J Control Theory Appl 2011 9 (3) 370–380
DOI 10.1007/s11768-011-0191-3

Approximate dynamic programming solutions with a single network adaptive critic for a class of nonlinear systems

Jie DING, S. N. BALAKRISHNAN
Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology, Rolla, MO 65401, U.S.A.

Abstract: Approximate dynamic programming (ADP) formulation implemented with an adaptive critic (AC)-based neural network (NN) structure has evolved as a powerful technique for solving the Hamilton-Jacobi-Bellman (HJB) equations. As interest in ADP and the AC solutions are escalating with time, there is a dire need to consider possible enabling factors for their implementations. A typical AC structure consists of two interacting NNs, which is computationally expensive. In this paper, a new architecture, called the 'cost-function-based single network adaptive critic (J-SNAC)' is presented, which eliminates one of the networks in a typical AC structure. This approach is applicable to a wide class of nonlinear systems in engineering. In order to demonstrate the benefits and the control synthesis with the J-SNAC, two problems have been solved with the AC and the J-SNAC approaches. Results are presented, which show savings of about 50% of the computational costs by J-SNAC while having the same accuracy levels of the dual network structure in solving for optimal control. Furthermore, convergence of the J-SNAC iterations, which reduces to a least-squares problem, is discussed; for linear systems, the iterative process is shown to reduce to solving the familiar algebraic Ricatti equation.

Keywords: Approximate dynamic programming; Optimal control; Nonlinear control; Adaptive critic; Cost-function-based single network adaptive critic; J-SNAC architecture

1 Introduction

Feedback control is the preferred solution for many systems because of its beneficial properties like robustness with respect to noise and modeling uncertainties. It is well-known that a dynamic programming formulation offers the most comprehensive solution approach to nonlinear optimal control in a state feedback form [1, 2]. However, solving the associated Hamilton-Jacobi-Bellman (HJB) equation demands a large (rather infeasible) amount of computation and storage space dedicated to this purpose. An innovative idea was proposed in [3] to get around this numerical complexity by using an ADP formulation. The solution to the ADP formulation is obtained through a dual NN approach called the adaptive critic (AC). In one version of the AC approach, called the heuristic dynamic programming (HDP), 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 the cost function to be minimized. Adaptive critic formulations can be found in many papers; some researchers have used ADP formulations to solve problems with finite state spaces in applications to behavioral and computer sciences and operations research and robotics [4–7]. Adaptive critics can also be considered in reinforcement learning designs [4, 5]. These formulations employ primarily cost-function-based adaptive critics that we consider in this paper. There are also many papers in the literature that use system science principles and neural networks to formulate the problems with applications to real-time feedback control of dynamic systems. In recent years, many researchers have paid more attention to ADP in order to obtain approximate solutions for the HJB equation [8–11]. Model-based synthesis of adaptive critic-based controllers presented by Balakrishnan [12], Prokhrov [13], and Venayagamoorthy [14] for systems driven by ordinary differential equations, and Padhi and Balakrishnan [15] for distributed parameter systems show that the ADP-based controllers stabilize the plants quite successfully. Ferrai et al. [16, 17] have implemented a global adaptive critic controller for a business jet. Yang et al. [18] apply an adaptive critic-based controller in a nonlinear nonaffine discrete-time system. Lendaris et al. [19] have successfully shown in simulations that the HDP method can prevent cars from skidding when driving over unexpected patches of ice. In fact, there are many variants of the AC designs [13]. Typically, the AC designs are formulated in a discrete framework. Hanselman et al. consider the use of continuous time adaptive critics in [20]. Recently, Lewis et al. [21] developed reinforcement learning methods, which require only output feedback and yet converge to an optimal controller.

While there has been a multitude of papers involving ACs, there is virtually no paper in the published literature that deals with the computational load (or alleviating it) associated with the AC designs. If the ADP formulation should find their way to engineering implementations, computationally efficient AC designs that show convergence are

Received 6 August 2010; revised 21 March 2011.
This work was supported by the National Aeronautics and Space Administration (NASA) (No.ARMD NRA NNH07ZEA001N-IRAC1), and the National Science Foundation (NSF).

badly needed. Balakrishnan et al. [22–25] has proposed a single network adaptive critic (SNAC) earlier. However, that structure was based on a critic network that outputs the costates as in the DHP. The problem in dealing with costates is that they have no physical meaning in an engineering problem like the physical states of a system. Therefore, it is very difficult to get an idea of the magnitude of costates; in a multivariable problem, different costates can have values that could vary by several orders of magnitudes making convergence of the critic network very difficult. In contrast, the use of cost in a critic network as in this paper has much more relevance and meaning. As opposed to the costates that have the same dimension as states in a multivariable problem and, therefore, demand a more diverse network structure and its training, the cost function is a scalar and, therefore, the critic network dimension is minimal. The contribution of this paper is a cost-based single network adaptive critic architecture. It captures the mapping between the states (at time k) and the optimal cost (from k to the end). Note that while costates have no physical meaning, the output of the cost network provides valuable information to the designer, which is an idea on the remaining cost involved at any stage of the process as a function of the states of the system. In fact, the J-SNAC proposed in this paper is applicable to the type of control-affine nonlinear problems presented in some recent papers [26–28] while saving about 50% computation time as compared to the typical HDP structure used in those papers due to the elimination of one network.

The rest of the paper is organized as follows: the development of optimal control equations through an ADP framework is provided in Section 2; how to solve the optimal control equations for a domain of input conditions with a dual network AC architecture is explained in Section 3; the J-SNAC structure and the steps to synthesize optimal control are developed in Section 4; and numerical results with two representative problems and the analysis of the AC and the J-SNAC in terms of accuracy and the tremendous savings in computing time with J-SNAC are presented in Section 5. Appendix 1 shows the reduction of the J-SNAC iterations to the Ricatti equation for linear systems, and Appendix 2 shows the steps in computing the J-SNAC weights by least-squares formulation.

2 Approximate dynamic programming

In this section, the principles of approximate (discrete) dynamic programming, which both the AC and the J-SNAC approaches rely upon, are described. An interested reader can find more details about the derivations in [3, 29]. Note that a prime requirement to apply the AC or the J-SNAC is to formulate the problem in discrete time. The control designer has the freedom to use any appropriate discretization scheme. For example, one can use the Euler approximation for the state equation and Trapezoidal approximation for the cost function [30]. In a discrete-time formulation, we want to find an admissible control U_k, which causes the system given by

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

to follow an admissible trajectory from an initial point X_0 to a final desired point X_N while minimizing a desired cost function J given by

$$J = \sum_{k=0}^{N-1} \Psi_k(X_k, U_k), \tag{2}$$

where the subscript k denotes the time step. X_k and U_k represent an $n \times 1$ state vector and an $m \times 1$ control vector, respectively, at time step k. The functions F 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 cost function leads to a regulator (infinite time) problem.

