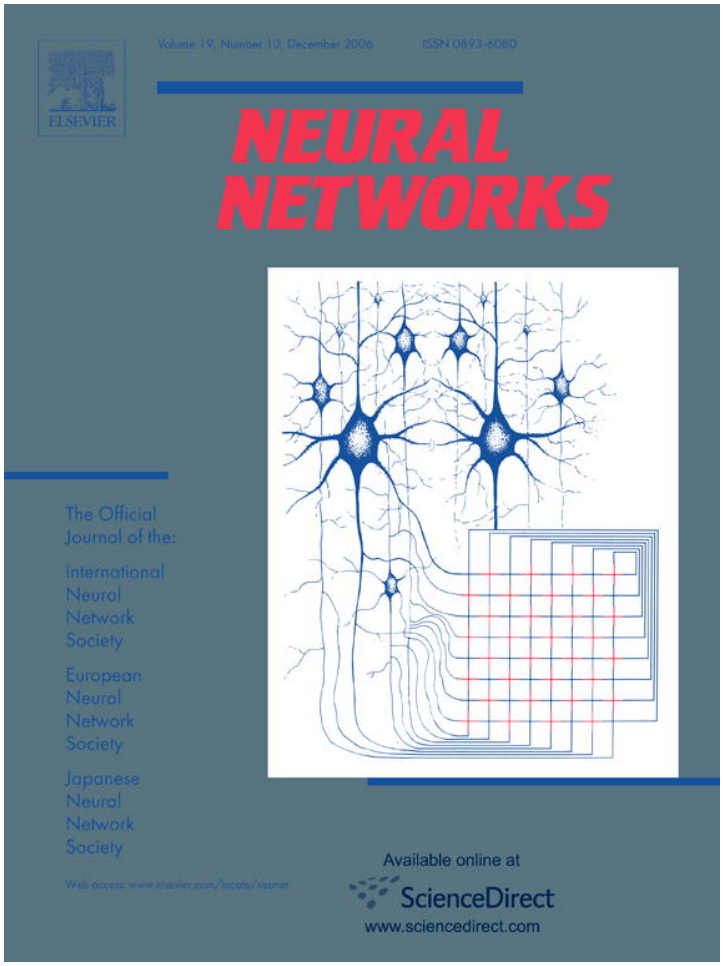

Available online at www.sciencedirect.com
ScienceDirect
Neural Networks 19 (2006) 1648–1660
Neural Networks
www.elsevier.com/locate/neunet

A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems

Radhakant Padhi[a,1], Nishant Unnikrishnan[b], Xiaohua Wang[b], S.N. Balakrishnan[b,*]

[a] *Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India*
[b] *Department of Mechanical and Aerospace Engineering, University of Missouri – Rolla, MO 65409, USA*

Received 27 January 2004; received in revised form 18 August 2006; accepted 18 August 2006

Abstract

Even though dynamic programming offers an optimal control solution in a state feedback form, the method is overwhelmed by computational and storage requirements. Approximate dynamic programming implemented with an Adaptive Critic (AC) neural network structure has evolved as a powerful alternative technique that obviates the need for excessive computations and storage requirements in solving optimal control problems. In this paper, an improvement to the AC architecture, called the "Single Network Adaptive Critic (SNAC)" is presented. This approach is applicable to a wide class of nonlinear systems where the optimal control (stationary) equation can be explicitly expressed in terms of the state and costate variables. The selection of this terminology is guided by the fact that it eliminates the use of one neural network (namely the action network) that is part of a typical dual network AC setup. As a consequence, the SNAC architecture offers three potential advantages: a simpler architecture, lesser computational load and elimination of the approximation error associated with the eliminated network. In order to demonstrate these benefits and the control synthesis technique using SNAC, two problems have been solved with the AC and SNAC approaches and their computational performances are compared. One of these problems is a real-life Micro-Electro-Mechanical-system (MEMS) problem, which demonstrates that the SNAC technique is applicable to complex engineering systems.

Keywords: Optimal control; Nonlinear control; Approximate dynamic programming; Adaptive critic; Single network adaptive critic; SNAC architecture

1. Introduction

Many difficult real-life control design problems can be formulated in the framework of optimal control theory. It is well-known that dynamic programming formulation offers the most comprehensive solution approach to nonlinear optimal control in a state feedback form (Bryson & Ho, 1975). A feedback control is desirable because of its beneficial properties like robustness with respect to noise and modeling uncertainties. However, solving the associated Hamilton–Jacobi–Bellman (HJB) equation demands a very large (rather infeasible) number of computations and storage space dedicated to this purpose. An innovative idea was proposed by Werbos (1992) 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 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. The optimal solution is reached after the two networks iteratively train each other successfully. This training process is carried out for a very large number of states within a 'domain of interest', within which the closed-loop state is supposed to lie during the duration of operation of the plant. Note that the domain of interest for which the networks need to be trained can be larger than the actual domain of operation, and hence it is usually not a difficult task to guess for such a domain while doing the off-line training.

* Corresponding address: Department of Mechanical and Aerospace Engineering, University of Missouri-Rolla, 1870 Miner Circle, Rolla, MO 65409, USA. Tel.: +1 573 341 4675; fax: +1 573 341 4607.
E-mail addresses: padhi@aero.iisc.ernet.in (R. Padhi), nu7v3@umr.edu (N. Unnikrishnan), wxw98@umr.edu (X. Wang), bala@umr.edu (S.N. Balakrishnan).
[1] Tel.: +91 80 2293 2756; fax: +91 80 2360 0134.

0893-6080/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2006.08.010

Nomenclature

X	State vector
U	Control vector
J	Cost function
Ψ	Utility function
λ	Costate vector
S_D	Ricatti matrix
K_D	Gain matrix
Δt	Discrete time interval
μ_T	Mean time
σ_T	Standard deviation
$\bar{y}_i$	Sample mean of ith data group
n	Data size
t_0	Test statistic
A	Area of electrostatic plate
ε	Permittivity
g_0	Initial air gap
m	Mass of electrostatic plate
b	Damping constant
R	Resistance
Q	Charge
g	Gap between plate and base
$\dot{g}$	Rate of change of gap
Q_W, R_W	Weighting matrices for state and control respectively

This DHP process, aided by the nonlinear function approximation capabilities of neural networks, overcomes the computational complexity that had been the bottleneck of the dynamic programming approach. Another important advantage of this method is that this solution can be implemented on-line. This is because for computing the control on-line, one needs only to use (not train) the neural networks. The time-consuming neural network training process can be carried out off-line. 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 and Balakrishnan (2000) for linear systems. A related but separate development towards stability and global convergence proofs is found in Murray, Cox, Lendaris, and Saeks (2002) for input-affine nonlinear systems with respect to their discussion on 'adaptive dynamic programming', the philosophy of which is closely related to approximate dynamic programming.

Among many successful uses of the AC method for nonlinear control design, we cite Balakrishnan and Biega (1996) in which the authors have solved an aircraft control problem using this technique and Han and Balakrishnan (2002) where the adaptive critic technique has been used for agile missile control. Another application of the adaptive critic technique can be seen in Venayagamoorthy, Harley, and Wunsch (2002) where the authors have used it for neurocontrol of a turbo generator. The authors in Ferrari and Stengel (2002) have implemented an adaptive critic global controller for a business jet. Recently, Padhi and Balakrishnan have extended the applicability of this technique to distributed parameter systems (Padhi, 2001; Padhi & Balakrishnan, 2003a, 2003b). In fact, there are various types of AC designs available in literature. An interested reader can refer to Prokhorov and Wunsch (1997) for more details about various types of AC designs. More details about the use of neural networks for control applications can be found in Hunt (1992), Miller, Sutton and Werbos (1990).

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 used in a 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. The computational savings are quantitatively demonstrated in this paper by solving two challenging nonlinear problems. Note that while applying both AC and SNAC techniques discussed in this paper, we assume that the plant equations and parameters are known.

In the control literature, there is an alternate approach for solving optimal control problems using a neural network trained using the back propagation through time (BPTT) approach (Prokhorov, 2003). An interested reader can find the details of BPTT in Werbos (1990). In Prokhorov (2003), the author has attempted to compare the computational complexity of the BPTT based approach with the approach presented in Padhi and Balakrishnan (2003a). Prokhorov was able to solve the problem successfully in a single network framework, as opposed to the dual-network framework presented in Padhi and Balakrishnan (2003a). However, even though the motivation was to carry out a comparison study for computational complexity, no 'quantitative' comparison has been made. In this paper, it is clearly shown through comparison studies why SNAC is better for a certain class of problems. Unlike the approach in Prokhorov (2003), the SNAC approach presented in this paper is more *control designer* friendly since the neural networks embed more control theoretic knowledge. For example, for control affine problems with a quadratic cost function it serves as the state dependent Ricatti operator, which has rich information and interpretation in a control theoretic sense.

