Adaptive critic based optimal neurocontrol for synchronous generator in power system using MLP/RBF neural networks

Jung-Wook Park

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Ronald G. Harley

Follow this and additional works at: http://scholarsmine.mst.edu/faculty_work

Part of the Electrical and Computer Engineering Commons

Recommended Citation
http://scholarsmine.mst.edu/faculty_work/1639

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. For more information, please contact weaverjr@mst.edu.
Adaptive Critic Based Optimal Neurocontrol for Synchronous Generator in Power System Using MLP/RBF Neural Networks

Jung-Wook Park and Ronald G. Harley
The School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250 USA
E-mail: jwpark@ece.gatech.edu, rharley@ece.gatech.edu

Ganesh K. Venayagamoorthy
Department of Electrical and Computer Engineering
University of Missouri-Rolla
MO 65409-0249, USA
E-mail: ganeshv@ece.umr.edu

Abstract — This paper presents a novel optimal neurocontroller that replaces the conventional controller (CONVC), which consists of the automatic voltage regulator (AVR) and turbine governor, to control a synchronous generator in a power system using a multilayer perceptron neural network (MLPN) and a radial basis function neural network (RBFN). The heuristic dynamic programming (HDP) based on the adaptive critic design (ACD) technique is used for the design of the neurocontroller. The performance of the MLPN based HDP neurocontroller (MHDPC) is compared with the RBFN based HDP neurocontroller (RHDPC) for small as well as large disturbances to a power system, and they are in turn compared with the CONVC. Simulation results are presented to show that the proposed neurocontrollers provide stable convergence with robustness, and the RHDPC outperforms the MHDPC and CONVC in terms of system damping and transient improvement.

Index Terms— Adaptive critic design (ACD), heuristic dynamic programming (HDP), multilayer perceptron network (MLPN), optimal neurocontroller, radial basis function network (RBFN), synchronous generator.

I. INTRODUCTION

A synchronous generator in a power system is a nonlinear, fast acting, multiple-input multiple-output (MIMO) device [1], [2]. Conventional linear controllers (CONVC) for the synchronous generator consist of the automatic voltage regulator (AVR) to maintain constant terminal voltage and the turbine governor to maintain constant speed and power at some set point. They are designed to control, in some optimal fashion, the generator around one particular operating point; and at any other point the generator’s damping performance is degraded. As a result, sufficient margins of safety are included in the generator maximum performance envelope in order to allow for degraded damping when transients occur. Due to a synchronous generator’s wide operating range, its complex dynamics [3], [4], its transient performance, its nonlinearities, and a changing system configuration, it cannot be accurately modelled as a linear device.

Artificial neural networks (ANNs) offer an alternative for the CONVC as nonlinear adaptive controllers. Researchers in the field of electrical power engineering have until now used two different types of neural networks, namely, a multilayer perceptron network (MLPN), or a radial basis function network (RBFN), both in single and multi-machine power system studies [3]-[7]. Proponents of each type of neural network have claimed advantages for their choice of ANN, without comparing the performance of the other type for the same study. The applications of ANNs in the power industry are expanding, and at this stage there is no authoritative fair comparison between the MLPN and the RBFN [8], [9].

The authors’ earlier work comparing performance of the above two ANNs’ for the indirect adaptive control of the synchronous generator showed that the RBFN based neurocontroller improves the system damping and transient performance more effectively and adaptively than the MLPN based neurocontroller [9]. Also, the different damping properties of the above two neurocontrollers and the stability issue during transients were analyzed and proven based on the Lyapunov direct method. However, one cannot avoid the possibility of instability during steady state at the various different operating conditions when using the indirect adaptive control based on the gradient descent algorithm. To overcome the issue of instability and provide strong robustness for the controller, the adaptive critic design (ACD) technique [10]-[16] for the optimal control has been recently developed where the ANNs are used to identify and control the process.

Without the highly extensive computational efforts and difficult mathematical analyses required by using the dynamic programming (DP) in classical optimal control theory [17]-[20], the ACD technique provides an effective method to construct an optimal and robust feedback controller by exploiting backpropagation for the calculation of all the derivatives of a target quantity [10], [21] in order to minimize/maximize the heuristic cost-to-go approximation.