Remark 1 It is important to note that the control U_k must both stabilize the system on $\Omega \subset \mathbb{R}^{n\times 1}$ and make the cost functional value (2) finite so that the control is admissible [31].

The steps in obtaining optimal control are now described. First, the cost function (2) is rewritten to start from step k as

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

The cost, 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 costate vector at step k is defined as

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

The necessary condition for optimality is given by

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

Equation (6) can be further expanded as

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

The optimal control equation can, therefore, be written as

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

The costate equation is derived in the following way

$$\begin{aligned}\lambda_k &= \frac{\partial \Psi_k}{\partial X_k} + \frac{\partial J_{k+1}}{\partial X_k} = \frac{\partial \Psi_k}{\partial X_k} + \Big(\frac{\partial X_{k+1}}{\partial X_k}\Big)^{\mathrm{T}} \frac{\partial J_{k+1}}{\partial X_{k+1}} \\ &= \frac{\partial \Psi_k}{\partial X_k} + \Big(\frac{\partial X_{k+1}}{\partial X_k}\Big)^{\mathrm{T}} \lambda_{k+1}. \end{aligned} \tag{9}$$

In order to synthesize an optimal controller, the necessary conditions (1), (8), and (9) have to be solved simultaneously, along with appropriate boundary conditions.

For the regulator problems, 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 function are such that one can obtain explicit solution for the control variable in terms of the state and the cost variables from (8), J-SNAC is applicable. Note that many control affine nonlinear systems (of the form $X_{k+1} = f(X_k) + BU_k$) 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 into this class. In that case, the explicit expression for the control will be $U_k = -R^{-1}B^{\mathrm{T}}(\frac{\partial J_{k+1}}{\partial X_{k+1}})$. The major problem is how to find $(\frac{\partial J_{k+1}}{\partial X_{k+1}})$.

3 Adapive critic for optimal control synthesis

Approximate dynamic programming implemented with an adaptive critic (AC) neural network structure has evolved as a powerful alternate technique that obviates the need for excessive computations and storage requirements in solving optimal control problems. In this section, the AC approach for optimal control synthesis is described for comparison with the J-SNAC synthesis, which will be introduced later in details. In an AC framework, two NNs (called the action and critic networks) are iteratively trained.

3.1 State generation for neural network training

State generation is an important part of the training process for both the AC and the J-SNAC. For this purpose, define $S_i = \{X_k : X_k \in$ Domain of operation$\}$, where the action and critic networks have to be trained. This set is so chosen that its elements 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 problems, a stabilizing controller drives the states toward the origin. From this observation, a 'telescopic method' is arrived at as follows:

For $i = 1, 2, \ldots$, define the set $S_i = \{X_k : \|X_k\|_\infty \leqslant c_i\}$, where c_i is a positive constant. At the beginning, a small c_1 is fixed, and both the networks are trained with the states generated in S_1. After the networks converge (the convergence condition will be discussed in Section 3.3, 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 $c_i = c_1 + 0.05 \times (i-1)$ 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 for the action network, which captures the relationship between X_k and U_k, is shown as follows (Fig. 1):

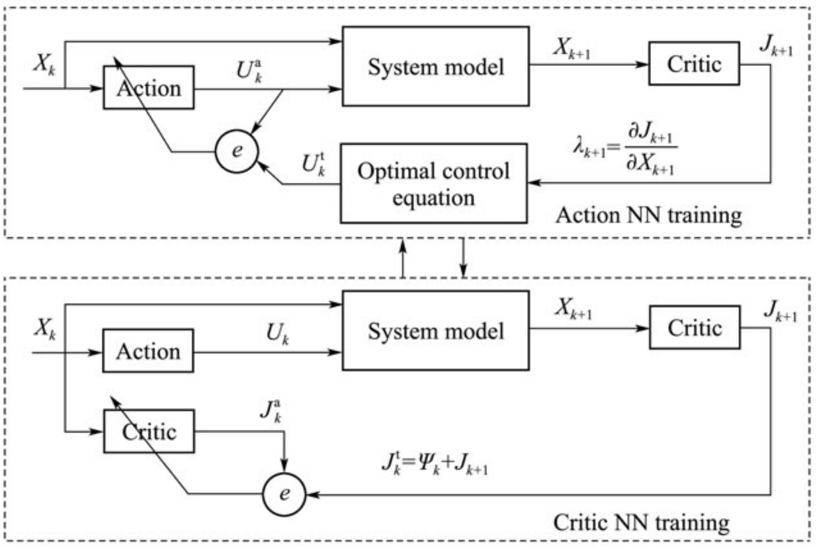


Fig. 1 AC network training scheme.

1) Generate S_i. For each element X_k of S_i, follow the steps:

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 J_{k+1};

d) Using X_k and J_{k+1}, calculate U_k^{t} (target U_k) by (8).

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

The steps for training the critic network, which captures the relationship between X_k and J_k, are listed as follows (Fig. 1):

1) Generate S_i. For each element X_k of S_i, follow the steps:

a) Input X_k to the critic network to obtain J_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 J_{k+1};

d) Using X_k, U_k and J_{k+1}, calculate J_k^{t} by (4).

2) Train the critic network for all X_k in S_i, the output being the corresponding J_k^{t}.

3.3 Convergence conditions

In order to check the convergence of the critic and the action networks, a set of new states, S_i^{c}, and target outputs are generated, as described in Section 3.2. Let the target outputs be J_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 J_k^{a} for the critic network and U_k^{a} for the action network. Tolerance values tol_{c} and tol_{a} are used as the convergence criteria for the critic and the action networks, respectively.

