Optimal Control Synthesis of a Class of Nonlinear Systems Using Single Network Adaptive Critics

Radhakant Padhi [1], *Nishant Unnikrishnan* [2], *S. N. Balakrishnan* [3]
Department of Mechanical and Aerospace Engineering
University of Missouri – Rolla, MO 65409, USA

Abstract

Adaptive critic (AC) neural network solutions to optimal control designs using dynamic programming has reduced the need of complex computations and storage requirements that typical dynamic programming requires. In this paper, a "single network adaptive critic"(SNAC) is presented. This approach is applicable to a class of nonlinear systems where the optimal control (stationary) equation is explicitly solvable for control in terms of state and costate variables. The SNAC architecture offers three potential advantages; a simpler architecture, significant savings of computational load and reduction in approximation errors. In order to demonstrate these benefits a real-life Micro-Electro-Mechanical-system (MEMS) problem has been solved. This demonstrates that the SNAC technique is applicable for complex engineering systems. Both AC and SNAC approaches are compared in terms of some metrics.

1 Introduction

It is well-known that the dynamic programming formulation offers the most comprehensive solution approach to nonlinear optimal control in a state feedback form [Bryson]. However, solving the associated Hamilton-Jacobi-Bellman (HJB) equation demands a very large (rather infeasible) amount of computations and storage space. An innovative idea was proposed in [[a]Werbos] to get around this numerical complexity by using an 'Approximate Dynamic Programming (ADP)' formulation. The solution to the ADP formulation is obtained through a dual neural network approach called Adaptive Critic (AC). In one version of the AC approach, called the Dual Heuristic Programming (DHP), one network (called the action network) represents the mapping between the state and control variables while a second network (called the critic network) represents the mapping between the state and costate variables. Optimal solution is reached after the two networks iteratively train each other successfully. This DHP process, overcomes the computational complexity that had been the bottleneck of the dynamic programming approach. Proofs for both stability of the AC algorithm as well as the fact that the process will converge to the optimal control is found in [Liu] for linear systems

Among many successful uses of this method for nonlinear control design, we cite [Balakrishnan] in which the authors have solved an aircraft control problem using this technique and [Han] where the adaptive critic technique has been used for agile missile control. Padhi et al. [[a-c]Padhi] have extended the applicability of this technique to distributed parameter systems. There are various types of AC designs available in literature. An interested reader can refer to [[a]Prokhorov] for more details.

In this paper a significant improvement to the adaptive critic architecture is proposed. It is named Single Network Adaptive Critic (SNAC) because it uses only the critic network instead of the action-critic dual network set up in typical adaptive critic architecture. SNAC is applicable to a large class of problems for which the optimal control (stationary) equation is explicitly solvable for control in terms of state and costate variables. As an added benefit, the iterative training loops between the action and critic networks are no longer required. This leads to significant computational savings besides eliminating the approximation error due to action networks.

In literature there is an alternate approach to solving the optimal control problem using a neural network trained by a 'back propagation through time' (BPTT) approach [[b]Prokhorov] (an interested reader can find the details of BPTT in [[b]Werbos]). Even though the motivation behind the above mentioned work was to carry out a comparison study of computational complexity, no 'quantitative' comparison was made. In this paper, it is clearly shown through comparison studies with the typical dual-network based AC approach why SNAC is better. The SNAC approach presented in this paper is more *control designer* friendly since the neural networks embed more control theoretic knowledge.

2. Approximate Dynamic Programming

In this section, the principles of approximate (discrete) dynamic programming, on which both AC and SNAC approaches rely upon are described. An interested reader can find more details about the derivations in [Balakrishnan, [a]Werbos].