In this paper, the background of adaptive critic designs with relation to optimal control theory, and a general description for the MLPN/RBFN, are presented. Based on the heuristic dynamic programming (HDP), which is a class of ACD family, the two optimal neurocontrollers using the MLPN and RBFN (called MHDPC and RHDPC, respectively) are designed. In addition, their performances for the real-time control of the synchronous generator connected to an infinite
bus are illustrated and compared with case studies by time-domain simulation.

II. BACKGROUND ON ADAPTIVE CRITIC DESIGNS AND DESCRIPTION OF MLPN/RBFN

How can the ANNs be applied to handle optimal control theory at the level of human intelligence? As one approach for solution of this problem, this section describes the framework behind the adaptive critic neural network based design for solving optimal control problems such as in the design of an optimal controller for the nonlinear synchronous generator in a power system network.

A. Optimal Control Problem

The continuous-time dynamic systems to be considered in finite state problem are as follows.

\[ \dot{x}(t) = f(x(t), u(t), t), \quad 0 \leq t \leq T \]  

(1)

where \( x(t) \in \mathbb{R}^n \) is the state vector at time \( t \), \( \dot{x}(t) \in \mathbb{R}^n \) is the vector of first order time derivatives of the states at time \( t \), \( u(t) \in U \subset \mathbb{R}^m \) is the control vector at time \( t \), and \( T \) is the terminal time. It is assumed that the system function \( f \) is continuously differentiable with respect to \( x \) and is continuous with respect to \( u \). The \textit{admissible} control functions, which are called \textit{control trajectories}, are the piecewise continuous functions \( \{u(t) : t \in [0, T]\} \) with \( u(t) \in U \) for all \( t \in [0, T] \). The task to be performed is to transfer the state from a known initial state \( x(0) \) to a specified final state \( x(T) \) in the target set of the state space. The task is implicitly specified by the performance criteria \( J(t, x) \), namely, optimal cost-to-go function at time \( t \) and state \( x \).

\[ J(t, x) = h(x(T)) + \int_0^T g(x(t), u(t))dt \]  

(2)

where \( h \) is the cost or penalty associated with the error in the terminal state at time \( T \), and \( g \) is the cost function associated with transient state errors and control effort. Then, the \textit{optimal control problem} can be considered as finding the \( u \in U \) to minimize the total cost function \( J \) in (2) subject to the dynamic system constraints in (1) and all initial and terminal boundary conditions that may be specified.

The Hamilton-Jacobi-Bellman (HJB) equation in (3), which is analogous with the DP algorithm, gives the solution to determine the optimal controls in off-line by deriving a partial differential equation satisfied by the optimal cost-to-go function \( J \) with assumed differentiability as the sufficient condition.

\[ 0 = \min_{u \in U} \left\{ g(x, u) + \nabla_t J(t, x) + \nabla_x J(t, x) f(x, u) \right\} \quad \text{for all} \quad t, x \]  

(3)

\[ J(T, x) = h(x), \quad \text{for boundary condition} \]

where \( \nabla_t \) denotes partial derivatives with respect to \( t \) and \( \nabla_x \) denotes an \( n \)-dimensional vector of partial derivatives with respect to \( x \). The HJB equation in (3) requires \( \nabla_x J \) to be known at all values of \( x \) and \( t \). However, the value of \( \nabla_x J \) is possible to be known at only one value of \( x \) for each \( t \) given in (4), and therefore \( \nabla_x J(t, x^*(t)) \) can be calculated more easily than the HJB equation. This is known as the \textit{adjoint equation} for the optimal state trajectory.

\[ u^*(t) = \arg \min_{u \in U} \left[ g(x^*(t), u) + \nabla_x J(t, x^*(t)) f(x^*(t), u) \right] \]  

(4)

where \( u^*(t) \) is the optimal control trajectory with corresponding state trajectory \( x^*(t) \) for all \( t \in [0, T] \). Then, the generalization of the calculus of variations known as the Pontryagin’s Minimum Principle is summarized as follows.

\[ p_0(t) = \nabla_x J(t, x^*(t)), \quad \dot{p}_0(t) = 0 \]  

(5)

\[ p(t) = \nabla_x J(t, x^*(t)), \quad p(T) = \nabla h(x^*(T)) \]  

(6)

\[ \dot{p}(t) = -\nabla_x f(x^*(t), u^*(t)) p(t) - \nabla_x g(x^*(t), u^*(t)) \]  

(7)

\[ u^*(t) = \arg \min_{u \in U} \left[ g(x^*(t), u) + p(t)^T f(x^*(t), u) \right] \]  

(8)

for all \( t \in [0, T] \).