The following quantities are defined as relative errors: $e_{\mathrm{c}_k} \equiv \|J_k^{\mathrm{t}} - J_k^{\mathrm{a}}\|/\|J_k^{\mathrm{t}}\|$ and $e_{\mathrm{a}_k} \equiv \|U_k^{\mathrm{t}} - U_k^{\mathrm{a}}\|/\|U_k^{\mathrm{t}}\|$. Also, define $e_{\mathrm{c}} \equiv \{e_{\mathrm{c}_k}\}$, $k = 1, \ldots, |S_I|$, and $e_{\mathrm{a}} \equiv \{e_{\mathrm{a}_k}\}$, $k = 1, \ldots, |S_I|$. When $e_{\mathrm{c}} < tol_{\mathrm{c}}$, the convergence criterion for the critic network training is met, and when

$e_a < tol_a$, the convergence criterion for the action network training is met. Similarly, the convergence check in the J-SNAC scheme is carried out by defining the relative error $e_{c_k} \equiv \|J_k^t - J_k^a\|/\|J_k^t\|$ and $e_c \equiv \{e_{c_k}\}$, $k = 1, \dots, |S_I|$, and the training process is stopped when $e_c < tol_c$.

3.4 Initialization of networks: pretraining

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 'pretraining'. The following approach works well for quadratic regulator problems. First, the system dynamics in (1) is linearized [32], and a linear system in discrete time formulation is obtained as

$$X_{k+1} = A_D X_k + B_D U_k. \tag{10}$$

Using the standard discrete linear quadratic regulator (DLQR) optimal control theory [1], we can solve for the gain matrix K_D. The following relationships hold:

$$U_k = -K_D X_k, \tag{11}$$

$$J_k = \frac{1}{2} X_k^T S_k X_k, \tag{12}$$

where S_k is the solution sequence to the discrete time Riccati equation [1]. The critic and the action networks are initially trained with the static relationships given in (11) and (12), respectively. Intuitively, the idea was to start with the relationships that are 'close' to the optimal relationships (at least in a small neighborhood of the origin).

4 J-SNAC synthesis

In this section, the newly developed 'cost-function-based single network adaptive critic' (J-SNAC) technique is discussed in detail. As mentioned in Section 1, the J-SNAC technique retains all the powerful features of the AC methodology while eliminating the action network completely. In the J-SNAC design, the critic network captures the functional relationship between the state X_k and the optimal cost J_k. Note that the J-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 cost J_{k+1} (control affine systems with quadratic cost functions fall into this class), but such a restriction is not needed for the AC technique. As mentioned earlier, (1), (8), and (9) have to be solved simultaneously, along with appropriate boundary conditions, for the synthesis of optimal control.

Reduction of the two network problem (AC) to a single network (J-SNAC) facilitates the derivation of a formulation (See Appendix 2) that reduces the cost network training to solving a set of nonlinear algebraic equations in the weights of the neural network. By further manipulation, this set of equations is reduced to solving a recursive least-squares problem in Appendix 2. The network output J_k is a function of the network weights $\hat{W}^k$ and states X_k, which is given by (a20). (a24) is linear in network weights, and therefore, with enough sample points, we can solve for the network weights iteratively.

What facilitates the elimination of the action network is the fact that the necessary condition for optimality in (8) in terms of the derivative of the cost neural network output is used in J-SNAC for the computation of control in lieu of an action network output as in an AC design. For a control affine problem with a quadratic cost, note that (8) simply reduces to $U_k = -R^{-1} B^T (\frac{\partial J_{k+1}}{\partial X_{k+1}})$.

4.1 Neural network training

In the J-SNAC approach, the steps for training the critic network are given as follows (Fig. 2):

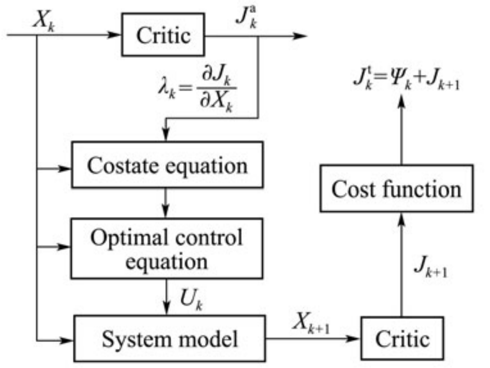


Fig. 2 J-SNAC network training scheme.

1) Generate S_i. For each element X_k of S_i, follow the steps:

a) Input X_k to the critic network to obtain $J_k = J_k^a$;

b) Calculate $\lambda_k = \dfrac{\partial J_k}{\partial X_k}$ and λ_{k+1} by (9);

c) Calculate U_k from (8);

d) Use X_k and U_k to get X_{k+1} from the state equation (1);

e) Input X_{k+1} to the critic network to get J_{k+1};

f) Use X_k, U_k, and J_{k+1}, to calculate J_k^t with (4).

2) Train the critic network for all X_k in S_i with the output corresponding with J_k^t, which means solving (a25) for network weights. The details are given in Appendix 2 (the weight update rule is given as

$$\hat{W}^{i+1} = (\Phi(X_k)^T \Phi(X_k))^{-1} \Phi(X_k)^T \mathrm{RHS}(X_k, \hat{W}^i)$$

in (a31)).

3) Check the convergence of the critic network. If yes, revert to step 1 with $i = i + 1$. Otherwise, repeat steps 1) and 2).

4) Continue steps 1)–3) until the input set $i = I$.

4.2 Convergence condition

First, a set S_i^c of states is generated, as explained in Section 3.1. Let these target output be J_k^t and outputs from the trained networks (using the same inputs from the set S_i^c) be J_k^a. A tolerance value tol_c is used to test the convergence of the critic network. By defining the relative error $e_{c_k} \equiv \|J_k^t - J_k^a\|/\|J_k^t\|$ and $e_c \equiv \{e_{c_k}\}$, $k = 1, \dots, |S_I|$, the training process is stopped when $e_c < tol_c$.

5 Numerical results

In this section, numerical results from two representative problems are reported. These are i) a Vander Pol's oscillator and ii) an electrostatic actuator. The goals of this study are a) to investigate the performance of the J-SNAC controller in stabilizing nonlinear systems and its accuracy versus the AC solutions and b) to compare and analyze the computation times of the J-SNAC and the AC. It should be clearly noted that the performance of both paradigms are expected to be the same since both are used to generate optimal control. The J-SNAC weights were initialized at zero in both examples.

Example 1 (Vander Pol's oscillator)

1) Problem description and optimality conditions.