In discrete-time formulation, the aim is to find an admissible control U_k, which causes the system described by the *state equation*

$$X_{k+1} = F_k(X_k, U_k) \qquad (1)$$

to follow an admissible trajectory from an initial point X_1 to a final desired point X_N while minimizing a desired cost function J given by

[1] Postdoctoral Fellow, Email: padhi@umr.edu
[2] Ph.D. Student, Email: nu7v3@umr.edu
[3] Professor (Contact Person), Email: bala@umr.edu, Tel: 1(573)341-4675, Fax: 1(573)341-4607

$$J = \sum_{k=1}^{N-1} \Psi_k(X_k, U_k) \tag{2}$$

where the subscript k denotes the time step. X_k and U_k represent the $n \times 1$ state vector and $m \times 1$ control vector, respectively, at time step k. The functions F_k and Ψ_k are assumed to be differentiable with respect to both X_k and U_k. Moreover, Ψ_k is assumed to be convex (e.g. a quadratic function in X_k and U_k). One can notice that when $N \to \infty$, this leads to the *infinite time* problem. The aim is to find U_k as a function of X_k, so that the control can be implemented as a feedback.

Now, the steps in obtaining optimal control are described. First, the cost function in Eq(20) is rewritten for convenience to start from time step k as

$$J_k = \sum_{\tilde{k}=k}^{N-1} \Psi_{\tilde{k}}\left(X_{\tilde{k}}, U_{\tilde{k}}\right) \tag{3}$$

Then J_k can be split into

$$J_k = \Psi_k + J_{k+1} \tag{4}$$

where Ψ_k and $J_{k+1} = \sum_{\tilde{k}=k+1}^{N-1} \Psi_{\tilde{k}}$ represent the *utility function* at time step k and the *cost-to-go* from time step $k+1$ to N, respectively. The $n \times 1$ *costate* vector at time step k is defined as

$$\lambda_k = \frac{\partial J_k}{\partial X_k} \tag{5}$$

For *optimal control (stationary) equation*, the necessary condition for optimality is given by

$$\frac{\partial J_k}{\partial U_k} = 0 \tag{6}$$

However,

$$\begin{aligned} \frac{\partial J_k}{\partial U_k} &= \left(\frac{\partial \Psi_k}{\partial U_k}\right) + \left(\frac{\partial J_{k+1}}{\partial U_k}\right) \\ &= \left(\frac{\partial \Psi_k}{\partial U_k}\right) + \left(\frac{\partial X_{k+1}}{\partial U_k}\right)^T \left(\frac{\partial J_{k+1}}{\partial X_{k+1}}\right) = \left(\frac{\partial \Psi_k}{\partial U_k}\right) + \left(\frac{\partial X_{k+1}}{\partial U_k}\right)^T \lambda_{k+1} \end{aligned} \tag{7}$$

Thus combining Eqs.(6) and (7), the *optimal control equation* can be written as

$$\left(\frac{\partial \Psi_k}{\partial U_k}\right) + \left(\frac{\partial X_{k+1}}{\partial U_k}\right)^T \lambda_{k+1} = 0 \tag{8}$$

The *costate equation* is derived in the following way

$$\begin{aligned} \lambda_k = \frac{\partial J_k}{\partial X_k} &= \left(\frac{\partial \Psi_k}{\partial X_k}\right) + \left(\frac{\partial J_{k+1}}{\partial X_k}\right) \\ &= \left[\left(\frac{\partial \Psi_k}{\partial X_k}\right) + \left(\frac{\partial U_k}{\partial X_k}\right)^T \left(\frac{\partial \Psi_k}{\partial U_k}\right)\right] + \left[\left(\frac{\partial X_{k+1}}{\partial X_k}\right) + \left(\frac{\partial X_{k+1}}{\partial U_k}\right)\left(\frac{\partial U_k}{\partial X_k}\right)\right]^T \left(\frac{\partial J_{k+1}}{\partial X_{k+1}}\right) \\ &= \left[\left(\frac{\partial \Psi_k}{\partial X_k}\right) + \left(\frac{\partial X_{k+1}}{\partial X_k}\right)^T \lambda_{k+1}\right] + \left(\frac{\partial U_k}{\partial X_k}\right)^T \left[\left(\frac{\partial \Psi_k}{\partial U_k}\right) + \left(\frac{\partial X_{k+1}}{\partial U_k}\right)^T \lambda_{k+1}\right] \end{aligned} \tag{9}$$

Note that by using Eq.(8), *on the optimal path,* the costate equation Eq.(9) can be simplified to

$$\lambda_k = \left(\frac{\partial \Psi_k}{\partial X_k}\right) + \left(\frac{\partial X_{k+1}}{\partial X_k}\right)^T \lambda_{k+1} \tag{10}$$

Eqs.(1), (8) and (10) have to be solved simultaneously, along with appropriate boundary conditions for the synthesis of optimal control. Some of the broad classes of problems include *fixed* initial and final states, *fixed* initial state and *free* final state etc. For infinite time regulator class of problems, however, the boundary conditions usually take the form: X_0 is fixed and $\lambda_N \to 0$ as $N \to \infty$.