Rest of the paper is organized as follows. In Section 2, the approximate dynamic programming technique is outlined. In Section 3, the AC (DHP) technique is outlined. In Section 4, the newly-developed SNAC technique is presented in detail. Numerical results for two different example problems (with increasing complexity) are presented in Section 5. Conclusions are drawn in Section 6.

2. Approximate dynamic programming

In this section, the principles of approximate (discrete) dynamic programming, on which both AC and SNAC

approaches rely are described. An interested reader can find more details about the derivations in Balakrishnan and Biega (1996), Werbos (1992).

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

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

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

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

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.

Note that a prime requirement to apply AC or SNAC is to formulate the problem in a discrete-time framework. If the discrete system dynamics and cost function are not available, the continuous expressions can be discretized before deriving the costate and optimal control equations. In this process, the control designer has the freedom of using any appropriate discretization scheme. For example, one can use Euler approximation for the state equation and Trapezoidal approximation for the cost function (Gupta, 1995).

Next, the steps in obtaining optimal control are described. First, the cost function in Eq. (2) is rewritten for convenience to start from time step k as

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

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

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

The necessary condition for optimality is given by

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

However,

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

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

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

The *costate equation* is derived in the following way

$$\lambda_k = \frac{\partial J_k}{\partial X_k} = \left(\frac{\partial \Psi_k}{\partial X_k}\right) + \left(\frac{\partial J_{k+1}}{\partial X_k}\right) = \left(\frac{\partial \Psi_k}{\partial X_k}\right) + \left(\frac{\partial X_{k+1}}{\partial X_k}\right)^{\mathrm{T}} \left(\frac{\partial J_{k+1}}{\partial X_{k+1}}\right). \tag{9}$$

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

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

Eqs. (1), (8) and (10) have to be solved simultaneously, along with appropriate boundary conditions, for the synthesis of optimal control. Note that the equations derived satisfy only necessary conditions. The sufficiency condition is very difficult to verify in the case of nonlinear discrete-time systems. In this work, we assume sufficiency conditions to hold good. We also assume that a unique optimal controller exists which will drive the system through the optimal trajectory.

Some of the broad classes of problems include *fixed* initial and final states, *fixed* initial state and *free* final state etc. For the infinite time regulator class of problems, however, the boundary conditions usually take the form: X_0 is fixed and $\lambda_N \to 0$ as $N \to \infty$. If the state equation and cost functions are such that one can obtain an explicit solution for the control variable in terms of state and costate variables from Eq. (8), then only SNAC is applicable. Note that many control affine nonlinear systems (of the form $X_{k+1} = f(X_k) + g(X_k)U_k$) with a quadratic cost function (of the form $J = \frac{1}{2}\sum_{k=1}^{\infty}(X_k^{\mathrm{T}} Q X_k + U_k^{\mathrm{T}} R U_k)$) fall under such a class. In this case the explicit expression for the control will be $U_k = -R^{-1}[g(X_k)]^{\mathrm{T}} \lambda_{k+1}$. Such problems have wide applicability in many real-life problems, including aircraft and robot control problems. Note that whenever the problem formulation allows the use of SNAC, it should always be preferred over AC because of its potential advantages. Since the control at step k is a function of costate at step $k+1$, closed form solutions do not exist except for linear systems and a limited number of nonlinear systems. Even though the dynamic programming approach offers such a framework, it is well-known that the approach runs into the "curse-of-dimensionality" issue, requiring huge (infeasible) amounts of computational and storage requirements. Avoiding this computational complexity is the main contribution of the AC and SNAC techniques. The SNAC technique leads to controller expressions explicitly in terms of the costate.

3. Adaptive critics for optimal control synthesis

In this section, the application of adaptive critics (AC) for optimal control synthesis is reviewed. In an AC framework, two neural networks (called the '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 $S_i = \{X_k : \|X_k\|_\infty \leq c_i\}$ 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 $i = 2, 3, \ldots$ are used in this study. 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, which captures the relationship between X_k and U_k, is as follows (Fig. 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 equation (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 equation (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, which captures the relationship between X_k and λ_k, are as follows (Fig. 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 equation (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 (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 the critic network and U_k^a for the action network. Tolerance values tol_c and tol_a are used as convergence criteria for the critic and action networks respectively. The following quantities are defined as relative errors: $e_{c_k} \triangleq \left(\|\lambda_k^t - \lambda_k^a\|/\|\lambda_k^t\|\right)$ and $e_{a_k} \triangleq \left(\|U_k^t - U_k^a\|/\|U_k^t\|\right)$. Also define $e_c \triangleq \{e_{c_k}\},\ k = 1, \ldots, |S|$ and $e_a \triangleq \{e_{a_k}\},\ k = 1, \ldots, |S|$. When $\|e_c\| < \text{tol}_c$, the convergence criterion for the critic network training is met and when $\|e_a\| < \text{tol}_a$, the convergence criterion for the action network is met.

After successful training runs of the action and critic networks (i.e. after the convergence criteria are met), cycle error criteria are checked. For the training cycle $n > 1$, the error is defined as $\text{err}_{c_n} = \|e_{c_n} - e_{c_{n-1}}\|/\|e_{c_n}\|$ and $\text{err}_{a_n} = \|e_{a_n} - e_{a_{n-1}}\|/\|e_{a_n}\|$ for the critic and the action networks respectively. Also by defining $\text{tol}_{c_f} = \beta_c\ \text{tol}_c$, and $\text{tol}_{a_f} = \beta_a\ \text{tol}_a$ where $0 < \beta_c, \beta_a \leq 1$, (for $n > 1$) if both $|\text{err}_{c_n} - \text{err}_{c_{n-1}}| < \text{tol}_{c_f}$ and $|\text{err}_{a_n} - \text{err}_{a_{n-1}}| < \text{tol}_{a_f}$, the cycle convergence criterion has been met. Further discussion on this adaptive critic method can be found in Balakrishnan and Biega (1996), Padhi (2001), Werbos (1992). 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

Initialization plays an important role in any optimization process. Before starting the training process outlined in Section 3.2, the networks should be appropriately initialized (we call this process 'pre-training'). The following approach works well for the quadratic regulator design problems. First, the system dynamics in Eq. (1) is linearized (Gopal, 1993) and the following standard representation of a linear system in the discrete time formulation is obtained

$$X_{k+1} = A_D X_k + B_D U_k. \tag{11a}$$

Using the standard *discrete linear quadratic regulator* (DLQR) optimal control theory (Bryson & Ho, 1975; Lewis, 1992), we can solve for the Ricatti matrix S_D and Gain matrix K_D. With the availability of S_D and K_D, we know from DLQR theory that the following relationships are satisfied

$$\lambda_k = S_D X_k \tag{11b}$$

$$U_k = -K_D X_k. \tag{11c}$$

The critic and action networks are initially trained with the static relationships given in Eqs. (11b) and (11c) respectively, before starting the actual AC training process outlined in Section 3.2. Intuitively, the idea is to start with the relationships that are 'close' to the optimal relationships (at least in a small neighborhood of the origin).

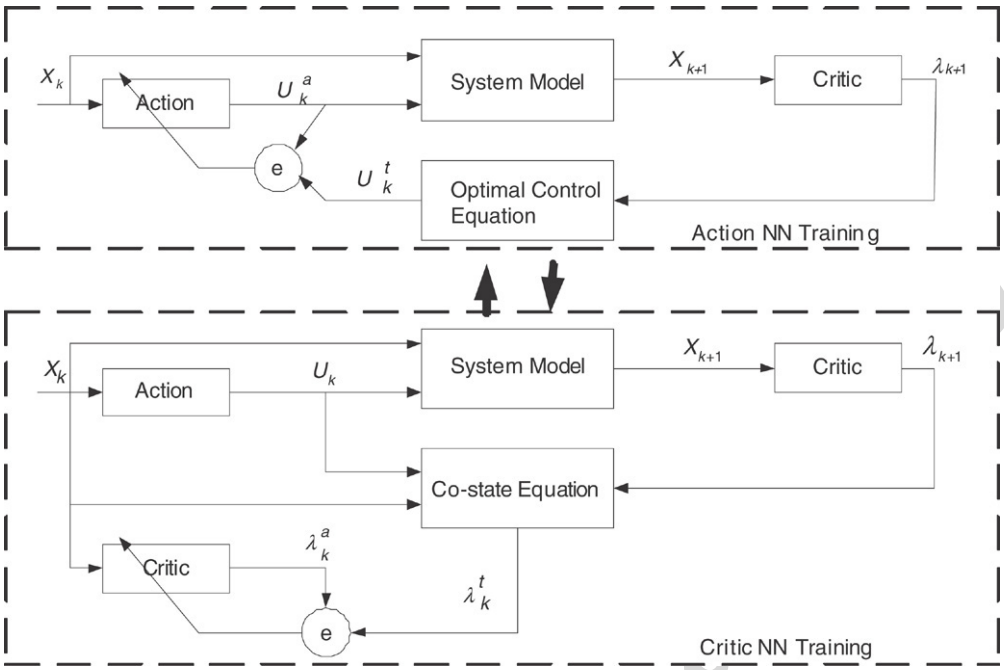

Fig. 1. Adaptive critic network training.