The motivation for selecting the Vander Pol's oscillator problem is that it is a benchmark nonlinear problem [33]. 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. The system dynamics of a Vander Pol's oscillator is given by

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \alpha(1 - x_1^2)x_2 - x_1 + u. \end{cases} \tag{13}$$

Our goal is to drive $X \equiv [x_1 \;\; x_2]^{\mathrm{T}} \to 0$ as $t \to \infty$. A relevant quadratic cost function is formulated as

$$J = \frac{1}{2}\int_0^{\infty} (X^{\mathrm{T}} Q_{\mathrm{W}} X + R_{\mathrm{W}} u^2)\mathrm{d}t, \tag{14}$$

where $Q_{\mathrm{W}} \geqslant 0$ and $R_{\mathrm{W}} \geqslant 0$. Discretization of (13) and (14) leads to

$$\begin{cases} x_{1_{k+1}} = x_{1_k} + \Delta t x_{2_k}, \\ x_{2_{k+1}} = x_{2_k} + \Delta t(\alpha(1 - x_{1_k}^2)x_{2_k} - x_{1_k} + u_k), \end{cases} \tag{15}$$

$$J = \sum_{k=1}^{N\to\infty} \frac{1}{2}(X_k^{\mathrm{T}} Q_{\mathrm{W}} X_k + R_{\mathrm{W}} u_k^2)\Delta t. \tag{16}$$

2) Selection of design parameters.

For this problem, we chose

$$\Delta t = 0.001, \;\; Q_{\mathrm{W}} = \mathrm{diag}\{1, 2\}, \;\; R_{\mathrm{W}} = 1,$$
$$tol_{\mathrm{a}} = tol_{\mathrm{c}} = 10^{-4}, \;\; \alpha = 0.9.$$

The domain of interest (for NN training) was chosen to be $S_I = \{X : |x_i| \leqslant 1, i = 1, 2\}$. The 'telescopic method' was used for state generation. Each time, 2000 points were randomly selected for training the networks.

In AC synthesis, both the critic and the action network structures were selected to be 2-15-1 (i.e., two neurons in the input layer, 15 neurons in the hidden layer, and 1 neuron in the output layer). In J-SNAC synthesis, the cost J_k is a function of the network weights $\hat{W}^k$ and states X_k, which is given by (a20) as $J_k = (\hat{W}^k)^{\mathrm{T}}\phi(X_k)$. In the Vander Pol's oscillator problem, the basis functions $\phi(X)$ are selected as $[X_1 \;\; X_2 \;\; X_1^2 \;\; X_2^2 \;\; X_1X_2]^{\mathrm{T}}$ and m, and the number of sets in (a25) is selected as 2000.

3) Analysis of results.

After synthesizing the NNs, as discussed in Sections 3 and 4, simulation studies were carried out.

The plots of the states (position and velocity) trajectories with AC and J-SNAC are shown in Figs. 3 and 4, respectively. It can be seen that both the AC and J-SNAC techniques perform well in regulating the states (i.e., driving them to zero). Figs. 5 and 6 show that the corresponding control goes to zero as well. These results show that the J-SNAC approach is as good as the AC approach in synthesizing the optimal controller. Fig. 7 shows the convergence of the neural network weights of the J-SNAC by using the least-squares method (Appendix 2).

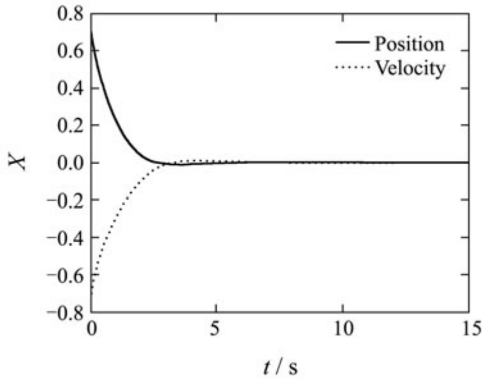


Fig. 3 Position and velocity trajectories (AC).

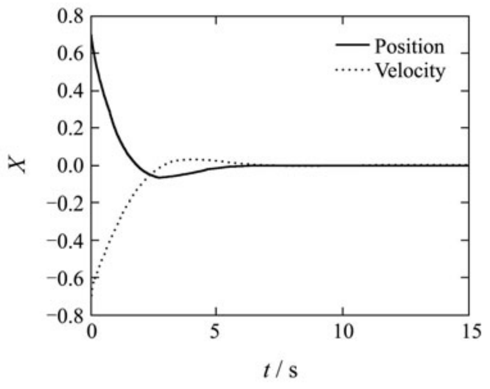


Fig. 4 Position and velocity trajectories (J-SNAC).

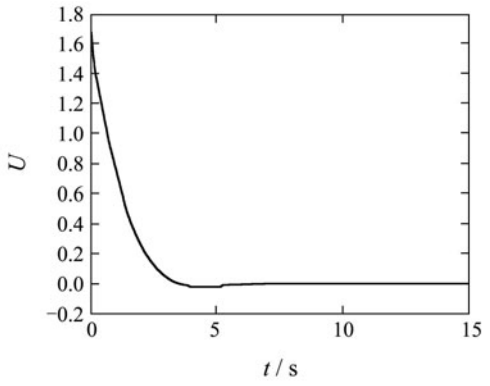


Fig. 5 Control history (AC).

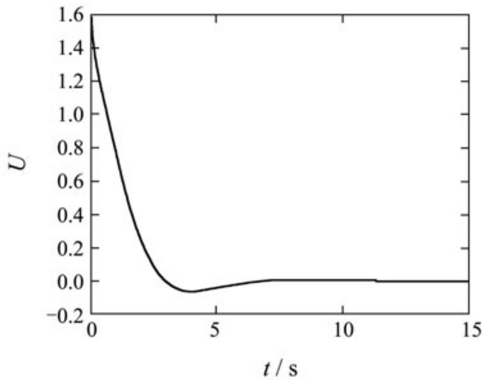


Fig. 6 Control history (J-SNAC).

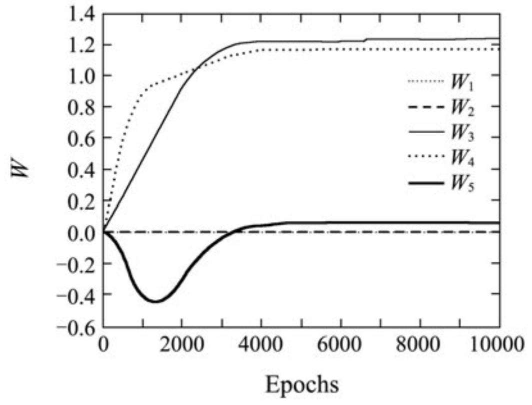


Fig. 7 History of the neural network weights (J-SNAC).