3. Adaptive Critics for Optimal Control Synthesis

In this section, the process of adaptive critics (AC) for optimal control synthesis is reviewed. In an AC framework, two neural networks (called as 'action' and 'critic' networks) are iteratively trained. After successful training, these networks capture the relationship between state and control and state and costate variables respectively. We review the steps in this section in fair detail.

3.1 State Generation for Neural Network Training

State generation is an important part of training procedures for both the AC and the newly-developed SNAC. For this purpose, define $S_i = \{X_k : X_k \in \textit{Domain of operation}\}$ where the action and critic networks have to be trained. This is chosen so that the elements of this set cover a large number of points of the state space in which the state trajectories are expected to lie. Obviously it is not a trivial task before designing the control. However, for the regulator class of problems, a stabilizing controller drives the states towards the origin. From this observation, a 'telescopic method' is arrived at as follows.

For $i = 1, 2, \ldots$ define the set S_i as $S_i = \{X_k : \|X_k\|_\infty \le c_i\}$ where, c_i is a positive constant. At the beginning, a small value of c_1 is fixed and both the networks are trained with the states generated in S_1. After convergence, c_2 is chosen such that $(c_2 > c_1)$. Then the networks are trained again for states within S_2 and so on. Values of $c_1 = 0.05$ and $c_i = c_1 + 0.05(i-1)$ for $i = 2, 3, \cdots$ are used in this study in Subsections 5.2 and 5.3. The network training is continued until $i = I$, where S_I covers the domain of interest.

3.2 Neural network training

The training procedure for the action network is as follows (Figure 1):

1. Generate set S_i (see Section 3.1). For each element X_k of S_i, follow the steps below:
   - **a.** Input X_k to the action network to obtain U_k
   - **b.** Get X_{k+1} from state Eq.(1) using X_k and U_k
   - **c.** Input X_{k+1} to the critic network to get λ_{k+1}

d. Using X_k and λ_{k+1}, calculate U_k^t (target U_k) from the optimal control Eq.(8)

2. Train the action network for all X_k in S_i, the output being corresponding U_k^t.

The steps for training the critic network are as follows (Figure 1):

1. Generate set S_i (see Section 3.1). For each element X_k of S_i, follow the steps below:

   a. Input X_k to the action network to obtain U_k

   b. Get X_{k+1} from the state Eq.(1) using X_k and U_k

   c. Input X_{k+1} to the critic network to get λ_{k+1}

   d. Using X_k and λ_{k+1}, calculate λ_k^t from the costate equation Eq.(10)

2. Train the critic network for all X_k in S_i, the output being corresponding λ_k^t.

3.3 Convergence Conditions

In order to check the individual convergence of the critic and action networks, a set of new states, S_i^c and target outputs are generated as described in Section 3.2. Let these target outputs be λ_k^t for the critic network and U_k^t for the action network. Let the outputs from the trained networks (using the same inputs from the set S_i^c) be λ_k^a for critic network and U_k^a for action network. Tolerance values tol_c and tol_a are used as convergence criteria for the critic and action networks respectively.