B. Adaptive Critic Designs

For constant coefficient systems of which the operating time is very long, especially in real-time operation, it is often justifiable to assume that the terminal time is infinitely far in the future, which is called \textit{infinite horizon problem}. This approximation may cause little or no degradation in optimality because the optimal time-varying gains such as the costate equation in (7) approach constant values in a few time stages. Thus, the optimal gains are constant for most of the operating period.

The continuous-time cost function \( J \) in (2) can be reformulated as the total cost-to-go function of the infinite horizon problem in (9) for the discrete-time dynamic system.

\[ J_\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k g(x(k), u(k)) \]  

(9)

where \( k \) is a discrete time index at each step, \( J_\pi(x_0) \) denotes the cost associated with an initial state \( x_0 \), and a control policy \( \pi = \{u_0, u_1, \ldots\} \), and \( \gamma \) is the discount factor \( (0 < \gamma < 1) \). The Bellman equation using the DP in (10) is iteratively solved at each time step to find the optimal control \( u^* \) corresponding to the optimal cost-to-go function \( J^* \) in (11).

\[ J_{k+1}(x) = \min_{u \in U} \left[ g(x, u) + \gamma J_k(f(x, u)) \right], \quad k = 0, 1, \ldots \]  

(10)

\[ J_0(x) = 0 \quad \text{for all} \quad x \]

\[ J^*(x) = \lim_{k \to \infty} (T^k J_0)(x), \quad \text{for all} \quad x \in S \]  

(11)
where \((TJ)(x)\) is a DP mapping function defined in (12) on the
state space \(S\) for any function \(J: S \rightarrow \mathbb{R}\).

\[
(TJ)(x) = \min_{u \in C} \left[ g(x, u) + \gamma J(f(x, u)) \right]
\] (12)

However, the above optimal control theory cannot readily
be applied to deal with a large number of control variables of a
nonlinear dynamic system such as synchronous generators in a
multi-machine power system. Also, the classical DP algorithm
requires extensive computations and memory, known as the
so-called “curse of dimensionality”. To overcome this
problem, several alternative methods have been proposed
depending on manner in which the cost-to-go approximation is
selected, and one of those approaches is the neuro-dynamic
programming (NDP) using some form of “least-squares fit”
for the heuristic cost-to-go approximation [19].

Adaptive critic designs (ACD) technique can be classified
as one of the NDP families using function approximator such
as ANN architectures. In other words, this novel technique
provides an alternative approach to handle the optimal control
problem combining concepts of the reinforcement learning
and the approximate dynamic programming (ADP). The
illustration relating the optimal control theory to the ACD is
shown in Fig. 1. The ACD described in this paper uses three
different types of neural networks, namely the critic, model,
and action.

![Fig. 1. Optimal controller design for infinite horizon problem: Optimal control theory versus adaptive critic designs (ACD)](image)

In Fig.1, the utility function or cost function \(U^C\) to be
minimized is called “reinforcement” in the ACD. In applying
the ANNs to reinforcement learning, there are two major steps
to account for the link between present actions and future
consequences for the ACD technique [10]. The first step is to
build a “model” network for identifying the plant, and use
backpropagation to calculate the derivatives of future utility
with respect to present actions through the model network.
The second step is to adapt a “critic” network, a special
network that outputs an estimate of the total future value of
\(U^C\), which will arise from the present and past states and the
control information. From the viewpoint of optimal control
theory, the backpropagation is the same as the first-order
calculus of variations to calculate the costate equation in (7)
by taking the derivatives. Likewise in the adaptive critic, \(J^C(k)\)
can be derived using the ADP. In other words, the critic
network learns to approximate the heuristic cost-to-go function in (13).

\[
J^C(k) = \sum_{p=0}^{\infty} \gamma^p U^C(k+p)
\] (13)

where \(\gamma\) is the discount factor \((0 < \gamma < 1)\). After minimizing the
\(J^C\) in (13) by the critic network, the “action” network is
trained with the estimated output backpropagated from the
critic network to obtain the converged weight for the optimal
control \(u^*\). All the steps in both optimal controller design
methods in Fig. 1 are carried out in an off-line mode. The
design of the model, critic, and action networks are described
in Section III together with their mathematical analyses.

