critic Network is replaced by the utility function directly, the optimal Action Network derivation process uses a modified perspective. The next subsection describes this modification without losing the generality.

A. HDP for the DUAC

To maintain the generality of the HDP algorithm, The DUAC algorithm utilizes a different perspective of the ∂J/∂A gradient discussed earlier in Section II. Fig. 2 describes this perspective. From Fig. 2 the Action Network is trained to find the optimal control input for the dynamic system under its current conditions to maintain the utility function to its minimum. This method of training is particularly useful when the system is subject to disturbances such as load for a power plant. Once the optimal actions are derived for particular disturbance signal, another Action Network can be trained to model the optimal control input as function of the disturbance signal. This latter Action Network can then be used as the controller for the system later on.

In order to derive the optimal control action, it is clear from Fig. 3 that the Action Network must converge to the argument of the minimum (argmin) of the utility function against the set of possible control actions. The Error Backpropagation (EBP) algorithm readily does this in principle if the error is defined as ∂U/∂A, i.e. the Utility-Action gradient, given in Eq. (5). Note that this time U is used instead of J.

$$\Delta W_A = -\eta_A \frac{\partial U}{\partial A}\frac{\partial}{\partial W_A}\left(\frac{\partial U}{\partial A}\right) \tag{5}$$

In this section, we will discuss the Action Network and its training methodology and leave the discussion of the Model Network to a later section. This is because the Model Network is system dependent and not general to the DUAC.

B. The Action Network

The Action Network can be implemented as a single-hidden-layer-single-output-layer Multi-Layer-Perceptron Neural Network as shown in Fig. 3. The activation function for the hidden layer is the logsigmoidal function, or the unipolar soft activation function, to model the nonlinear output of the Action Network. The Action Network topology is system dependent, which will be described later. The weights following the output layer set the velocity, i.e. the rate at which the desired output response of plant should converge. Based on the architecture of Fig. 3, Eq. (4) can be

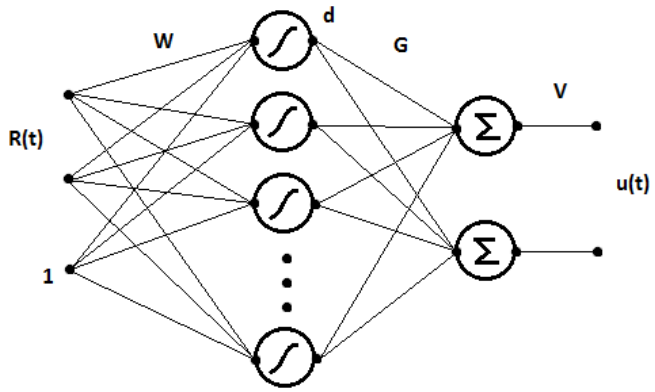


Fig. 3. The Action Network Architecture. The input-hidden layer weights are W, hidden-output layer weights are G and the outputs are weighted by V representing the maximum control velocity bound.

rewritten as Eq. (6). In the Eq. (6), η_A is the learning rate while μ_A is the momentum rate.

$$\Delta G = -\eta_A V\left(\frac{\partial U}{\partial A}\right)d - \mu_A \Delta G$$
$$\Delta W = -\eta_A VG\left(\frac{\partial U}{\partial A}\right)d(1-d)\begin{bmatrix} R(t) \\ 1 \end{bmatrix} - \mu_A \Delta W \tag{6}$$

C. Evaluating the Utility-Action gradient ∂U/∂A

The Action Network is randomly initialized before performing the utility minimization. During the utility

Pseudo Code of the DUAC:

Let the Action Network be denoted as a function $f_A(R(t))$ and the Model Network as $f_M(R(t),u(t))$. Let σ be a constant < 1 and rand ~ uniform random number between 0 and 1.

Phase I: Train Model Network

1) Obtain training data of $R(t)$ and $u(t)$ from simulation of the system under consideration.
2) Create a mapping $R(t+1) = f_M(R(t),u(t))$.
3) Validate the feedback system by connecting the state output of the Model Network to its state input through a time-delay.