Note that there is no unique way of carrying out the pre-training process. Even though we have used the approach discussed above in the demonstrative problems in Section 5, the pre-training could have also been done with any other stabilizing control solution. For example, in Balakrishnan and Biega (1996) this pre-training process is carried out with just an arbitrarily chosen (non-optimal) stabilizing controller, before starting the AC process.

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 the 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 state X_k and costate λ_{k+1}, whereas in the AC design the critic network captures the relationship between state X_k and costate λ_k. However, the SNAC method is applicable only for problems where the optimal control equation (8) is explicitly expressible for control variable U_k in terms of the state variable X_k and costate variable λ_{k+1} (control affine systems with quadratic cost functions fall into this class), where such a restriction is not there for the AC technique. As mentioned earlier, Eqs. (1), (8) and (10) have to be solved simultaneously, along with appropriate boundary conditions, for the synthesis of optimal control. If the state equation and cost functions are such that one can obtain explicit solution for the control variable in terms of state and costate variables from Eq. (8), then only SNAC is applicable. Note that many control affine nonlinear systems (of the form $X_{k+1} = f(X_k) + g(X_k)U_k$) with a quadratic cost function (of the form $J = \frac{1}{2}\sum_{k=1}^{\infty}(X_k^{\mathrm{T}} Q X_k + U_k^{\mathrm{T}} R U_k)$) fall under such a class. In this case the explicit expression for the control will be $U_k = -R^{-1}[g(X_k)]^{\mathrm{T}}\lambda_{k+1}$. Due to the fact that this explicit relation exists between the costates and the controller, the SNAC structure involves mapping between X_k (state at instant k) and λ_{k+1} (costate at instant $k+1$) and computes the control value. On examining the optimal control expression in terms of the costate ($U_k = -R^{-1}[g(X_k)]^{\mathrm{T}}\lambda_{k+1}$) and comparing it with the optimal control law for a linear system of the form $x_{k+1} = Ax_k + Bu_k$, it can be seen from the structure in Fig. 2 that the critic network maps the relation $\lambda_{k+1} = \left(I + \mathrm{PBR}^{-1}B^{\mathrm{T}}\right)^{-1} PA\ X_k$ for linear systems where P is the solution of the algebraic Riccati equation. This work extends this for nonlinear systems using the nonlinear function approximation properties of neural networks.

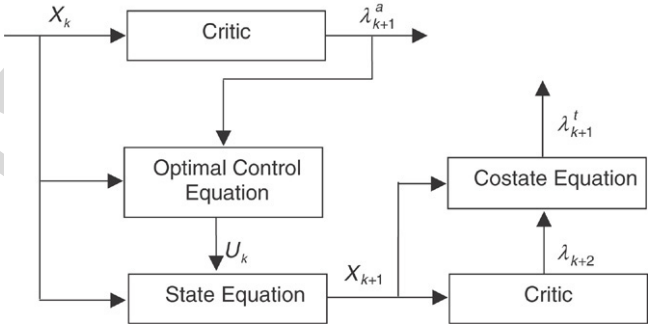

Fig. 2. Single network adaptive critic scheme.

4.1. Neural network training

In the SNAC approach, the steps for training the critic network, which captures the relationship between X_k and λ_{k+1}, are as follows (Fig. 2):

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

a. Input X_k to the critic network to obtain $\lambda_{k+1} = \lambda^a_{k+1}$
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 equation (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 equation (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 (Section 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 Section 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 $e_{c_k} \triangleq \left(\|\lambda^t_{k+1} - \lambda^a_{k+1}\|/\|\lambda^t_{k+1}\|\right)$ and $e_c \triangleq \left\{e_{c_k}\right\}$, $k = 1, \ldots, |S|$., the training process is stopped when $\|e_c\| < \text{tol}$.

4.3. Initialization of networks: Pre-training

For regulator problems, as in Section 3.4, the idea is to pre-train the neural network(s) with the solution for the linearized problem (using DLQR theory). However, note that S_D gives the relationship between X_k and λ_k (see Eq. (11b)), whereas the critic network in SNAC has to be trained to capture the functional relationship between X_k and λ_{k+1}. This can be done by observing that

$$\begin{aligned}\lambda_{k+1} &= S_D X_{k+1} \\ &= S_D\,(A_D X_k + B_D U_k) \\ &= S_D\,(A_D X_k - B_D K_D X_k) \\ &= \tilde{S}_D X_k \end{aligned} \tag{11d}$$

where $\tilde{S}_D \triangleq S_D\,(A_D - B_D K_D)$. The relationship in Eq. (11d) is used to pre-train the networks.

Once the iterative process of training the critic network is accomplished, the SNAC approach converges to the solution of the Riccati equation. This has been shown in the Appendix. As in the case of numerical solutions to the algebraic Riccati equation for linear systems, an initial stabilizing controller is required to ensure convergence of SNAC to the solution of the algebraic Riccati equation.

5. Numerical results

In this section, numerical results from two representative problems are reported. These are (i) a Van der Pol's oscillator and (ii) an electrostatic actuator. The goals of this study are (i) to investigate the performance of the newly-developed SNAC controller in stabilizing nonlinear systems 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 (Hagan, Demuth, & Beale, 1996) for training the networks.

5.1. Example 1: Van der Pol's oscillator

5.1.1. Problem description and optimality conditions

Motivation for selecting a Van der Pol oscillator was that it is a nonlinear benchmark problem (Yesildirek, 1994). The homogeneous system for this problem has an unstable equilibrium point at the origin ($x_1 = x_2 = 0$) and the system has a stable limit cycle as well. These properties make it a non-trivial regulator problem in the sense that without applying an appropriate control, the states starting from any non-zero initial condition will never go to zero (they will rather develop towards the limit cycle).

The system dynamics of a Van der Pol's oscillator is given by

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \alpha(1 - x_1^2)x_2 - x_1 + (1 + x_1^2 + x_2^2)u. \end{aligned} \tag{12}$$

Our goal in this problem was to drive $X \triangleq [x_1, x_1]^{\mathrm{T}} \to 0$ as $t \to \infty$. A quadratic cost function is formulated as

$$J = \frac{1}{2}\int_0^\infty \left(X^{\mathrm{T}} Q_W\, X + R_W\, u^2\right)\, \mathrm{d}t \tag{13}$$

where $Q_W = \mathrm{diag}(q_1, q_2)$ and $R_W = 1$. Discretization of Eqs. (12) and (13) using Euler and trapezoidal methods [Gupta] leads to

$$\begin{bmatrix} x_{1_{k+1}} \\ x_{2_{k+1}} \end{bmatrix} = \begin{bmatrix} x_{1_k} \\ x_{2_k} \end{bmatrix} + \Delta t \times \begin{bmatrix} x_{2_k} \\ \alpha(1-x_{1_k}^2)x_{2_k} - x_{1_k} + (1 + x_{1_k}^2 + x_{2_k}^2)u_k \end{bmatrix} \tag{14}$$

$$J = \sum_{k=1}^{N\to\infty} \frac{1}{2}\left(X_k^{\mathrm{T}} Q_W X_k + R_W u_k^2\right)\Delta t. \tag{15}$$

Note that the discretized cost function is slightly different from would have been obtained by using the trapezoidal method as such. We ignore the slight discrepancy at $k = 1, (N-1)$ for the sake of simplicity (the case $k = N-1$ does not matter here since $N \to \infty$). By substituting $\Psi_k = (X_k^{\mathrm{T}} Q_W X_k + R_W u_k^2)\Delta t/2$ in Eqs. (8) and (10), yields the following optimal control and costate equations.

$$u_k = -r^{-1}(1 + x_{1_k}^2 + x_{2_k}^2)\lambda_{2_{k+1}} \tag{16}$$

$$\begin{bmatrix} \lambda_{1_k} \\ \lambda_{2_k} \end{bmatrix} = \Delta t\; Q_W\; X_k + \begin{bmatrix} 1 & \Delta t[2x_{1_k}u_k - 1 - 2\alpha\; x_{1_k}x_{2_k}] \\ \Delta t & (1 + \Delta t)[2x_{2_k}u_k + \alpha(1 - x_{1_k}^2)] \end{bmatrix}\begin{bmatrix} \lambda_{1_{k+1}} \\ \lambda_{2_{k+1}} \end{bmatrix}. \tag{17}$$

5.1.2. Selection of design parameters

For this problem, we chose $\Delta t = 0.01$, $Q_W = \text{diag}(1, 2)$ and $R_W = 1, \text{tol}_a = \text{tol}_c = 0.05$ and $\beta_c = \beta_a = 0.2$. The domain of interest (for neural network training) was chosen to be $S_I = \{X : |x_i| \leq 1,\ i = 1, 2\}$. The 'telescopic method' described in Section 3.1 was used for state generation. Each time 1000 points were randomly selected for training the networks.