C. Multilayer Perceptron Network (MLPN)

In this paper, the MLPN consists of three-layers of neurons
(input, hidden and output layer as shown in Fig. 2) interconnected by the weight vectors, \(W\) and \(V\).

![Fig. 2. The MLPN structure](image)

The weights of the MLPN are adjusted/trained using the
gradient descent based backpropagation algorithm. The
activation function for neurons in the hidden layer is given by
the following sigmoidal function.

\[
f(x) = \frac{1}{1+\exp(-x)}
\] (14)

The output layer neurons are formed by the inner products
between the nonlinear regression vector from the hidden layer
and the output weight matrix, \(V\). Generally, the MLPN starts
with random initial values for its weights, and then computes a one-pass backpropagation algorithm at each time step \( k \), which consists of a forward pass propagating the input vector through the network layer by layer, and a backward pass to update the weights by the gradient descent rule. By trial and error, fourteen, ten, and thirteen neurons in the hidden layer for the model, action, and critic network, respectively, are optimally chosen for this study. These values depend on a trade-off between convergence speed and accuracy.

**D. Radial Basis Function Network (RBFN)**

Like the MLPN, the RBFN also consist of three-layers (Fig. 3). However, the input values are each assigned to a node in the input layer and passed directly to the hidden layer without weights. The hidden layer nodes are called RBF units, determined by a parameter vector called \( \text{center} \) and a scalar called \( \text{width} \). The gaussian density function is used as an activation function for the hidden neurons in Fig. 3.

The overall input-output mapping equation of the RBF is as follows.

\[
y_j = b_j + \sum_{j=1}^{h} \exp \left( - \frac{\|X - C_j\|^2}{\beta_j^2} \right)
\]

where \( X \) is the input vector, \( C_j \) is the \( j^{th} \) center of RBF unit in the hidden layer, \( b_j \) is the number of RBF units, \( b_j \) and \( v_{ji} \) are the bias term and the weight between the hidden and output layers, respectively, and \( y_j \) is the \( i^{th} \) output. Once the centers of RBF units are established, the width of the \( i^{th} \) center in the hidden layer is calculated by (16).

\[
\beta_i = \left( \frac{1}{n} \sum_{j=1}^{h} \sum_{k=1}^{n} \left( \|X_k - C_i\| \right)^2 \right)^{1/2}
\]

where \( e_{ki} \) and \( e_{kj} \) are the \( k^{th} \) value of the center of \( i^{th} \) and \( j^{th} \) RBF units. In (15) and (16), \( \| \cdot \| \) represents the euclidean norm.

There are four different ways for input-output mapping using the RBFN, depending on how the input data is fed to the network [22].

- Batch mode clustering of centers and batch mode gradient decent for linear weights.
- Batch mode clustering of centers and pattern mode gradient decent for linear weights.
- Pattern mode clustering of centers and pattern mode gradient decent for linear weights.
- Pattern mode clustering of centers and batch mode gradient decent for linear weights.

To avoid the extensive computational complexity during training, the batch mode \( k \)-means clustering algorithm for centers is initially calculated for the centers of the RBF unit. Thereafter, the pattern mode least-mean-square (LMS) algorithm is calculated to update the output linear weights [8], [9]. By trial and error, twelve neurons for the model network and six neurons for the action and critic networks in the hidden layer are optimally chosen for this study.

### III. HEURISTIC DYNAMIC PROGRAMMING NEUROCONTROLLER

The structure of the HDP configuration is shown in Fig. 4. The critic network is connected to the action network through the model network, and is therefore called a model-dependent critic designs. All three theses different ANNs are described in the following subsections.