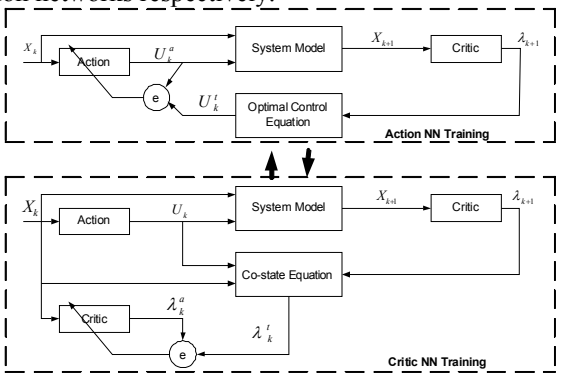

Figure 1: Adaptive Critic Network Training

The following quantities are defined as relative errors: $e_{c_k} \triangleq \left(\left\|\lambda_k^t - \lambda_k^a\right\| / \left\|\lambda_k^t\right\|\right)$ and $e_{a_k} \triangleq \left(\left\|U_k^t - U_k^a\right\| / \left\|U_k^t\right\|\right)$. Also define $e_c \triangleq \left\{e_{c_k}\right\}, k = 1, \cdots, |S|$ and $e_a \triangleq \left\{e_{a_k}\right\}, k = 1, \cdots, |S|$. When $\|e_c\| < tol_c$, the convergence criterion for the critic network training is met and when $\|e_a\| < tol_a$, the convergence criterions for the action network is met.

After successful training runs of the action and critic networks (i.e. after the convergence criterions are met), cycle error criterion are checked. For the training cycle $n > 1$, the error is defined as $err_{c_n} = \left\|e_{c_n} - e_{c_{n-1}}\right\| / \left\|e_{c_n}\right\|$ and $err_{a_n} = \left\|e_{a_n} - e_{a_{n-1}}\right\| / \left\|e_{a_n}\right\|$ for the critic and the action networks respectively. Also by defining $tol_{c_f} = \beta_c\, tol_c$, and $tol_{a_f} = \beta_a\, tol_a$ where $0 < \beta_c, \beta_a \le 1$, (for $n > 1$) if both $\left|err_{c_n} - err_{c_{n-1}}\right| < tol_{c_f}$ and $\left|err_{a_n} - err_{a_{n-1}}\right| < tol_{a_f}$, the cycle convergence criterion has been met. Further discussion on this adaptive critic method can be found in [[a]Werbos, Balakrishnan, [a]Padhi]. Note that this iterative training cycle will not be needed in the newly-developed SNAC technique (Section 4).

3.4 Initialization of networks: Pre-training

Note that during the process of action network training, the critic network is assumed to be optimal and vice versa. Consequently, there is a need to start with 'good' initial weights for the networks to lead to convergence. A process called "pre-training" is used for this purpose. This is carried out before starting the AC or SNAC training cycle. The neural networks are initially trained with the solution of the linearized problem using the standard linear quadratic regulator (LQR) theory [Bryson]. Intuitively, the idea is to start with a solution that is guaranteed to be 'close enough to' the optimal solution, at least in a small neighborhood of the origin. This approach is followed in the problems discussed in Section 5.

4. Single Network Adaptive Critic (SNAC) Synthesis

In this section, the newly developed single network adaptive critic (SNAC) technique is discussed in detail. As mentioned in Section 1, the SNAC technique retains all powerful features of the AC methodology while eliminating the action network completely. Note that in the SNAC design, the critic network captures the functional relationship between states X_k at stage k, and the costates λ_{k+1} at (k+1), whereas in the AC design the critic network captures the relationship between states X_k at stage k, and the costates λ_k at stage k. The SNAC method though is applicable only for problems where the optimal control equation Eq.(8) is explicitly solvable for control variable U_k in terms of the state variable X_k and costate variable λ_{k+1} (e.g. Systems that are affine in control fall into this class if the associated cost function is quadratic). This is not a hugely restrictive since many engineering problems such as aerospace, mechanical and chemical processes fall under this class.

4.1 Neural Network training

The steps in SNAC neural network training are as follows (Figure 2):

1. Generate S_i (see Subsection 3.1). For each element X_k of S_i, follow the steps below:

   a. Input X_k to the critic network to obtain $\lambda_{k+1} = \lambda_{k+1}^a$

   b. Calculate U_k, form the optimal control equation since X_k and λ_{k+1} are known.

   c. Get X_{k+1} from the state Eq.(1) using X_k and U_k

d. Input X_{k+1} to the critic network to get λ_{k+2}

e. Using X_{k+1} and λ_{k+2}, calculate λ^t_{k+1} from costate Eq.(10)

2. Train the critic network for all X_k in S_i; the output being corresponding λ^t_{k+1}.
3. Check for convergence of the critic network (Subsection 4.2). If convergence is achieved, revert to step 1 with $i = i+1$. Otherwise, repeat steps 1-2.
4. Continue steps 1-3 this process until $i = I$.