In the AC synthesis, the critic network was selected such that it consists of two sub-netwroks, each having a 2-6-1 structure (i.e. two neurons in the input layer, six neurons in the hidden layer and one neuron in the output layer). Similarly in the SNAC synthesis, the critic network was also assumed to have two sub-networks, each having a 2-6-1 structure. For the activation functions, the hyperbolic tangent function was selected for the input and hidden layers and linear function was chosen for the output layer (in both critic and action networks).

The reason for choosing two 2-6-1 sub-networks as the structure for the critic network (instead of a 2-6-2 network) is as follows. In optimal control theory, even though the costate (critic) variables are essential, these variables have no physical meaning. Prior to arriving at the optimal solution, the order of magnitudes the costates can take is unknown. Because of this, one element of the costate vector can be of very low order whereas the other one can be very high order. In such a case, even though it is theoretically sound to assume a single 2-6-2 network, there can be numerical problems in training. It may also lead to non-optimal values for some components of the costate vector. This may happen in spite of the convergence check criterion being met, because the error in the high magnitude component may suppress the error in the low magnitude component in the back-propagation training process. Such problems will not arise if we assume two 2-6-1 networks instead, since in this case the weights of the individual networks can see only the error for a particular component of the costate vector. For the action network in the AC synthesis, a 2-6-1 structure was chosen. Note that the control vector for this problem has only one component, and hence, there is no need for a sub-network structure.

It is well-known in the neural network literature that a two layer neural network with sufficient number of neurons in the hidden layer can approximate any continuous function with arbitrarily small error bound (Barto, Sutton, & Anderson, 1983). However, to the best of our knowledge, the number of neurons required for any particular application is still an open problem. Hence, selecting the structure of a neural network is more of an art than science. In our application problems, we selected six neurons for the hidden layer and this was satisfactory in the sense that we were able to meet the convergence tolerance values that we chose, which led to satisfactory simulation results.

5.1.3. Analysis of results

After synthesizing the neural networks as discussed in Section 4, we carried out simulation studies to validate the methods proposed. Arbitrary initial conditions for the states from the same domain of initial conditions as used for training were chosen. Using the synthesized neural networks for computing control, the system was simulated for 10 s (large enough for our purpose). Simulations showed that both the states were successfully driven to zero (the control goes to zero as well) with time. Figs. 3 and 4 are plots of the system state (i.e. position and velocity) histories. It can be seen that both the AC and SNAC techniques perform well in regulating the states (i.e. driving them to zero). Fig. 5 is the plot of the corresponding control history, which as expected, goes to zero as well. Note that the trajectories coming out of the AC and the SNAC techniques are pretty much close to each other. This shows that the newly developed SNAC approach is as good as the AC approach in synthesizing the optimal controller.

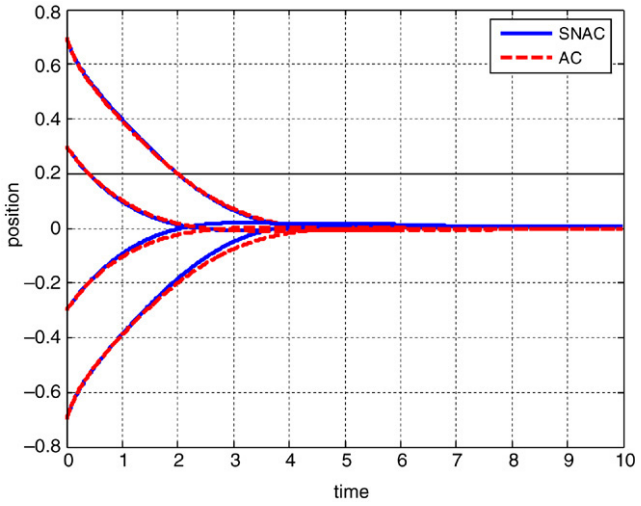

Fig. 3. Position trajectories (Van der Pol's oscillator).

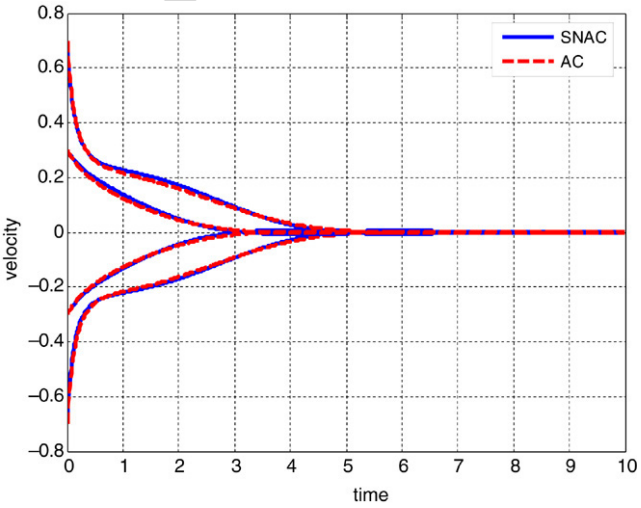

Fig. 4. Velocity trajectories (Van der Pol's oscillator).

As pointed out earlier, one of the main advantages of the SNAC approach over the AC is that it leads to substantial computational savings. To demonstrate this quantitatively, both techniques were statistically analyzed based on ten independent runs. Fig. 6 shows the time taken by the AC and SNAC methods to complete the process. This plot clearly indicates that it takes significantly less time to train the network using the SNAC

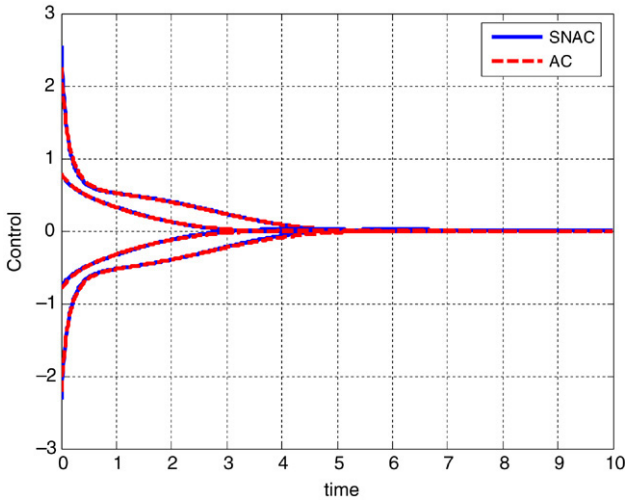

Fig. 5. Corresponding control trajectories (Van der Pol's oscillator).

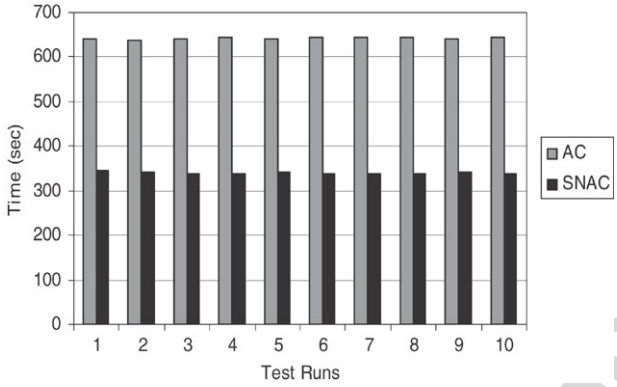

Fig. 6. Training time comparison [AC/SNAC] (Van der Pol's oscillator).

technique as compared to the AC technique. It was observed that $\mu_{T_{\text{SNAC}}} = 0.52\ \mu_{T_{\text{AC}}}$, where $\mu_{T_{\text{SNAC}}} = 339.8445$ s and $\mu_{T_{\text{AC}}} = 642.0185$ s are the mean times taken to train (including checking for convergence) using the SNAC and AC schemes respectively. It was also observed that $\sigma_{T_{\text{SNAC}}} = 1.676839$ s and $\sigma_{T_{\text{AC}}} = 1.887459$ s, where $\sigma_{T_{\text{SNAC}}}$ and $\sigma_{T_{\text{AC}}}$ are the standard deviations for the SNAC and AC schemes respectively. The small standard deviation values indicate that there is not much variation in total time taken for each test run, and hence both the techniques are fairly consistent.