As pointed out earlier, one of the main advantages of the J-SNAC over the AC is that it results in computational savings since one network is eliminated. To demonstrate this quantitatively, both techniques were statistically analyzed based on 10 independent runs. Convergence times for the AC and the J-SNAC techniques to complete the training are presented in Fig. 8. This plot clearly indicates that as compared to the AC technique, it takes significantly less time to train the J-SNAC network. Note that $\mu_{T_{\text{J-SNAC}}} = 0.49\mu_{T_{\text{AC}}}$, where $\mu_{T_{\text{J-SNAC}}} = 733.7\,\text{s}$ and $\mu_{T_{\text{AC}}} = 1495.1\,\text{s}$ are the mean times taken to train by using the J-SNAC and the AC schemes respectively. Next, the data were examined as if they were a random sample from a normal distribution. The two-sample t-test is used to compare whether the two means are found to be significantly different. The statistic [34] used to test this is

$$t_0 = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{\frac{SD_1^2}{n_1} + \frac{SD_2^2}{n_2}}}, \tag{17}$$

where $\bar{y}_i$ is the sample mean of the ith group, is the sample size, SD_i is the standard deviation of each sample. To determine whether to reject the hypothesis that the two means of training time data are the same, t_0 was compared to the t distribution with $2n-2$ degrees of freedom. If $|t_0| > t_{\frac{\alpha}{2},2n-2}$ (where α is the significance level of the test) it would imply that the two means are different. It was seen that $t_0 = 244.82 > t_{0.025,18} = 2.101$. Hence, it can be concluded that the two means (training time using J-SNAC and training time using AC) are significantly different.

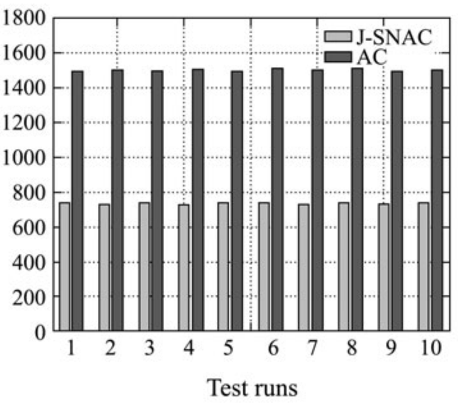


Fig. 8 Training time with both methods.

Example 2 (A MEMS actuator, shown in Fig. 9)

1) Problem statement and optimality conditions.

The next problem considered in this study is an MEMS device, namely, an electrostatic actuator [35]. This problem is used to demonstrate that the J-SNAC technique is applicable for complex engineering systems of practical significance.

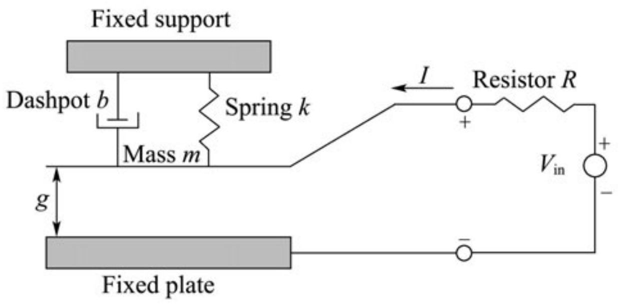


Fig. 9 Electrostatic actuator.

The governing equations are given by

$$\begin{cases} \dot{Q} - \dfrac{1}{R}(V_{\text{in}} - \dfrac{Qg}{\varepsilon A}) = 0, \\ m\ddot{g} + b\dot{g} + k(g - g_0) + \dfrac{Q^2}{2\varepsilon A} = 0, \end{cases} \tag{18}$$

where the various parameters used in (18) are shown in Table 1.

Table 1 Parameters used in modeling the actuator.

Parameter	Symbol	Value	Units
Area	A	100	μm^2
Permittivity	e	1	$\text{C}^2 \cdot \mu\text{m}^2/\text{N}$
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

$$Z = [z_1\ \ z_2\ \ z_3]^{\text{T}} = [Q\ \ g\ \ \dot{g}]^{\text{T}},$$

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, which in this study is selected as $0.5\,\mu$m. At the equilibrium point, $z_2 = 0.5, \dot{Z} = 0$. The equilibrium states are obtained as $Z_0 = [10\ \ 0.5\ \ 0]^{\text{T}}$, and the associated steady state controller value is $V_{\text{in}_0} = 0.05$. The deviated state is defined as

$$X = [x_1\ \ x_2\ \ x_3]^{\text{T}} \triangleq Z - Z_0$$

and the deviated control $u \triangleq (V_{\text{in}} - V_{\text{in}_0})$. In terms of these variables, the error dynamics is given by

$$\begin{cases} \dot{x}_1 = \dfrac{1}{R}(u - \dfrac{x_1}{2\varepsilon A} - \dfrac{x_2}{\sqrt{\varepsilon A}} - \dfrac{x_1 x_2}{\varepsilon A}), \\ \dot{x}_2 = x_3, \\ \dot{x}_3 = -\dfrac{1}{m}(\dfrac{x_1^2}{2\varepsilon A} + \dfrac{x_1}{\sqrt{\varepsilon A}} + kx_2 + bx_3 + \dfrac{1}{2} + \dfrac{k}{2} - \dfrac{g_0}{k}). \end{cases} \tag{19}$$

An optimal regulator problem can now be formulated to drive $X \to \infty$ with a cost function, J, which is the same as

(14). The state equation and cost function are discretized as

$$\begin{cases} x_{1_{k+1}} = x_{1_k} + \Delta t(\frac{1}{R}(u_k - \frac{x_{1k}}{2\varepsilon A} - \frac{x_{2k}}{\sqrt{\varepsilon A}} - \frac{x_{1k}x_{2k}}{\varepsilon A})), \\ x_{2_{k+1}} = x_{2_k} + \Delta t(x_{3k}), \\ x_{3_{k+1}} = x_{3_k} + \Delta t(-\frac{1}{m}(\frac{x_{1k}^2}{2\varepsilon A} + \frac{x_{1k}}{\sqrt{\varepsilon A}} + kx_{2k} + bx_{3k} \\ \qquad + \frac{1}{2} + \frac{k}{2} - \frac{g_0}{k})), \end{cases} \tag{20}$$

$$J = \sum_{k=1}^{N\to\infty} \frac{1}{2}(X_k^{\mathrm{T}} Q_{\mathrm{W}} X_k + R_{\mathrm{W}} u_k^2)\Delta t. \tag{21}$$

2) Selection of design parameters.