Phase II: Train Action Network to find optimal control action

1) Initialize Action Network weights W and G and action $u_0(t)$ = σ•rand.
2) Compute action $u_1(t) = f_A(R(t))$.
3) Compute future states $R_1(t+1) = f_M(R(t), u_1(t))$ and $R_0(t+1) = f_M(R(t), u_0(t))$.
4) Evaluate utilities $U(R_1(t+1))$ and $U(R_0(t+1))$.
5) Compute gradient $\partial U/\partial A = (U(R_1(t+1)) - U(R_0(t+1)))/(u_1(t) - u_0(t))$.
6) Backpropagate $\partial U/\partial A$ through $f_A(\bullet)$.
7) Store $u_0(t) = u_1(t)$ and compute new action $u_1(t) = f_A(R(t))$.
8) Repeat steps 3) to 7) until termination criterion, pick the $u(t)$ with minimum utility $U(R(t+1))$ and dispatch the $u(t)$ to the system.

Phase III: Train Action Network to model the Optimal Controller

1) Initialize Action Network $f_A(R(t),L(t))$ where $L(t)$ is the disturbance signal.
2) Train the Action Network to model the optimal action $u^*(t) = f_A(R(t),L(t))$.

Fig. 4. Pseudo Code for computing Utility-Action gradient for backpropagating.

minimization, the current system state is inputted into the Action Network to give an input control action to the Model Network. Then another control action input is initialized to be some random value. The two control action inputs are inputted into the Model Network to provide estimates of the future system states. The utility values of the two estimated states are evaluated and the gradient ∂U/∂A is computed. This ∂U/∂A is backpropagated through the Action Network which then gives a new control action input for the same current system state. Then a new ∂U/∂A is computed between the new control action input estimate and the previous control action input estimate. This process is repeated until ∂U/∂A converges to zero. The pseudo code for this process is given in Phase II of Fig. 4. Fig. 4 shows pseudo code of the overall DUAC algorithm.

IV. THE TEST SYSTEM: S-DOMAIN MODEL OF THE PLANT

A model power plant was created in Simulink using the SimPowerSystems Toolbox to generate data regarding the behavior of a power plant for time-varying loads. The model consists of a turbine under constant torque, a rectifier, a buck-boost converter, an inverter, a transformer, a PID controller and a few loads. Based on the simulations of the Simulink modeled power plant (a sample shown in Fig. 5) an s-domain analogous model was created.

In Fig. 5 the load profile shows magnitude of three-phase loads randomly switched on and off. The output Root-Mean-Square (RMS) voltage can be seen to drop when the load increases and overshoot when the load decreases. The PID controller varies the duty-cycle ratio of a Pulse-Width-Modulated (PWM) signal, which is the control input of the power plant, to stabilize the output RMS voltage with respect to the reference RMS voltage.

The s-domain analogy model was created to replicate the RMS voltage response of the power plant with the PID controller. It can be clearly noted that the PID controller acts as a restoring force on the output RMS Voltage. Hence a mass-spring-damper system was used to model this behavior and is shown in Fig. 6. To derive the s-domain analogous model of the power plant, the PID controller's s-domain

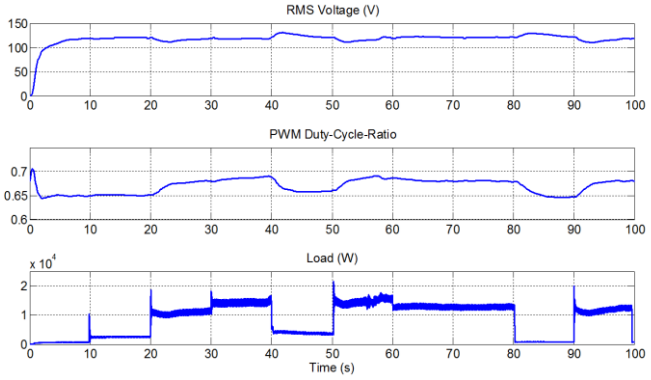


Fig. 5. A sample Simulink simulation of the SimPowerSystem model power plant.