Next, we viewed the data (run time) as if they were a random sample from a normal distribution. In that case the appropriate test statistic for comparing whether the two means are significantly different would be a two-sample t-Test (Montgomery, 2001). The statistic used to test this is

$$t_0 = \frac{\bar{y}_1 - \bar{y}_2}{S_p\sqrt{\frac{1}{n} + \frac{1}{n}}} \tag{18}$$

where $\bar{y}_i$ is the sample mean of the ith group, n is the sample size, S_p^2 is an estimate of the common variance. Details on how to compute S_p^2 are given in Montgomery (2001). To determine whether to reject the hypothesis that the two means of training time data are the same, we compare t_0 to the t distribution with

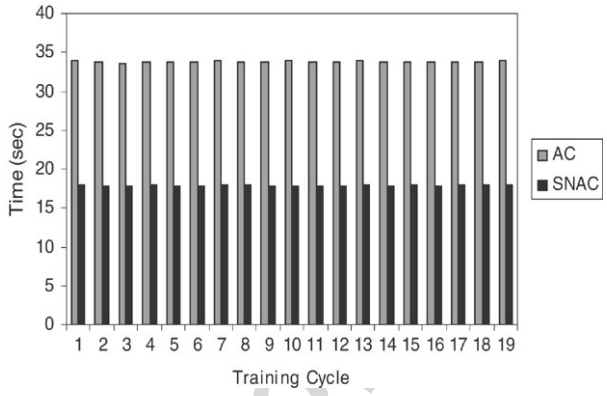

Fig. 7. Average training times for each step (Van der Pol's oscillator).

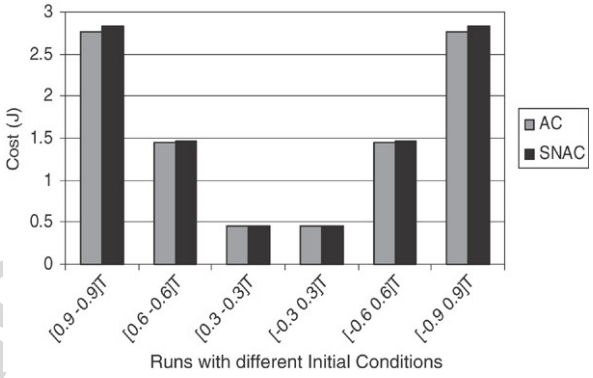

Fig. 8. Cost comparison for different initial conditions (Van der Pol's oscillator).

$2n - 2$ degrees of freedom. If $|t_0| > t_{\alpha/2,2n-2}$ (where α is the significance level of the test) we would conclude that the two means are different. It was seen that $t_0 = 378.4785 > t_{0.025,18} = 2.101$. Hence, we conclude that the two means (training time using SNAC and training time using AC) are significantly different (which we have already observed from the fact that $\mu_{T_{\text{SNAC}}} = 0.52\ \mu_{T_{\text{AC}}}$).

Fig. 7 compares the average times taken by AC and SNAC for each step in the telescopic training process discussed earlier. The average was taken over ten independent runs. From the figure, it is evident that the means of the two sets of data are significantly different. The cost comparison based on Eq. (15) for six different sets of initial conditions using the AC and SNAC methodologies for a simulation of $t_f = 10$ has been given in Fig. 8. It can be seen that the costs for the AC and SNAC schemes are very close to each other in each case.

Next, a paired comparison design (Montgomery, 2001) test was conducted to determine whether the two sets of costs are statistically different. Blocking is a design technique used to improve precision with which comparisons among the factors of interest are made. A block is a set of relatively homogeneous experimental conditions. In our case, the initial condition of simulation is the block. The test statistic to test whether the two sets of data are statistically different is

$$t_0 = \frac{\bar{d}}{S_d\sqrt{n}} \tag{19}$$

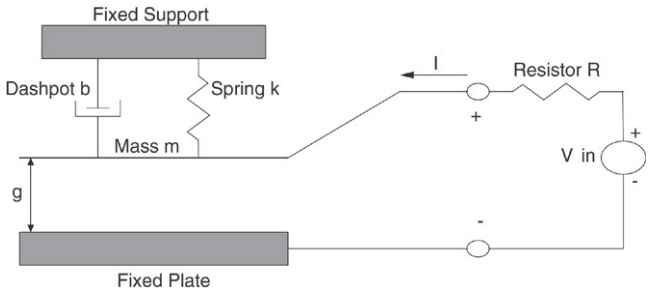

Fig. 9. Electrostatic actuator.

where $\bar{d}$ is the sample mean of the differences, S_d is the sample standard deviation of the differences and n is the number of samples. An interested reader can refer to Montgomery (2001) for details. The computed value of the paired t-test statistic is $t_0 = -0.2857$. Since $|t_0| = 0.2857 < t_{0.025,5} = 2.571$, we conclude that the two sets of data are not significantly different. This means that the costs generated by both SNAC and AC simulations are statistically similar, which confirms our claim once more that the SNAC technique finds out the same optimal controller as the AC technique.

5.2. Example 2: A micro-electro-mechanical-system (MEMS) actuator

5.2.1. Problem statement and optimality conditions

The next problem considered in this study is a MEMS device, namely electrostatic actuator (Senturia, 2001). 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 Fig. 9.

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

$$\begin{aligned} &\dot{Q} - \frac{1}{R}\left(V_{\text{in}} - \frac{Qg}{\varepsilon A}\right) = 0 \\ &m\ddot{g} + b\dot{g} + k(g - g_0) + \frac{Q^2}{2\varepsilon A} = 0 \end{aligned} \tag{20}$$

where Q denotes the charge, g the gap between the plate and the base, 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 represents the mechanical inertia of the moving plate, a dashpot b captures the mechanical damping forces that arise from the viscosity of the air that gets squeezed when the plate moves, a spring k represents the stiffness encountered when the plate actuator moves, a source resistor R for the voltage source that drives the transducer. The various parameters used in Eq. (20), along with their associated values are given in Table 1.

Table 1
Parameters used in modeling the electrostatic actuator (Senturia, 2001)

Parameter	Symbol	Value	Units
Area	A	100	μm^2
Permittivity	ε	1	$C^2/N\,\mu m^2$
Initial gap	g_0	1	μm
Mass	m	1	mg
Damping constant	b	0.5	mg/s
Spring constant	k	1	mg/s^2
Resistance	R	0.001	Ω

Defining the state variable $Z = [z_1\ z_2\ z_3]^{\text{T}} = [Q\ g\ \dot{g}]^{\text{T}}$, Eq. (20) can be written as

$$\begin{aligned} \dot{z}_1 &= \frac{1}{R}\left(V_{\text{in}} - \frac{z_1 z_2}{\varepsilon A}\right) \\ \dot{z}_2 &= z_3 \\ \dot{z}_3 &= -\frac{1}{m}\left(\frac{z_1^2}{2\varepsilon A} + bz_3 + k(z_2 - g_0)\right). \end{aligned} \tag{21}$$

The function of the control input in this problem is to bring the plate to a 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 μm. An optimal controller is designed to drive the plate to the desired value. At the equilibrium point, $z_2 = 0.5$, $\dot{Z} = 0$. Using this information in Eq. (21) leads to

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{R}\left(V_{\text{in}} - \frac{z_1}{2\varepsilon A}\right) \\ z_3 \\ \frac{1}{m}\left(-\frac{z_1^2}{2\varepsilon A} - k\,(0.5 - g_0) - bz_3\right) \end{bmatrix}. \tag{22}$$

Solving Eq. (22) for z_1, z_3 and V_{in} the values of the states at the equilibrium (operating) point are obtained as $Z_0 = \begin{bmatrix} 10 & 0.5 & 0 \end{bmatrix}^{\text{T}}$ and the associated steady state controller value is given by $V_{\text{in}_0} = 0.05$. Next the deviated state is defined as $X = [x_1\ x_2\ x_3]^{\text{T}} \triangleq Z - Z_0$ and deviated control $u \triangleq \left(V_{\text{in}} - V_{\text{in}_0}\right)$. In terms of these variables, the error dynamics of the system is

$$\begin{aligned} \dot{x}_1 &= \frac{1}{R}\left(u - \frac{x_1}{2\varepsilon A} - \frac{x_2}{\sqrt{\varepsilon A}} - \frac{x_1 x_2}{\varepsilon A}\right) \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -\frac{1}{m}\left(\frac{x_1^2}{2\varepsilon A} + \frac{x_1}{\sqrt{\varepsilon A}} + kx_2 + bx_3 + \frac{1}{2} + \frac{k}{2} - \frac{g_0}{k}\right). \end{aligned} \tag{23}$$