In this study, we chose $\Delta t = 0.001$, $Q_{\mathrm{W}} = I_3$, $R_{\mathrm{W}} = 10$, and $tol_{\mathrm{a}} = tol_{\mathrm{c}} = 10^{-4}$. The domain of interest (for NN training) was chosen to be $S_I = \{X : |x_i| \leqslant 1, i = 1, 2, 3\}$. The 'telescopic method' was used for state generation. Each time, 2000 training points were randomly selected.

In the AC synthesis, both the critic and the action networks were selected as a 3-25-1. In the J-SNAC synthesis, the basis function $\phi(X)$ is selected as

$$[X_1\ X_2\ X_3\ X_1^2\ X_2^2\ X_3^2\ X_1X_2\ X_2X_3\ X_1X_3]^{\mathrm{T}},$$

and m in (a25) is selected as 2000.

3) Analysis of results.

Simulations were carried out by using the same initial conditions for both the AC and the J-SNAC schemes.

Time histories of Q, g, and $\dot{g}$ from using the AC are shown in Fig. 10, Fig. 11, and Fig. 12, respectively and by J-SNAC are shown in Fig. 13, Fig. 14, and Fig. 15, respectively.

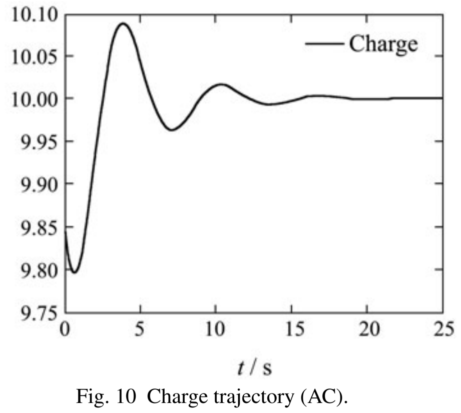


Fig. 10 Charge trajectory (AC).

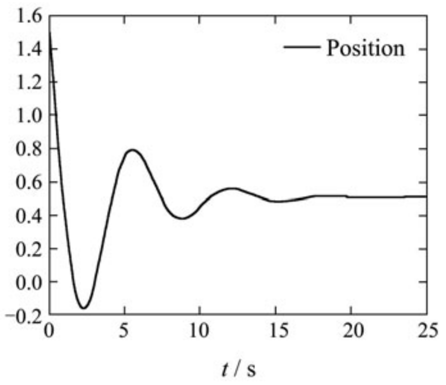


Fig. 11 Position trajectory (AC).

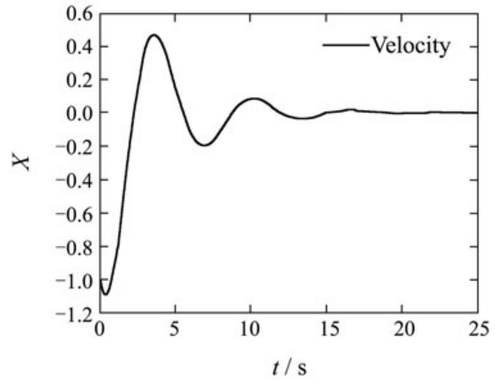


Fig. 12 Velocity trajectory (AC).

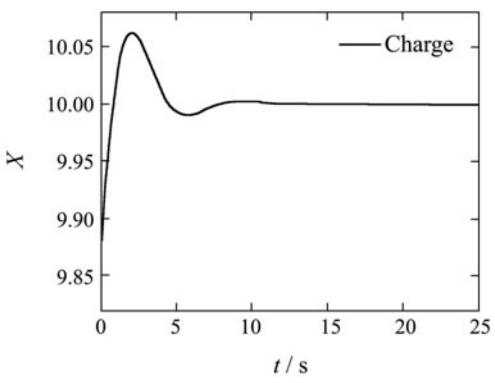


Fig. 13 Charge trajectory (J-SNAC).

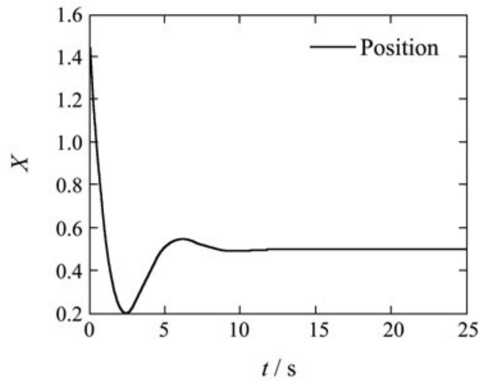


Fig. 14 Position trajectory (J-SNAC).

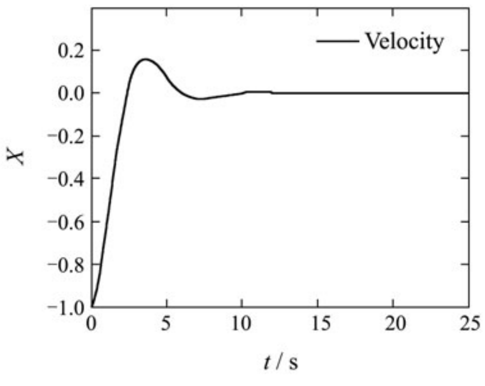


Fig. 15 Velocity trajectory (J-SNAC).

Figs. 16 and 17 show the control histories obtained from using the AC and the J-SNAC schemes, respectively. These figures indicate that both the AC and the J-SNAC performed well in driving the states to their equilibrium values.

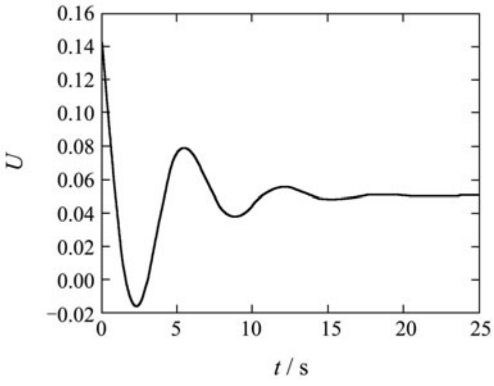


Fig. 16 Control history (AC).

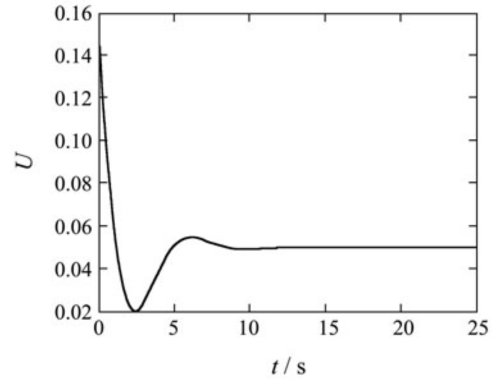


Fig. 17 Control history (J-SNAC).