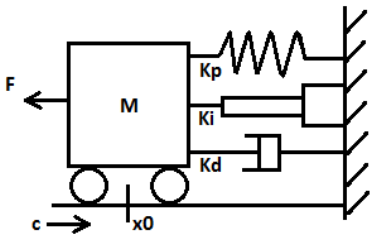


Fig. 6. The generic model of the power plant with a PID control with all non-zero gains (note that the Ki is modeled as an electronically controlled muscle arm).

transfer function was simplified to having only the Integral gain. This was done because the SimPowerSystem model's PID controller was configured in PI configuration and removing the Proportional (P) gain, by setting it to zero, also restored the RMS voltage output of the plant. This response encouraged the simplification for analysis purpose only.

Using the model of Fig. 6 and the aforementioned simplification, the power plant's s-domain transfer function *G(s)* was derived as given by Eq. (7a).

$$G(s) = \frac{1/M}{s + c/M} \tag{7a}$$

The PID controller's s-domain transfer function *H(s)* with only the Integral gain is given by Eq. (7b).

$$H(s) = \frac{K_i}{s} \tag{7b}$$

Then applying the *G(s)* and *H(s)* to the overall feedback system transfer function, the closed loop transfer function was derived as shown in Eq. (8).

$$\frac{V(s)}{L(s)} = \frac{G(s)}{1 + G(s)H(s)}$$
$$\frac{V(s)}{L(s)} = \frac{(1/M)}{s^2 + (c/M)s + (K_i/M)} \tag{8}$$

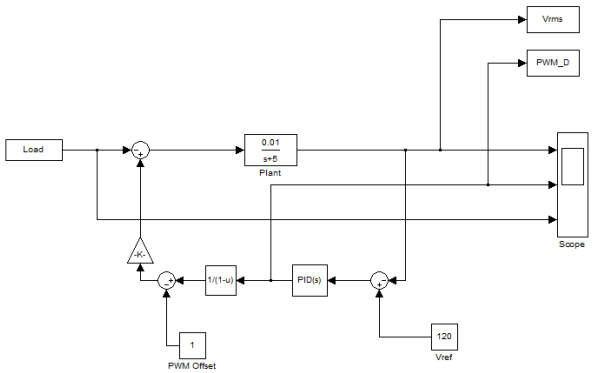


Fig. 7. The s-domain analogous model of the SimPowerSystem's model power plant.

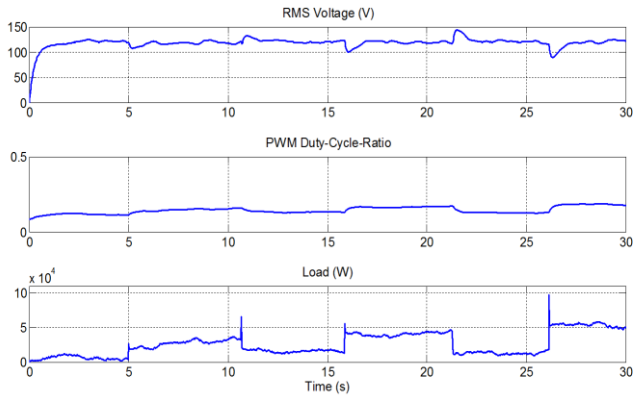


Fig. 8. Simulation of the s-domain analogous model of the SimPowerSystem power plant model and control system.

In Eq. (8), the *V(s)* is the s-domain transform of the output RMS voltage while the *L(s)* is the transform of the consumer load profile. The right-hand-side of the bottommost Eq. (8), when inverse transformed to the time domain has a response equivalent the RMS voltage response of Fig. 5.