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

$$J = \frac{1}{2}\int_0^\infty (X^{\text{T}} Q_w\, X + R_w u^2)\, \text{d}t \tag{24}$$

where $Q_w \geq 0$ and $R_w > 0$ are weighting matrices for state and control respectively. Next, using the Euler and trapezoidal techniques (Gupta, 1995) the state equation and cost function

$$\begin{bmatrix} x_{1_{k+1}} \\ x_{2_{k+1}} \\ x_{3_{k+1}} \end{bmatrix} = X_k + \Delta t \begin{bmatrix} \dfrac{u_k}{R} - \dfrac{x_{1_k}}{2\varepsilon A R} - \dfrac{x_{2_k}}{R\sqrt{\varepsilon A}} - \dfrac{x_{1_k} x_{2_k}}{R\varepsilon A} \\ x_{3_k} \\ -\dfrac{x_{1_k}^2}{2\varepsilon A m} - \dfrac{x_{1_k}}{m\sqrt{\varepsilon A}} - \dfrac{k x_{2_k}}{m} - \dfrac{b x_{3_k}}{m} - \dfrac{1}{2m} - \dfrac{k}{2m} + \dfrac{g_0}{km} \end{bmatrix}$$

Box I.

were discretized as follows: (see Box I)

$$J = \sum_{k=1}^{N\to\infty} \frac{1}{2}(X_k^{\mathrm{T}} Q_W X_k + R_W u_k^2)\Delta t. \tag{25}$$

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

$$u_k = -R_w^{-1} \frac{\lambda_{1_{k+1}}}{R} \tag{26}$$

$$\lambda_k = \Delta t \; Q_w \; X_k + \left[\frac{\partial F_k}{\partial X_k}\right]^{\mathrm{T}} \lambda_{k+1} \tag{27}$$

where F_k stands for all the terms on the right hand side of Box I.

5.2.2. Selection of design parameters

For this problem, values of $\Delta t = 0.01$, $Q_w = I_3$ and $R_w = 1$, $\mathrm{tol}_a = \mathrm{tol}_c = 0.05$ and $\beta_c = \beta_a = 0.01$ were chosen and the domain of the state $S_I = \{X : |x_i| \le 1, \; i = 1, 2, 3\}$. The 'telescopic method' described in Section 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 selected for the convergence check.

Following the discussion in Section 5.1.2, in the AC synthesis the critic network was selected to have three sub-networks, each having a 3-6-1 structure. A 3-6-1 network was selected as the action network. Three sub-networks each having a 3-6-1 structure were used in the SNAC process. In each network, the hyperbolic tangent function was chosen for the input and hidden layers and a linear function was chosen for the output layer.

5.2.3. Analysis of results

Simulations were carried out using the same initial conditions for both AC and SNAC schemes. For demonstration purposes, the initial condition chosen is $[Q \;\; g \;\; \dot{g}]_{t=0}^{\mathrm{T}} = [9.85 \;\; 1.5 \;\; -1]^{\mathrm{T}}$. Figs. 10–12 show the trajectories of Q, g and $\dot{g}$ respectively for a time of twenty five seconds using both AC and SNAC techniques. These figures indicate that both the AC and SNAC schemes performed well to drive the states to their respective values. As before, the state trajectories from both AC and SNAC techniques are very similar to each other. Fig. 13 shows the control trajectory obtained from using the two schemes, which again shows that the SNAC technique is capable of obtaining the same optimal control solution as the AC technique. Note that the control trajectories in both schemes also drive toward a steady state value $V_{\mathrm{in}_0} = 0.05$. It can be seen from Fig. 11 that the position of the actuator has been forced to

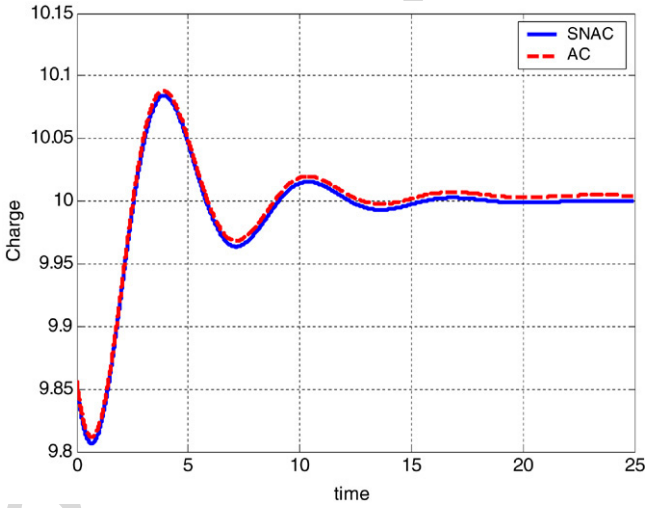

Fig. 10. SNAC/AC trajectories for charge (Electrostatic actuator).

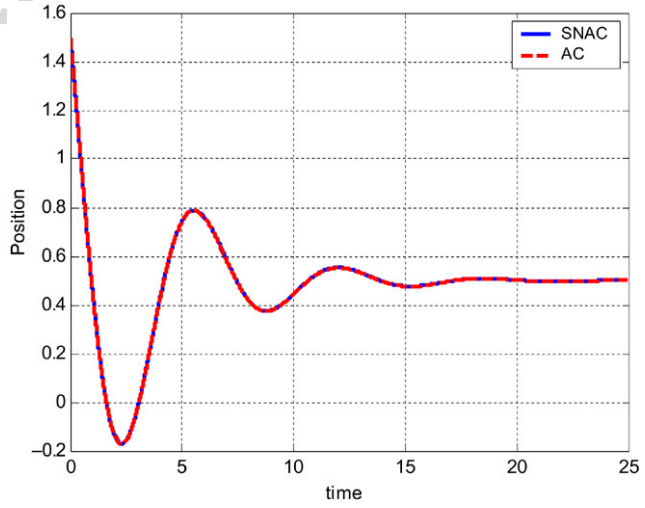

Fig. 11. SNAC/AC trajectories for position (Electrostatic actuator).

the desired value of 0.5 μm. 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.

We carried out analysis similar to the case in Section 5.1 for comparing the computational performances of the two techniques. Fig. 14 is an illustration of the times taken by each scheme under comparison (AC/SNAC) in our study for the MEMS problem, over ten independent runs.

It was observed that $\mu_{T_{\mathrm{SNAC}}} = 0.59 \; \mu_{T_{\mathrm{AC}}}$, where $\mu_{T_{\mathrm{SNAC}}} = 531.4063$ s and $\mu_{T_{\mathrm{AC}}} = 890.3377$ s are the mean times taken to train (including checking for convergence) in the SNAC and AC schemes respectively. It was also observed that

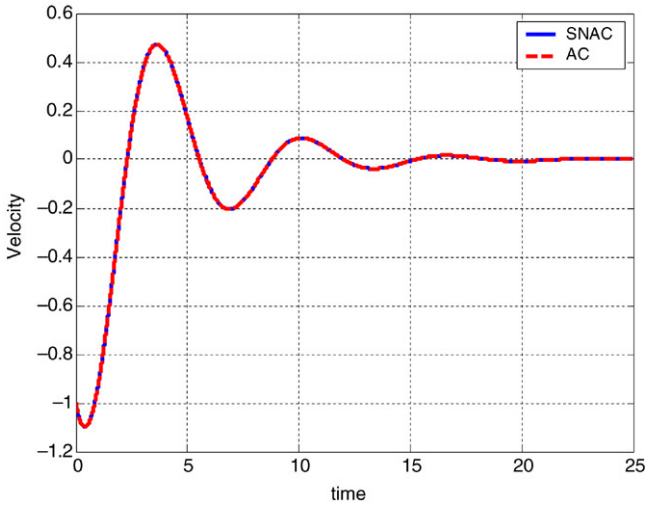

Fig. 12. SNAC/AC trajectories for velocity (Electrostatic actuator).

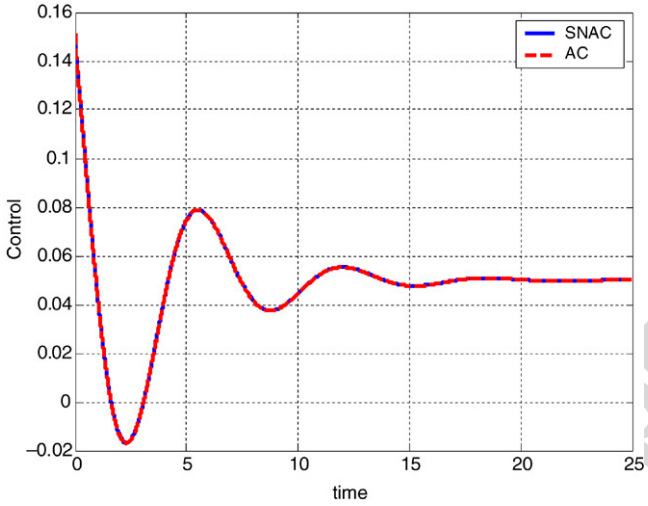

Fig. 13. Associated control trajectories (Electrostatic actuator).