**A. Plant Modeling**

The synchronous generator, turbine, exciter and transmission system connected to an infinite bus in Fig. 5 form the plant (dotted block in Fig. 5.) that has to be
controlled. Nonlinear equations are used to describe and simulate the dynamics of the plant in order to generate the data for the optimal neurocontrollers. On a physical plant, this data would be measured. The generator (G) with its damper windings is described by the seventh order d-q axis set of equations with the generator current, speed, and rotor angle as the state variables [1], [2]. In the plant, $P_t$ and $Q_t$ are the real and reactive power at the generator terminal, respectively, $Z_e$ is the transmission line impedance, $P_m$ is the mechanical input power to the generator, $V_{ref}$ is the infinite bus voltage, $\Delta \omega$ is the speed deviation, $\Delta V_t$ is the terminal voltage deviation, $V_t$ is the terminal voltage, $\Delta V_{ref}$ is the reference voltage deviation, $V_{ref}$ is the reference voltage, $\Delta P_m$ is the input power deviation, and $P_m$ is the turbine input power.

**B. Design of the Model Network**

Fig. 6 illustrates how the model network (identifier) is trained to identify the dynamics of the plant. The input vector, $U_m(k)$ consists of the turbine input power deviation ($\Delta P_m$) and exciter input voltage deviation ($\Delta V_{ref}$), that is, $U_m(k) = [\Delta P_m(k), \Delta V_{ref}(k)]$, and is fed into the plant with the vector, $\text{Ref}(k) = [P_o(k), V_o(k)]$. The input signals of $U_m(k)$ are generated as small pseudo-random binary signals (PRBSs) with a sampling period of 20 ms. The output vector of the plant, $Y_p(k)$ consists of the speed deviation ($\Delta \omega$) and terminal voltage deviation ($\Delta V_t$), that is, $Y_p(k) = [\Delta \omega(k), \Delta V_t(k)]$. The model network output, $\hat{Y}_m(k) = \hat{f}(X_m(k))$, where $X_m(k)$ is the input vector to the function $\hat{f}$ by the model network consisting of three time lags of system input and output, respectively.

The residual vector, $E_m(k)$ given in (18) is used for updating the model network’s weights $W_m(k)$ during training by the backpropagation algorithm.

$$E_m(k) = Y_p(k) - \hat{Y}_m(k)$$  \hspace{1cm} (18)

This training is carried out at several different operating conditions within the stability limit of the synchronous generator until satisfactory identification results are obtained. Then, the weights $W_m$ of the model network are fixed during the development of the critic and action networks.

**C. Design of the Critic Network**

The critic network in the HDP approximates the function $J^C$ itself in (13). The configuration for training the critic network is shown in Fig. 7. The Bellman equation in DP in (10) is implemented by the ADP using two critic networks. From (10), we get the following thing.

$$e_{dp} = [g(\mathbf{x}, \mathbf{u}) + \gamma J^C_k(\mathbf{f}(\mathbf{x}, \mathbf{u}))] - J^C_{k+1}(\mathbf{x})$$  \hspace{1cm} (19)

Note that the time indexing in (19) needs to be reversed for the problem discussed in this paper. In other words, the initial cost-to-function $J^C$ at time zero has a positive value $\alpha$ because the initial weights $W_C(0)$ of critic network are randomly chosen and the value of $J^C$ is kept minimizing as the time goes to an infinite. So, the following the error equation for the adaptation of critic network can be obtained.

$$e_c(k) = J^C_k(R(k)) - \gamma J^C_{k+1}(R(k+1)) - U^C_k(R(k))$$  \hspace{1cm} (20)

where $R(k+1)$ and $R(k)$ is a vector of observables of the plant, which are the output vectors from the model network in Fig. 6 at present and two consecutive past time stages for each vector. Then, the critic network’s weights $W_C$ are updated as follows.

$$W_C(k+1) = W_C(k) + \Delta W_C(k)$$  \hspace{1cm} (21)

$$\Delta W_C(k) = -\eta_c \cdot e_c(k) \cdot \frac{\partial e_c(k)}{\partial W_C(k)}$$  \hspace{1cm} (22)
where $\eta_C$ is the positive learning rate.