Using the above modeling equations, a Simulink model of the SimPowerSystem's model power plant and control system was created as shown on Fig. 7. For neural network modeling purpose, which will be discussed later in section V, the buck-boost converter gain equation was changed to a boost converter gain equation and hence the 1/(1-u) function block following the PID(s) block instead of a u/(1-u) function block in Fig. 7.

A typical plot of the Simulink system in Fig. 7 is shown in Fig. 8. It is worthy to note the similarities of Fig. 8 with Fig. 5.

V. Neural Network Modeling of the Plant

For the DUAC algorithm to optimize the system under consideration, the algorithm must obtain some knowledge about the dynamics of the system. This was the essential idea of the Backpropagated Critic Architecture discussed earlier in section II. In our work we used a neural network to model the dynamics of the plant.

The neural network model topology was designed with respect to the system under consideration. In section IV, the power plant was modeled as a first order Laplace Transfer Function. Hence a single linear neuron with a time-delay state feedback can be used to model the behavior of the power plant. The single neuron with state feedback is expected to contain the discrete-time coefficients of the equivalent continuous-time state-space model of the Laplace Transfer Function. Included with the linear neuron is a non-linear neural network that models the non-linearity of the boost converter. Although the SimPowerSystem power plant has a buck-boost converter, the s-domain analogous model system of Fig. 3 uses a boost converter gain equation. This was done because for a neural network to correctly model the transient dynamics of the continuous-time system, a zero initial condition was required for the system's control signal. Since a buck-boost converter requires a non-zero initial condition on the control signal, the neural network failed to model the transient dynamics of the system in Fig. 7.

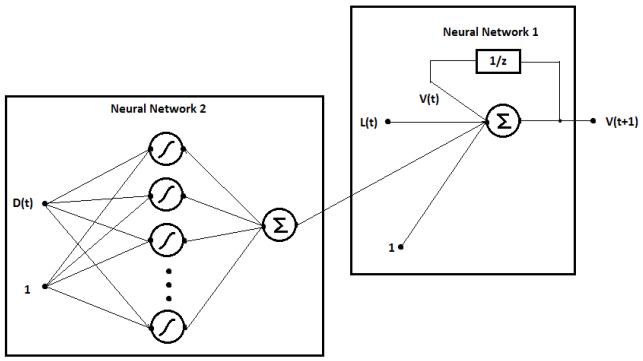


Fig. 9. The Model Network Architecture. D(t) is the PWM Duty-Cycle-Ratio control signal for the boost converter, L(t) is the load signal on the power plant and V(t) and V(t+1) are the current and future RMS voltage states respectively.

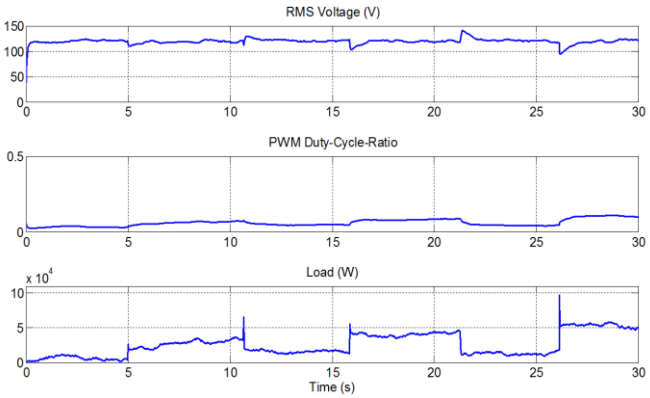


Fig. 10. Simulation for validation of the Model Network for the analogy model system.

Hence the Model Network for the DUAC algorithm consists of two neural networks connected in a cascaded fashion shown in Fig. 9. The Model Network was trained using the EBP algorithm. It can be noted that the Model Network response shown in Fig. 10 does not exactly match the response of the s-domain model, but they are similar. Since this paper mainly focuses on the DUAC algorithm, the trained Model Network was still used for optimization.