4.2 Convergence Condition

Convergence check in the SNAC scheme is carried out as in the AC case. First a set S^c_i of states is generated as explained in Subsection 3.1. Let these target output be λ^t_{k+1} and the outputs from the trained networks (using the same inputs from the set S^c_i) be λ^a_{k+1}. A tolerance value tol is used to test the convergence of the critic network. By defining the relative error $e_{c_k} \triangleq \left(\left\|\lambda^t_{k+1} - \lambda^a_{k+1}\right\| / \left\|\lambda^t_{k+1}\right\|\right)$ and $e_c \triangleq \left\{e_{c_k}\right\}, k = 1, \cdots, |S|$. , the training process is stopped when $\|e_c\| < tol$.

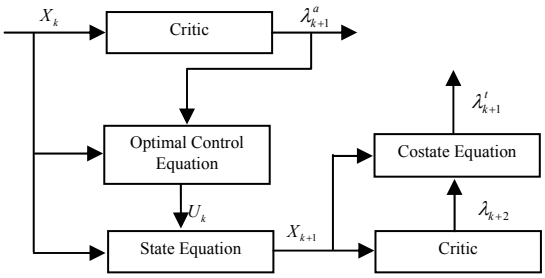

Figure 2: Single Network Adaptive Critic Scheme

4.3 Initialization of Networks: Pre-training

By using the standard discrete Linear Quadratic Regulator (LQR) theory, the Riccati matrix S_D and gain matrix K_D are obtained for use in pretraining[Bryson]. Note that S_D gives the relationship between X_k and λ_k, whereas the critic network in the SNAC has to be trained to capture the functional relationship between X_k and λ_{k+1}. This can be done by observing that

$$\lambda_{k+1} = S_d X_{k+1} = \tilde{S}_d X_k \tag{11}$$

where $\tilde{S}_d \triangleq S_d\left(A_d - B_d K_d\right)$. Eq.(11) is to pre-train the networks.

5. Numerical Results

In this section, numerical results from a representative problem is reported. The goals of this study are (i) to investigate the performance of the newly-developed SNAC controller in stabilizing a nonlinear system and (ii) to compare quantitatively the computations in using the SNAC and the AC. A personal computer having a Pentium III processor with 930 MHz speed and 320 MB of RAM was used to conduct the numerical experiments. The software used for training was MATLAB V. 5.2, Release 12. The Neural Network Toolbox V.3.0 in MATLAB was used with the Levenberg-Marquardt back-propagation scheme for training the networks.

5.1 Example 1: A Micro-Electro-Mechanical-System (MEMS) Actuator

5.1.1 Problem statement and optimality conditions

The problem considered in this study is a MEMS device, namely electrostatic actuator [Senturia]. In addition to demonstrating the computational advantage, this problem also proves that the SNAC technique is applicable for complex engineering systems of practical significance. The schematic diagram for this problem is as shown in Figure 3.

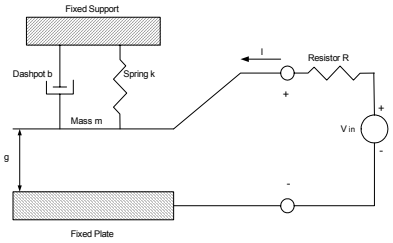

Figure 3: Electrostatic Actuator

There are two domains that are interlinked in the dynamics of the system. One is the electrical domain and the other is the mechanical domain. The governing equations are given by

$$\begin{aligned} &\dot{Q} - \frac{1}{R}\left(V_{in} - \frac{Qg}{\varepsilon A}\right) = 0 \\ &m\ddot{g} + b\dot{g} + k(g - g_0) + \frac{Q^2}{2\varepsilon A} = 0 \end{aligned} \tag{12}$$

where Q denotes the charge, g the gap between the plate and the base ($g_0 = 1\mu m$), and $\dot{g}$ represents the rate of change of the gap when the plate moves. V_{in} is the input voltage that is used to move the plate to the desired position. The mass m ($= 1mg$) represents the mechanical inertia of the moving plate, a dashpot b ($= 0.5mg/s$) captures the mechanical damping forces that arise from the viscosity of the air that gets squeezed when the plate moves, a spring k ($= 1\,mg/s^2$) represents the stiffness encountered when the plate actuator moves, a source resistor R ($= 0.001\,\Omega$) for the voltage source that drives the transducer. [Senturia]