where

\[ Y_{d(k+1)} \]
\[ Y_{d(k)} \]
\[ Y_{d(k-1)} \]
\[ Y_{d(k-2)} \]
\[ Y_{d(k+1)} \]
\[ J_{C}^{(R(k+1))} \]
\[ U_{C}^{(R(k))} \]
\[ \sum_{i} e_{C} \]

Fig. 7. Critic adaptation in HDP: The same critic network is shown for two consecutive times, $k+1$ and $k$. The critic’s output $J_{C}^{(R(k+1))}$ at time $k+1$, is necessary for the approximate dynamic programming (ADP) to generate a target signal $\gamma J_{C}^{(R(k+1))} + U_{C}^{(R(k))}$ for training the critic network.

The training for critic network by the backpropagation algorithm is carried out until the value of $J^C$ is minimized as small as possible, which is almost zero. This adaptation process is considered as the value iteration in (12) to reach the optimal cost-to-go function $J^s$ in (11) by the ADP provided from two critic neural networks.

**D. Design of the Action Network**

The input of the action network in Fig. 4 is the output vector of the plant, $Y_P$ and its two time-delayed values, and the output vector of the action network is $A(k) = [\Delta P_m(k), \Delta V_{ref}(k)]$.

The objective of the action network shown in Fig. 4 is to find the optimal control $u^*$, as in (8), to minimize $J^s$ in the immediate future, thereby optimizing the overall cost expressed as a sum of all $U^C$ over the horizon of the problem in (13). This is achieved by training the action network with an error vector $e_A(k)$ in (23).

\[ e_A(k) = \frac{\partial J^s(k)}{\partial A(k)} \]  

(23)

The derivative of the cost function $J^s(k)$ with respect to $A(k)$ in (23) is obtained by backpropagating $\partial J^s/\partial J^C$ (recall that the HDP approximates the function $J^s$ itself) through the critic network and then through the pretrained model network to the action network. This gives $\partial J^s(k)/\partial Y_M(k)$ and $\partial J^C(k)/\partial A(k)$ in Fig. 4 for the weights $W_A(k)$ and the output vector $A(k)$ of the action network. The expression for the weights’ update in the action network is given in (24).

\[ \Delta W_A(k) = -\eta_A \cdot e_A(k) \cdot \frac{\partial e_A(k)}{\partial W_A(k)} \]  

(24)

where $\eta_A$ is the positive learning rate. The mathematical closed forms of $\partial J^C(k)/\partial Y_M(k)$ and $\partial J^C(k)/\partial A(k)$ are given in (25) and (26) for the MLPN and RBFN, respectively.

\[ \frac{\partial J^C}{\partial Y_M} = \frac{\partial J^C}{\partial t} \frac{\partial p_L}{\partial q_L} \frac{\partial p_L}{\partial q_L} \frac{\partial q_L}{\partial q_L} \frac{\partial q_L}{\partial q_L} \frac{\partial q_L}{\partial q_L} \frac{\partial Y_M}{\partial Y_M} \]  

\[ = \left[ \left\{ f_i(q_i)(1 - f_i(q_i))W_{C,i} \right\} \sum_{j=1}^{m} 1 \right] \text{MLPN} \]  

(25)

\[ \frac{\partial J^C}{\partial A} = \frac{\partial J^C}{\partial t} \frac{\partial p_L}{\partial q_L} \frac{\partial p_L}{\partial q_L} \frac{\partial q_L}{\partial q_L} \frac{\partial q_L}{\partial q_L} \frac{\partial q_L}{\partial q_L} \frac{\partial A}{\partial A} \]  

\[ = \left[ \left\{ \frac{2}{2} \right\} f_i(q_i)(1 - f_i(q_i))W_{M,i} \right\} \sum_{j=1}^{m} 1 \right] \text{MLPN} \]  

(26)

where:

- $t$ is target value.
- $m_i$ is the number of neurons in the hidden layer.
- $p$ is the output of the activation function for a neuron.
- $q$ is the regression vector as the activity of a neuron.
- $L$ and $l$ denote the output and hidden layer, respectively.
- The subscripts, $M$ and $C$ for RBFN denote the model and critic network, respectively.
- The function $f_i$ is the sigmoidal function in (14).
- The function $f_2$ is the gaussian density function defined in the right-hand side in (15) as an exponential form.