VI. Test Results

Figures 11 and 12 show sample test experiments of the DUAC algorithm on the Model Network modeled system. In each experiment the system was subjected to different load profiles varying between 1 to 100 Kilowatts. The DUAC algorithm was able to find the optimal control actions needed to stabilize the RMS voltage output for strong load variations as well as weak load variations. The DUAC algorithm took 20 iterations during each time step to find the optimal control signal. During each time step, the Action Network was reinitialized to enable faster convergence towards the optimal control signal.

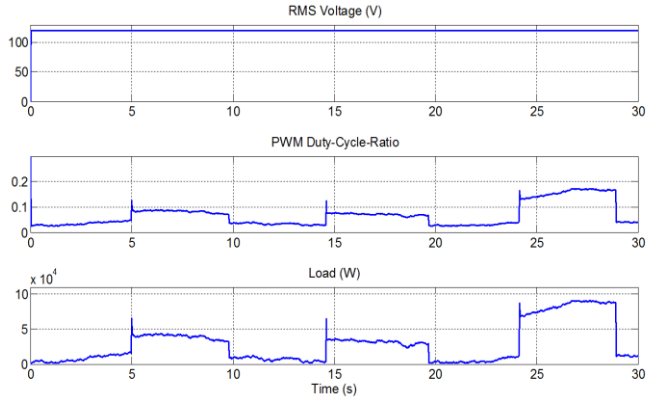


Fig. 11. Test experiment I of the DUAC algorithm on the power plant model.

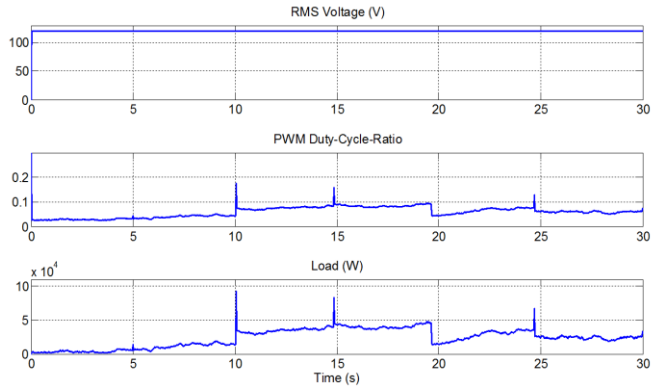


Fig. 12. Test experiment II of the DUAC algorithm on the power plant model.

VII. Conclusions and Future Work

The DUAC algorithm was presented in this paper by modifying the traditional method of ACD into a simplified and more robust controller. The DUAC algorithm was implemented using the Backpropagated Critic Architecture with a modified Heuristic Dynamic Programming (HDP) method of training the Action Network's optimal policy. Two test experiments indicated an output RMS voltage fluctuation variance of no more than 9.7964e-4 excluding the startup time-steps. The results are very encouraging towards using the DUAC algorithm for the SimPowerSystem model power plant as the next step.

VIII. Acknowledgement

The authors acknowledge support for this work from Idaho National Laboratory through the U.S. Department of Energy Office of Electrical Delivery and Energy Reliability under DOE Idaho Operations Office Contract DE-AC07-05ID14517.

References

[1] J. Liang, R. G. Harley and G. K. Venayagamoorthy, "Adaptive Critic Design based Dynamic Optimal Power Flow Controller for a Smart Grid," *Proc. IEEE Symp. on Computational Intelligence Applications in Smart Grids*, 2011, pp. 17 – 24.

[2] V. S. Pappala, M. Wilch, S. N. Singh and I. Erlich, "Reactive Power Management in Offshore Wind Farms by Adaptive PSO," *Proc. Intl. Conf. on Intelligent Systems Applications to Power Systems*, 2007, pp. 1 – 8.

[3] J-C Hwang, J-C Chen, J. S. Pan and Y-C Huang, "CSO and PSO to Solve Optimal Contract Capacity for High Tension Customers," *Proc. Intl. Conf. on Power Electronics and Drive Systems*, 2009, pp. 246 – 251.