Fig. 18 shows the convergence of the neural network weights of the J-SNAC by using least-squares method (Appendix 2).

Convergence times for the AC and the J-SNAC techniques to complete the training are presented in Fig. 19.

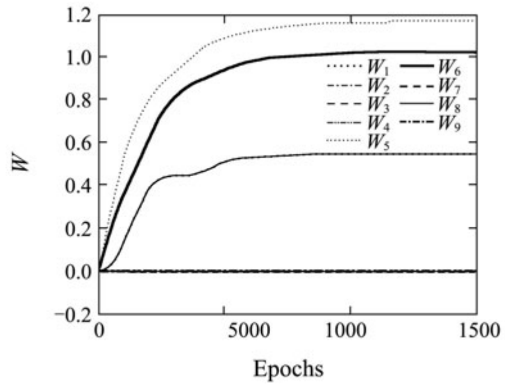


Fig. 18 History of the neural network weights (J-SNAC).

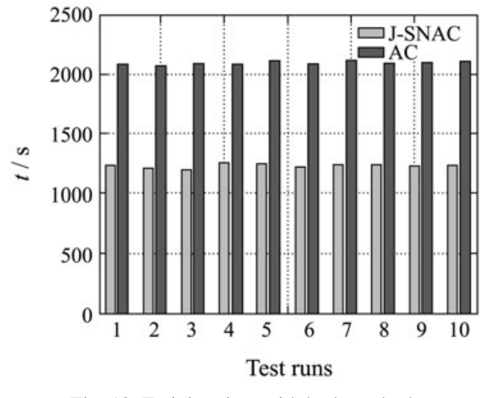


Fig. 19 Training time with both methods.

This plot clearly indicates that as compared to the AC technique, it takes significantly less time to train the J-SNAC network. Note that $\mu_{\mathrm{T_{J-SNAC}}} = 0.60\mu_{\mathrm{T_{AC}}}$, where $\mu_{\mathrm{T_{J-SNAC}}} = 1229.0$ sec and $\mu_{\mathrm{T_{AC}}} = 2062.1$ sec are the mean times taken to train by using the J-SNAC and the AC schemes, respectively. It was seen that by using (17), $t_0 = 116.06 > t_{0.025,18} = 2.101$. Hence, we conclude that the two means are significantly different.

6 Conclusions

A neural-network-based approach, called J-SNAC, for optimal control synthesis was presented. J-SNAC is applicable to the class of control-affine nonlinear systems that can be used to characterize many engineering systems, such as robots, aircraft, MEMS, etc. While the performance of both procedures are about the same, the use of this single network J-SNAC was shown to result in significant reductions in computation time over a typical two-network solution. This factor may enable the AC design to be implemented in real-life problems. The weight training for the J-SNAC was shown to reduce to just a recursive least-squares procedure. It was further shown that for linear systems with a quadratic cost, the iterative process could be reduced to the familiar algebraic Ricatti equation (See Appendix 1).

Acknowledgements

The views of the authors do not necessarily represent the views of the NSF and NASA.

References

[1] F. L. Lewis. *Applied Optimal Control and Estimation*. New York: Prentice Hall, 1992.

[2] A. E. Bryson, Y. C. Ho. *Applied Optimal Control*. London: Taylor & Francis, 1975.

[3] P. J. Werbos. Approximate dynamic programming for real-time control and neural modeling. *Handbook of Intelligent Control*, New York: Van Nostrand, 1992: 493 – 525.

[4] A. G. Barto. Connectionist learning for control: an overview. *Neural Networks for Control*. Cambridge: MIT Press, 1991: 5 – 58.

[5] A. Barto, T. Dieterich. Reinforcement learning and its relation to supervised learning. *Learning and Approximate Dynamic Programming*. Piscataway: Wiley-IEEE Press, 2004: 47 – 63.

[6] W. Powell, B. Van Roy. ADP for high-dimensional resource allocation problems. *Learning and Approximate Dynamic Programming*. Piscataway: Wiley-IEEE Press, 2004: 261 – 283.

[7] D. P. Bertsekas, J. N. Tsitsiklis. *Neuro-dynamic Programming*. Belmont: Athena Scientific, 1996.

[8] A. Al-Tamimi, F. L. Lewis, M. Abu-Khalaf. Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *IEEE Transactions on Systems, Man, Cybernetics – Part B*, 2008, 38(4): 943 – 949.

[9] S. N. Balakrishnan, J. Ding, F. L. Lewis. Issues on stability of adp feedback controllers for dynamical systems. *IEEE Transaction on Systems, Man, Cybernetics – Part B*, 2008, 38(4): 913 – 917.

[10] B. Li, J. Si. Robust dynamic programming for discounted infinite-horizon markov decision processes with uncertain stationary transition matrices. *Proceedings of IEEE International Symposiom Approximate Dynamic Programming and Reinforcement Learning*, New York: IEEE, 2007: 96 – 102.

[11] P. J. Werbos. Using ADP to understand and replicate brain intelligence: the next level design. *Proceedings of IEEE Symposium*

Approximately Dynamic Programming and Reinforcement Learning, New York: IEEE, 2007: 209 – 216.

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

[13] D. Prokhorov, D. Wunsch. Adaptive critic designs. *IEEE Transactions on Neural Networks*, 1995, 8(9): 1367 – 1372.

[14] G. Venayagamoorthy, R. Harley, D. Wunsch. Dual heuristic programming excitation neurocontrol for generators in a multimachine power system. *IEEE Transactions on Industry Applications*, 2003, 39(2): 382 – 384.

[15] R. Padhi, N. Unnikrishnan, X. Wang, et al. A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems. *Neural Network*, 2006, 19(10): 1648 – 1660.

[16] S. Ferrari, R. Stengel. An adaptive critic global controller. *American Control Conference*, New York: IEEE, 2002: 2665 – 2670.

[17] S. Ferrari, R. Stengel. Classical/neural synthesis of nonlinear control systems. *Journal of Guidance, Control and Dynamics*, 2002, 25(3): 442 – 448.

[18] Q. Yang, J. Vance, S. Jagannathan. Control of nonaffine nonlinear discrete-time systems using reinforcement-learning-based linearly parameterized neural networks. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 2008, 38(4): 994 – 1001.