**E. Training Procedure for the HDP**

The general training procedure to adapt the HDP in off-line is explained in [10] and [12]. It consists of two training cycles: one for the critic network and the other for the action network. It is assumed that the model network is already trained and has its weights $W_M$ fixed. The critic network’s adaptation is initially carried out and alternated with action network’ until an acceptable performance is achieved. The training procedure for the adaptation of critic and action network is shown in Fig. 8. The weights $W_C$ are initialized with small random values. In the critic network’s training cycle, the incremental optimization is carried out by (20), (21) and (22). In the action network’s training cycle, the incremental learning is carried out by (23) and (24). It is important that the whole system consisting of the ACD and plant would remain stable while both the critic and action network undergo adaptation thus, the initial weights of the action network are those that ensure stabilizing control at an operating point.

Each training cycle (lengths of the corresponding training cycles for the critic and action network respectively) is continued until convergence of the network’s weights. The
convergence of the action network’s weights means that the training procedure has found weights that yield optimal control like the $u^*$ in (8) for the plant under consideration.

**Fig. 8. Training procedure for the HDP**

IV. SIMULATION RESULTS: MHDPC VERSUS RHDPC

After training the critic and action network off-line with the acceptable performance, the MHDPC and RHDPC with fixed weights are ready to control the plant for real-time operation. The performances of the optimal neurocontrollers, which are the MHDPC and RHDPC trained with deviation signals, are compared with CONVC for the improvement of system damping and transient stability. Two different types of disturbances, namely a ±5% step change in the reference voltage of exciter and a three phase short circuit at the infinite bus are carried out to evaluate the performance of the controllers. The CONVC has been tuned by the method explained in [5].

A. ±5% Step Changes in the Reference Voltage of Exciter to Represent a Small Impulse Type Disturbance

The plant is operating at a steady state condition ($P_e = 1$ [pu], $Q_e = 0.234$ [pu], and $Z_e = 0.02 + j0.4$ [pu]). At $t=1$ s, a 5% step increase in the reference voltage of the exciter is applied. At $t=12$ s, the 5% step increase is removed, and the system returns to its initial operating point.

The results in Figs. 9 and 10 show that the optimal neurocontrollers improve the transient system damping compared to the CONVC, and that the RHDPC outperforms the MHDPC, i.e. the RHDPC has the faster transient response than the MHDPC.

**Fig. 9. ±5% Step changes in reference voltage of exciter: Rotor angle**

**Fig. 10. ±5% Step changes in reference voltage of exciter: Terminal voltage**

B. Three Phase Short Circuit Test to Represent a Large Impulse Type Disturbance

A severe test is now carried out to evaluate the performances of the controllers under a large disturbance. At $t=0.3$ s, a temporary three phase short circuit is applied at the infinite bus for 100 ms from $t=0.3$ s to 0.4 s for the plant operating at the same steady state condition as previous test.

The results of this test, comparing the performance of the MHDPC, RHDPC, and CONVC, are shown in Figs. 11, 12 and 13. They show that the optimal neurocontrollers (MHDPC/RHDPC) damp out the low frequency oscillations for the rotor angle ($\delta$), the speed ($\omega$), and terminal voltage ($V_t$) more effectively than the CONVC.

**Fig. 11. Three phase short circuit test: Rotor angle**
V. CONCLUSIONS

This paper has shown the adaptive critic neural network design as an alternative to the classical optimal control method. The MLPN and RBFN based HDP optimal neurocontrollers (MHDPC/RHDP) have been designed for the control of a synchronous generator connected to an infinite bus. The results show that not only do the optimal neurocontrollers improve the system damping and dynamic transient stability more effectively than the CONVC for the large disturbance such as a three phase short circuit short, but also the RHDPC has a faster transient response than the MHDPC for a small disturbance like a ±5% step changes in the reference voltage of exciter.

Finally, the HDP based optimal neurocontrollers provide the robust feedback loop in the real-time operation without the necessity of continuously on-line training thus, overcoming the instability issue associated with the neural network based controllers.

REFERENCES