[4] C-H Kung, M. J. Devaney, C-M Huang and C-M Kung, "Power Source Scheduling and Adaptive Load Management via a Genetic Algorithm Embedded Neural Network," *Proc. of Instrumentation and Measurement Tech. Conf.*, 2000, vol. 2, pp. 1061 – 1065.

[5] L. Gomes, F. Fernandes, T. Sousa, M. Silva, H. Morais, Z. Vale and C. Ramos, "Contextual Intelligent Load Management with ANN Adaptive Learning Module," *Proc. Intl. Conf. on Intelligent System Application to Power Systems*, 2011, pp. 1 – 6.

[6] H-C Sun, K-Y Huang and Y-C Huang, "Hybrid Intelligent Approach for Load Control and Management," *Proc. Intl. Conf. on Innovative Computing Information and Control*, 2008, pp. 1 – 4.

[7] R. L. Welch and G. K. Venayagamoorthy, "HDP based Optimal Control of a Grid Independent PV System," *IEEE Power Engg. Society Genral Meeting*, 2006, pp. 1 – 6.

[8] S. Mohagheghi, G. K. Venayagamoorthy and R. G. Harley, "Adaptive Critic Design based Neuro-Fuzzy Controller for a Static Compensator in a Multimachine Power System," *IEEE Trans. on Power Systems*, vol. 21, no. 4, pp. 1744 – 1754, Nov. 2006.

[9] C. Yan, G. K. Venayagamoorthy and K. Corzine, "Hardware Implementation of an AIS-Based Optimal Excitiation Controller for and Electric Ship," *IEEE Trans. on Industry Applications*, vol. 47, no. 2, pp. 1060 – 1070, Mar./Apr. 2011.

[10] L. Karunarathne, J. T. Economou and K. Knowles, "Model based Power and Energy Management System for PEM Fuel Cell/Li-Ion Battery driven Propulsion System," *Proc. Intl. Conf. on Power Electronics, Machines and Drives*, 2010, pp. 1 – 6.

[11] K-Y Huang, H-C Chin and Y-C Huang, "A Model Reference Adaptive Control Strategy for Interruptible Load Management," *IEEE Trans. on Power Systems*, vol. 19, no. 1, pp. 683 – 689, Feb. 2004.

[12] R. L. Welch and G. K. Venayagamoorthy, "Comparison of two Optimal Control Strategies for a Grid Independet Photovoltaic System," *Proc. Intl. Conf. on Industry Applications*, 2006, vol. 3, pp. 1120 – 1127.

[13] S. Ray, G. K. Venayagamoorthy, B. Chaudhuri and R. Majumder, "Comparison of Adaptive Critic based and Classical Wide-Area Controller for Power Systems," *IEEE Trans. on Systems, Man and Cybernetics – Part B: Cybernetics*, vol. 38, no.4, pp. 1002 – 1007, Aug. 2008.

[14] J-W Park, G. K. Venayagamoorthy and R. G. Harley, "Adaptive Critic Designs and their Implementations on Different Neural Network Architectures," *Proc. Intl. Joint Conf. on Neural Networks*, 2003, vol. 3, pp. 1879 – 1884.

[15] P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. dissertation, Committee on Applied Mathematics, Harvard Univ., Cambridge, MA, 1974.

[16] P. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*. Cambridge, MA: MIT Press, 1990, pp. 67 – 95.

[17] P. Werbos, "Backpropagation and neurocontrol: A review and prospectus," *Proc. Intl. Joint Conf. on Neural Networks*, 1989, vol. 1, pp. 209 – 216.

[18] D. V. Prokhorov and D. C.Wunsch, II, "Adaptive critic designs," *IEEE Trans. Neural Networks*, vol. 8, no. 5, pp. 997 – 1007, Sep. 1997.

[19] J. Si, A. G. Barto, W. B. Powell and D. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*. New York: Wiley, Jul. 2004