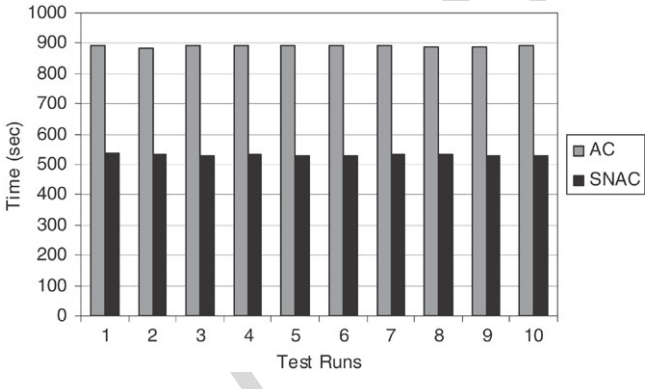

Fig. 14. Training time comparison [AC/SNAC] (Electrostatic actuator).

$\sigma_{T_{\text{SNAC}}} = 2.224001$ s and $\sigma_{T_{\text{AC}}} = 2.098956$ s, where $\sigma_{T_{\text{SNAC}}}$ and $\sigma_{T_{\text{AC}}}$ are the standard deviations for the SNAC and AC schemes respectively. The small standard deviation values indicate that there is not much variation in total time taken for each test run, and hence, both the techniques are fairly consistent. A two-sample t-test was performed on the data over the ten independent runs (similar to the exercise in Section 5.1.3). The test statistic obtained was $t_0 = 371.1623$. Since $t_0 = 371.1623 > t_{0.025,18} = 2.101$, we conclude that the two training time means are statistically different.

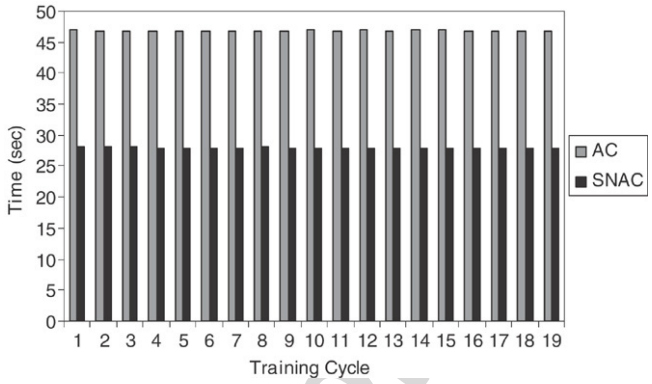

Fig. 15. Average training times for each step (Electrostatic actuator).

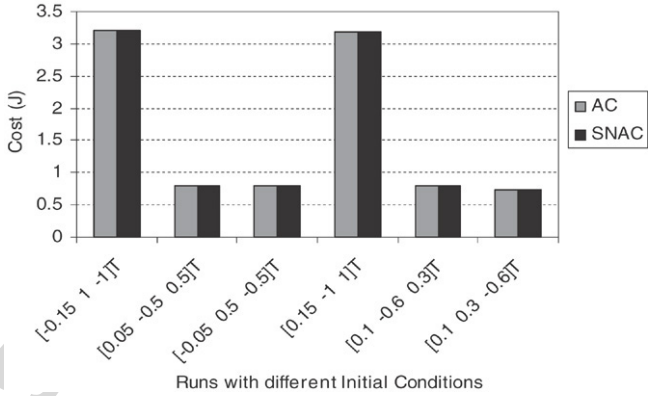

Fig. 16. Cost comparison for different initial conditions (Electrostatic actuator).

Fig. 15 compares the average times taken by AC and SNAC for each step in the telescopic training process discussed earlier. The average was taken over ten independent runs. From the figure, it is evident that the means of the two sets of data are significantly different. The cost comparison based on Eq. (25) for six different sets of initial conditions using the AC and SNAC techniques for a simulation of $t_f = 25$ s (large enough for the $t_f \to \infty$ approximation) has been given in Fig. 16. It can be seen that the costs for AC and SNAC schemes are very close to each other in each case. On conducting a paired comparison test (similar to the test mentioned in Section 5.1.3) on the two sets of cost data over six tests for the cost function, the test statistic was calculated to be $t_0 = 0.5371$. Since $t_0 = 0.5371 < t_{0.025,5} = 2.571$, we conclude that the two sets of cost data are not statistically different and hence the SNAC technique leads to the same optimal solution as the AC technique.

6. Conclusions

A new single network adaptive critic (SNAC) approach is presented for optimal control synthesis. 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. Huge computational savings with SNAC have been demonstrated by using two interesting examples. In addition, the MEMS problem also demonstrates that SNAC is applicable for complex engineering systems of practical significance. Note that while applying both AC and SNAC techniques discussed in this paper, we assume that the plant equations and parameters are known.

Acknowledgements

This research was supported by NSF-USA grants 0201076 and 0324428. The authors also express their gratitude to the anonymous reviewers, whose constructive criticisms led to substantial improvements in this paper.

Appendix

Consider a linear system described by $x_{k+1} = Ax_k + Bu_k$. The optimal control equation can be derived to be $u_k = -R^{-1}[B]^{\mathrm{T}}\lambda_{k+1}$. On substituting the optimal control equation in the state variable equation we get a closed form system as follows. The costate equation is also given below.

A.1. Dominant equations

$$\begin{aligned} x_{k+1} &= Ax_k - BR^{-1}B^{\mathrm{T}}\lambda_{k+1} \\ \lambda_k &= Qx_k + A^{\mathrm{T}}\lambda_{k+1} \end{aligned} \tag{A.1}$$

where
$x \in \mathbb{R}^{n\times 1}$ is the state variable, $\lambda \in \mathbb{R}^{n\times 1}$ is the costate variable, $Q \in \mathbb{R}^{n\times n} \geq 0$ is the penalty on states, $R \in \mathbb{R}^{m\times m} > 0$ is the penalty on control,
$A \in \mathbb{R}^{n\times n}$, $\det(A) \neq 0$, $B \in \mathbb{R}^{n\times m}$, (A, B) *controllable*.

A.2. Iterative process

The nonlinear relationship between the costate at step $k+1$ and the state at step k from the SNAC based critic neural network can be expressed by the following relation

$$\lambda_{k+1} = g(x_k). \tag{A.2}$$

On substituting Eq. (A.2) in the costate equation in Eq. (A.1), we get,

$$\begin{aligned} g^{n+1}(x_k) &= Qx_{k+1} + A^{\mathrm{T}}g^n(x_{k+1}) \\ x_{k+1} &= Ax_k - BR^{-1}B^{\mathrm{T}}g^n(x_k), \quad n = 0, 1, 2, 3, \ldots \end{aligned} \tag{A.3}$$

In the above equation, n is the training iteration number.

Eq. (A.3) can be re-written as

$$\begin{aligned} g^{n+1}(x_k) = {} & Q(Ax_k - BR^{-1}B^{\mathrm{T}}g^n(x_k)) \\ & + A^{\mathrm{T}}g^n(Ax_k - BR^{-1}B^{\mathrm{T}}g^n(x_k)). \end{aligned} \tag{A.4}$$

Claim 1. *Consider a linear system where the mapping between λ_{k+1} and x_k is linear. Let the initial approximation be the linear relation $\lambda_{k+1} = g^0 x_k$. The mappings obtained at each training iteration $g^1, g^2, \ldots, g^n$ will all be linear functions of x_k.*

Proof. By mathematical induction

(1) If $g^0(x_k) = g^0 x_k$, then $g^1(x_k) = g^1 x_k$

$$\begin{aligned} g^1(x_k) &= Q(Ax_k - BR^{-1}B^{\mathrm{T}}g^0x_k) \\ &\quad + A^{\mathrm{T}}g^0(Ax_k - BR^{-1}B^{\mathrm{T}}g^0x_k) \\ &= Q(A - BR^{-1}B^{\mathrm{T}}g^0)x_k \\ &\quad + A^{\mathrm{T}}g^0(A - BR^{-1}B^{\mathrm{T}}g^0)x_k \\ &= (Q + A^{\mathrm{T}}g^0)(A - BR^{-1}B^{\mathrm{T}}g^0)x_k \\ &\triangleq g^1 x_k \end{aligned}$$

(2) If $g^k(x_k) = g^k x_k$, then $g^{k+1}(x_k) = g^{k+1}x_k$

$$\begin{aligned} g^{k+1}(x_k) &= Q(Ax_k - BR^{-1}B^{\mathrm{T}}g^kx_k) \\ &\quad + A^{\mathrm{T}}g^k(Ax_k - BR^{-1}B^{\mathrm{T}}g^kx_k) \\ &= Q(A - BR^{-1}B^{\mathrm{T}}g^k)x_k \\ &\quad + A^{\mathrm{T}}g^k(A - BR^{-1}B^{\mathrm{T}}g^k)x_k \\ &= (Q + A^{\mathrm{T}}g^k)(A - BR^{-1}B^{\mathrm{T}}g^k)x_k \\ &\triangleq g^k x_k \end{aligned}$$

(3) $g^1, g^2, \ldots, g^n$ will all be linear functions of x_k.