Defining the state variable $Z = [z_1\ z_2\ z_3]^T = [Q\ g\ \dot{g}]^T$, Eq.(12) can be written as

$$\begin{aligned} \dot{z}_1 &= \frac{1}{R}\left(V_{in} - \frac{z_1 z_2}{\varepsilon A}\right) \\ \dot{z}_2 &= z_3 \\ \dot{z}_3 &= -\frac{1}{m}\left(\frac{z_1^2}{2\varepsilon A} + bz_3 + k(z_2 - g_0)\right) \end{aligned} \tag{13}$$

The function of the control input in this problem is to bring the plate to some desired position, i.e. the gap g has to be maintained at some desired value. We selected

the desired value of the gap as $0.5\,\mu m$. An optimal controller is designed to drive the plate to the desired value. At the equilibrium point, $z_2 = 0.5,\ \dot{Z} = 0$. Solving Eq.(13) for z_1, z_3 and V_{in} the values of the states at the equilibrium (operating) point are obtained as $Z_0 = [10 \quad 0.5 \quad 0]^T$ and the associated steady state controller value is given by $V_{in_0} = 0.05$. Next the deviated state is defined as $X = [x_1\ x_2\ x_3]^T \triangleq Z - Z_0$ and deviated control $u \triangleq V_{in} - V_{in0}$. In terms of these variables, the error dynamics of the system is

$$\begin{aligned}
\dot{x}_1 &= \frac{1}{R}\left(u - \frac{x_1}{2\varepsilon A} - \frac{x_2}{\sqrt{\varepsilon A}} - \frac{x_1 x_2}{\varepsilon A}\right) \\
\dot{x}_2 &= x_3 \\
\dot{x}_3 &= -\frac{1}{m}\left(\frac{x_1^2}{2\varepsilon A} + \frac{x_1}{\sqrt{\varepsilon A}} + kx_2 + bx_3 + \frac{1}{2} + \frac{k}{2} - \frac{g_0}{k}\right)
\end{aligned} \tag{14}$$

Now an optimal regulator problem can be formulated to drive $X \to 0$ with a cost function, J as

$$J = \frac{1}{2}\int_0^\infty \left(X^T Q_w X + R_w u^2\right) dt \tag{15}$$

where $Q_w \geq 0$ and $R_w > 0$ are weighting matrices for state and control respectively. As in Subection 5.1, the state equation and cost function were discretized as follows:

$$\begin{bmatrix} x_{1_{k+1}} \\ x_{2_{k+1}} \\ x_{3_{k+1}} \end{bmatrix} = X_k + \Delta t \begin{bmatrix} \frac{u_k}{R} - \frac{x_{1_k}}{2\varepsilon AR} - \frac{x_{2_k}}{R\sqrt{\varepsilon A}} - \frac{x_{1_k} x_{2_k}}{R\varepsilon A} \\ x_{3_k} \\ -\frac{x_{1_k}^2}{2\varepsilon Am} - \frac{x_{1_k}}{m\sqrt{\varepsilon A}} - \frac{kx_{2_k}}{m} - \frac{bx_{3_k}}{m} - \frac{1}{2m} - \frac{k}{2m} + \frac{g_0}{km} \end{bmatrix} \tag{16}$$

$$J = \sum_{k=1}^{N\to\infty} \frac{1}{2}\left(X_k^T Q_W X_k + R_W u_k^{\,2}\right)\Delta t \tag{17}$$

Next, using $\Psi_k = (X_k^T Q_W X_k + R_W u_k^2)\,\Delta t/2$ in Eqs.(8) and (10), the optimal control and costate equation can be obtained as follows:

$$u_k = -R_w^{-1}\frac{\lambda_{1_{k+1}}}{R} \tag{18}$$

$$\lambda_k = \Delta t\, Q_w X_k + \left[\frac{\partial F_k}{\partial X_k}\right]^T \lambda_{k+1} \tag{19}$$

5.1.2 Selection of design parameters

For this problem, values of $\Delta t = 0.01$, $Q_w = I_3$ and $R_w = 1$, $tol_a = tol_c = 0.05$ and $\beta_c = \beta_a = 0.01$ were chosen and the domain of the state $S_1 = \{X : |x_i| \leq 1, i = 1,2,3\}$. The 'telescopic method' described in subsection 3.1 was used for state generation. Each time 1000 points were randomly selected for training the networks. In SNAC synthesis, the tolerance value $tol = 0.05$ was used for convergence check. In the AC synthesis, three sub-networks each having a 3-6-1 structure were used as critics and a 3-6-1 network was used as the action network. In each network, hyperbolic tangent functions for the input and hidden layers and linear function for the output layer served as activation functions.

5.1.3 Analysis of results

Simulations were carried out using the same initial conditions for both AC and SNAC schemes. One set of initial conditions used was $[Q \ \ g \ \ \dot{g}]_{t=0}^T = [9.85 \ \ 1.5 \ \ -1]^T$. Figure 4 shows the trajectory of Q for both AC and SNAC techniques. Likewise Figures 5 and 6 show g and $\dot{g}$ trajectories respectively. Figure 7 shows the control trajectory obtained from using the two schemes.

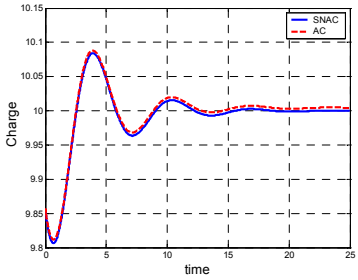

Figure 4: SNAC/AC State 1 trajectories

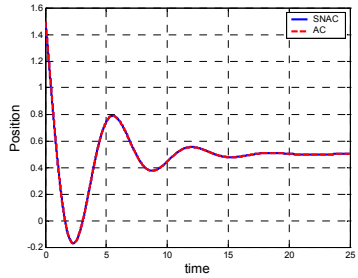

Figure 5: SNAC/AC State 2 trajectories

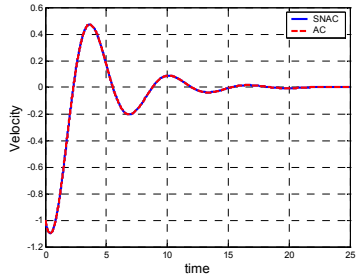

Figure 6: SNAC/AC State 3 trajectories

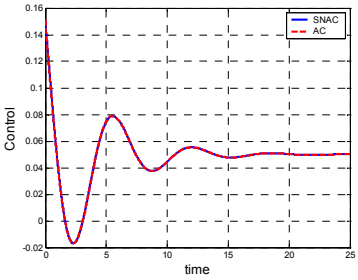

Figure 7: Associated control trajectories

Figures 4-6 indicate that both the AC and SNAC schemes performed well to drive the states to their respective values. It can be seen from Figure 5 that the

position of the actuator has been forced to the desired value of $0.5\ \mu m$. The velocity of the plate is driven to the steady state value of zero and the charge is driven to the steady state desired value. The control signal in both schemes drive toward a steady state value and are very close to each other (Figure 7).

Table 1 gives average training times and standard deviations for ten independent runs used in the AC scheme to find the optimal neural controller. (Total time: $T_{AC} = 890.33767$ seconds). Table 2 gives the average time taken to train the critic network in the SNAC methodology. (Total time: $T_{SNAC} = 531.40634$ seconds). It was observed that $T_{SNAC} = 0.59\, T_{AC}$, i.e. the training time for the SNAC method is $59\,\%$ of the training time for the AC technique. Small values of standard deviations once again indicate that the ten runs were very similar to each other from computational complexity considerations. The cost analysis for the two techniques from different initial conditions for $t_f = 25$ shown in Table 3 show that both schemes are close to each other as expected.