Assuming a linear SNAC approximation, according to Claim 1, the recursive relation can be written as

$$\begin{aligned} g^{n+1}x_k = {} & Q(A - BR^{-1}B^{\mathrm{T}}g^n)x_k \\ & + A^{\mathrm{T}}g^n(A - BR^{-1}B^{\mathrm{T}}g^n)x_k. \end{aligned} \tag{A.5}$$

This relation exists for all x_k. The mapping g^{n+1} can be expressed as

$$g^{n+1} = (Q + A^{\mathrm{T}}g^n)(A - BR^{-1}B^{\mathrm{T}}g^n). \tag{A.6}$$

Claim 2. *If the iterative process of training the critic network in the SNAC method converges, the method converges to the solution of the algebraic Riccati equation.*

Proof. Assume the converged value of the linear mapping between x_k and λ_{k+1} to be g. We can write

$$g = (Q + A^{\mathrm{T}}g)(A - BR^{-1}B^{\mathrm{T}}g). \tag{A.7}$$

The Discrete Time Riccati equation is as follows

$$P = A^{\mathrm{T}}PA - A^{\mathrm{T}}PB(R + B^{\mathrm{T}}PB)^{-1}B^{\mathrm{T}}PA + Q. \tag{A.8}$$

It can be seen that the critic network in the SNAC structure ends up being the relation

$$g = \left(I + PBR^{-1}B^{\mathrm{T}}\right)^{-1} PA. \tag{A.9}$$

The matrix inversion lemma is as follows

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}. \tag{A.10}$$

Eq. (A.9) comes from the relations

$$\lambda_{k+1} = gx_k \tag{A.11}$$

$$\lambda_k = Px_k. \tag{A.12}$$

Substituting Eq. (A.9) in Eq. (A.7), the whole equation turns into an equation in P. The following is the proof to show that the resulting equation is the Riccati equation itself.

$$\begin{aligned}(I + PBR^{-1}B^{\mathrm{T}})^{-1}P &= Q + A^{\mathrm{T}}(I + PBR^{-1}B^{\mathrm{T}})^{-1} \\ &\times PA - QBR^{-1}B^{\mathrm{T}}(I + PBR^{-1}B^{\mathrm{T}})^{-1}P \\ &- A^{\mathrm{T}}(I + PBR^{-1}B^{\mathrm{T}})^{-1}PABR^{-1}B^{\mathrm{T}} \\ &\times (I + PBR^{-1}B^{\mathrm{T}})^{-1}P. \end{aligned} \tag{A.13}$$

Multiply both the left and right hand sides of Eq. (A.13) with $\left(P^{-1} + BR^{-1}B^{\mathrm{T}}\right)$. The left hand side of Eq. (A.13) can be reduced to

$$\begin{aligned} LS : (I + PBR^{-1}B^{\mathrm{T}})^{-1}P(P^{-1} + BR^{-1}B^{\mathrm{T}}) \\ = (P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1}(P^{-1} + BR^{-1}B^{\mathrm{T}}) \\ = I. \end{aligned} \tag{A.14}$$

The right hand side of Eq. (A.13) becomes

$$\begin{aligned} RS :=\ & Q(P^{-1} + BR^{-1}B^{\mathrm{T}}) + A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1} \\ & \times A(P^{-1} + BR^{-1}B^{\mathrm{T}}) \\ & - QBR^{-1}B^{\mathrm{T}} - A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1}ABR^{-1}B^{\mathrm{T}} \\ =\ & Q(P^{-1} + BR^{-1}B^{\mathrm{T}}) + (A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1} \\ & \times AP^{-1} + A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1}ABR^{-1}B^{\mathrm{T}}) \\ & - QBR^{-1}B^{\mathrm{T}} - A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1}ABR^{-1}B^{\mathrm{T}} \\ =\ & QP^{-1} + A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1}AP^{-1}. \end{aligned} \tag{A.15}$$

Equating the two sides we get

$$I = QP^{-1} + A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1}AP^{-1}. \tag{A.16}$$

That is,

$$P = Q + A^{\mathrm{T}}(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1}A. \tag{A.17}$$

From Eq. (A.10) we get

$$(P^{-1} + BR^{-1}B^{\mathrm{T}})^{-1} = P + PB(B^{\mathrm{T}}PB + R)^{-1}B^{\mathrm{T}}P. \tag{A.18}$$

Substitute Eq. (A.18) in Eq. (A.17) to obtain

$$P = Q + A^{\mathrm{T}}(P + PB(B^{\mathrm{T}}PB + R)^{-1}B^{\mathrm{T}})A. \tag{A.19}$$

The above equation turns out to be the discrete Riccatic equation as given below

$$P = Q + A^{\mathrm{T}}PA + A^{\mathrm{T}}PB(B^{\mathrm{T}}PB + R)^{-1}B^{\mathrm{T}}A. \tag{A.20}$$

This proves that for linear systems (where the mapping between the costate at stage $k + 1$ and the state at stage k is linear) the SNAC structure on convergence converges to the discrete Riccati equation.

References

Balakrishnan, S. N., & Biega, V. (1996). Adaptive-critic based neural networks for aircraft optimal control. *Journal of Guidance, Control and Dynamics*, *19*(4), 893–898.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neoronlike adaptive elements that can solve difficult control problems. *IEEE Transactions on Systems, Man and Cybernetics*, *SMC-13*, 834–846.

Bryson, A. E., & Ho, Y. C. (1975). *Applied optimal control*. Taylor and Francis.

Ferrari, S., & Stengel, R. F. (2002). An adaptive critic global controller, In *Proceedings of the American control conference* (pp. 2665–2670). Anchorage, USA.

Gopal, M. (1993). *Modern control system theory* (2nd ed.). Wiley.

Gupta, S. K. (1995). *Numerical methods for engineers*. Wiley Eastern Ltd. and New Age International Ltd.

Hagan, M. T., Demuth, H. B., & Beale, M. (1996). *Neural network design*. PWS Publishing Company.

Han, D., & Balakrishnan, S. N. (2002). Adaptive critics based neural networks for agile missile control. *Journal of Guidance, Control and Dynamics*, *25*, 404–407.

Hunt, K. J. (1992). Neural networks for control systems—A survey. *Automatica*, *28*(6), 1083–1112.

Lewis, F. (1992). *Applied optimal control and estimation*. Prentice-Hall.

Liu, X., & Balakrishnan, S. N. (2000). Convergence analysis of adaptive critic based optimal control, In *Proceedings of the American control conference* (pp. 1929–1933). Chicago, USA.

Miller, W. T., Sutton, R., & Werbos, P. J. (Eds.) (1990). *Neural networks for control*. MIT Press.

Montgomery, D. C. (2001). *Design and analysis of experiments* (5th ed.). John Wiley and Sons, Inc.

Murray, J. J., Cox, C. J., Lendaris, C. G., & Saeks, R. E. (2002). Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics- Part C: Applications and Reviews*, *32*, 140–153.

Padhi, R. (2001). Optimal control of distributed parameter systems using adaptive critic neural networks. Ph.D. Dissertation. University of Missouri Rolla.

Padhi, R., & Balakrishnan, S. N. (2003a). Optimal process control using neural networks. *Asian Journal of Control*, *5*(2), 217–229.

Padhi, R., & Balakrishnan, S. N. (2003b). Proper orthogonal decomposition based neurocontrol synthesis of a chemical reactor process using approximate dynamic programming. *Neural Networks*, *16*, 719–728.

Prokhorov, D. V., & Wunsch, D. C. II (1997). Adaptive critic designs. *IEEE Transactions on Neural Networks*, *8*, 997–1007.

Prokhorov, D. V. (2003). Optimal controllers for discretized distributed parameter systems, In *Proceedings of the American control conference* (pp. 549–554). Denver.

Senturia, S. D. (2001). *Microsystem design*. Kluwer Academic Publishers.

Venayagamoorthy, G. K., Harley, R. G., & Wunsch, D. C. (2002). Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbo generator. *IEEE Transactions on Neural Networks*, *13*, 764–773.

Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. In D. A. White, & D. A Sofge (Eds.), *Handbook of intelligent control*. Multiscience Press.

Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Yesildirek, A. (1994). Nonlinear systems control using neural networks. Ph.D. Thesis. Arlington: University of Texas.