Table 1: Average AC training data

	Critic training and convergence check	Action training and convergence check	Cycle convergence check
Time (sec)	203.3852	535.0037	151.9488
Std. dev	0.412157	1.836064	0.234116

Table 2: Average SNAC training data

	Critic training	Critic convergence check
Time (sec)	477.9031	53.50321
Std. dev	2.204302	0.065727

Table 3: SNAC/AC Cost comparison for different initial conditions

Initial condition X(0)	COST (SNAC)	COST (AC)
$[-0.15\ 1\ -1]^T$	3.1964	3.1971
$[0.05\ -0.5\ 0.5]^T$	0.7989	0.7989
$[-0.05\ 0.5\ -0.5]^T$	0.7987	0.799
$[0.15\ -1\ 1]^T$	3.1937	3.194
$[0.1\ -0.6\ 0.3]^T$	0.7994	0.7997
$[0.1\ 0.3\ -0.6]^T$	0.729	0.7292

6. Conclusions

A new single network adaptive critic (SNAC) approach was presented. This approach is applicable to a wide class of nonlinear systems. This technique essentially retains all the powerful properties of a typical adaptive critic (AC) technique. However, in SNAC the action networks are no longer needed. As an important additional advantage, the associated iterative training loops are also eliminated. This leads to a great simplification of the architecture and results in substantial computational savings. Besides, it also eliminates the neural network approximation error due to the eliminated action networks. Tremendous computational savings with the SNAC have been demonstrated by using an interesting example. In addition, the MEMS problem also demonstrates that it is applicable for complex engineering systems of practical significance.

Acknowledgement: This research was supported by NSF grants 0201076 and 0324428.

References

1. **Balakrishnan, S. N.** and **Biega, V**., "Adaptive-Critic Based Neural Networks for Aircraft Optimal Control", *Journ. of Guid., Control and Dynamics, Vol. 19, No. 4, July-Aug. 1996, pp. 893-898.*
2. **Bryson, A. E**. and **Ho, Y. C**., "Applied Optimal Control", *Taylor and Francis, 1975.*
3. **Han, D**. and **Balakrishnan S. N.,** "Adaptive Critics Based Neural Networks for Agile Missile Control", *Journ. of Guid., Control and Dynamics, Vol.25 ,2002, pp.404-407.*
4. **Liu, X**. and **Balakrishnan, S. N.,** "Convergence Analysis of Adaptive Critic Based Optimal Control", *Proceedings of the American Control Conference, 2000, Chicago, USA, pp. 1929-1933*.
5. **[a]Padhi, R.,** "Optimal Control of Distributed Parameter Systems Using Adaptive Critic Neural Networks", 2001*, Ph.D. Dissertation*, University of Missouri Rolla.
6. **[b]Padhi, R., Balakrishnan, S. N.** and **Randolph T. W.**, "Adaptive-Critic Based Optimal Neuro Control Synthesis for Distributed Parameter Systems", *Automatica, Vol.37, 2001, pp.1223-1234.*
7. **[c]Padhi, R.** and **Balakrishnan, S. N.**, "Proper Orthogonal Decomposition Based Neurocontrol Synthesis of a Chemical Reactor Process Using Approximate Dynamic Programming", *Neural Networks, Vol.16 , 2003, pp.719-728.*
8. **[a]Prokhorov, D.V.,** and **Wunsch, D.C. II**, "Adaptive Critic Designs", *IEEE Transactions on Neural Networks, Vol.8, 1997, pp.*997-1007.
9. **[b]Prokhorov, D.V.**, "Optimal Controllers for Discretized Distributed Parameter Systems", *Proceedings of the American Control Conference, 2003, Denver, pp.*549-554.
10. **Senturia, S. D.,** "Microsystem Design", *Kluwer Academic Publishers, 2001.*
11. **[a]Werbos, P. J.**, "Approximate Dynamic Programming for Real-time Control and Neural Modeling". In White D.A., & Sofge D.A (Eds.), *Handbook of Intelligent Control, Multiscience Press,* 1992.
12. **[b]Werbos, P. J., "**Backpropagation Through Time: What it Does and How to Do It", *Proceedings of the IEEE, Vol.78, No.10, 1990, pp.1550-1560.*