[19] G. Lendaris, L. Schultz, T. Shannon. Adaptive critic design for intelligent steering and speed control of a 2-axle vehicle. *IEEE/INNS/ENNS International Joint Conference on Neural Networks*, Los Alamitos, CA: IEEE Computer Society, 2000: 73 – 78.

[20] T. Hanselmann, L. Noakes, A. Zaknich. Continuous time adaptive critics. *IEEE Transactions on Neural Networks*, 2007, 18(3): 631 – 647.

[21] F. L. Lewis, K. G. Vamvoudakis. Optimal adaptive control for unknown systems using output feedback by reinforcement learning methods. *Proceedings of the 8th IEEE International Conference on Control & Automation*, New York: IEEE, 2010: 2138 – 2145.

[22] R. Padhi, S. N. Balakrishnan. Optimal beaver population management using reduced order distributed parameter model and single network adaptive critics. *American Control Conference*, New York: IEEE, 2004: 1598 – 1603.

[23] R. Padhi, N. Unnikrishnan, S. N. Balakrishnana. Optimal control synthesis of a class of nonlinear systems using single network adaptive critics. *American Control Conference*, New York: IEEE, 2004: 1592 – 1597.

[24] V. Yadav, R. Padhi, S. N. Balakrishnan. Robust/optimal temperature profile control using neural networks. *Proceedings of IEEE International Conference on Control Applications*, New York: IEEE, 2006: 1986 – 1991.

[25] S. Chen, Y. Yang, N. Nguyen, et al. SNAC convergence and use in adaptive autopilot design. *International Joint Conference on Neural Networks*, New York: IEEE, 2009: 530 – 537.

[26] L. Yang, J. Si, K. S. Tsakalis, et al. Direct heuristic dynamic programming for nonlinear tracking control with filtered tracking error. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 2009, 39(6): 1617 – 1622.

[27] F. Wang, H. Zhang, D. Liu. Adaptive dynamic programming: an introduction. *IEEE Computational Intelligence Magazine*, 2009, 4(2): 39 – 47.

[28] H. Zhang, Q. Wei, Y. Luo. A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 2008, 38(4): 937 – 942.

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

[30] S. K. Gupta. *Numerical Methods for Engineers*. New Delhi: New Age International Publishers, Wiley Eastern Limited, 1995.

[31] R. W. Beard. *Improving the Closed-loop Performance of Nonlinear Systems*. Ph.D. thesis. New York: Rensselaer Polytechnic Institute, 1995.

[32] M. Gopal. *Modern Control System Theory*. 2nd ed, New York: John Wiley & Sons, 1993.

[33] A. Yesildirek. *Nonlinear Systems Control Using Neural Networks*. Ph.D. thesis. Arlington: University of Texas, 1994.

[34] D. Shirley, W. Stanley. *Statistics for Research*. 2nd ed, New York: John Wiley & Sons, 1991.

[35] S. D. Senturia. *Microsystem Design*. Netherlands: Kluwer Academic Publishers, 2001.

Appendix 1

A.1 Optimal control equations

Consider a linear system described by

$$X_{k+1} = AX_k + BU_k. \tag{a1}$$

The optimal control, U_k, for a quadratic cost is given by

$$U_k = -R^{-1}B^{\mathrm{T}}\lambda_{k+1}. \tag{a2}$$

On substituting the optimal control equation in the state variable equation and cost function, we obtain

$$X_{k+1} = AX_k - BR^{-1}B^{\mathrm{T}}\lambda_{k+1}, \tag{a3}$$

$$J_k = J_{k+1} + \frac{1}{2}X_k^{\mathrm{T}}QX_k + \frac{1}{2}\lambda_{k+1}^{\mathrm{T}}BR^{-1}B^{\mathrm{T}}\lambda_{k+1}, \tag{a4}$$

where $X \in \mathbb{R}^{n\times 1}$ is the state variable. $\lambda \in \mathbb{R}^{n\times 1}$ is the costate variable. $Q \in \mathbb{R}^{n\times n} \geqslant 0$, $R \in \mathbb{R}^{m\times m} > 0$ are the penalties on states and control, respectively.

A.2 Convergence to discrete-time Riccati equation

The nonlinear relationship between the cost and the state at step k can be expressed by the relation

$$J_k = g(X_k). \tag{a5}$$

On substituting (a5) in (a4), we obtain

$$g^{n+1}(X_k) = g^n(X_{k+1}) + \frac{1}{2}(X_k^{\mathrm{T}}QX_k + (U_k^n)^{\mathrm{T}}RU_k^n). \tag{a6}$$

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

Claim 1 Consider a linear system where the mapping between J_k and X_k is quadratic. Let the initial approximation be the quadratic relation $J_k = \frac{1}{2}X_k^{\mathrm{T}}G^0X_k$, and the mappings obtained at each training iteration $g^1(X_k), g^2(X_k), \dots, g^n(X_k)$ will all be quadratic functions of X_k.

Proof By mathematical induction

1) If $g^0(X_k) = \frac{1}{2}X_k^{\mathrm{T}}G^0X_k$, then $g^1(X_k) = \frac{1}{2}X_k^{\mathrm{T}}G^1X_k$.

From (a6), we obtain

$$g^1(X_k) = g^0(X_{k+1}) + \frac{1}{2}(X_k^{\mathrm{T}}QX_k + (U_k^0)^{\mathrm{T}}RU_k^0), \tag{a7}$$

where

$$\begin{aligned} U_k^0 &= -R^{-1}B^{\mathrm{T}}\lambda_{k+1} = -R^{-1}B^{\mathrm{T}}\frac{\partial J_{k+1}^0}{\partial X_{k+1}} \\ &= -R^{-1}B^{\mathrm{T}}(A^{\mathrm{T}})^{-1}(\frac{\partial J_k^0}{\partial X_k} - QX_k) \\ &= -R^{-1}B^{\mathrm{T}}(A^{\mathrm{T}})^{-1}(G^0 - Q)X_k. \end{aligned} \tag{a8}$$

Substituting (a8) into (a7), we obtain

$$\begin{aligned} &g^1(X_k) \\ &= \frac{1}{2}X_{k+1}^{\mathrm{T}}G^0X_{k+1} + \frac{1}{2}X_k^{\mathrm{T}}QX_k \\ &\quad + \frac{1}{2}X_k^{\mathrm{T}}(G^0-Q)^{\mathrm{T}}(A)^{-1}BR^{-1}B^{\mathrm{T}}(A^{\mathrm{T}})^{-1}(G^0-Q)X_k \\ &= \frac{1}{2}X_k^{\mathrm{T}}((A - BR^{-1}B^{\mathrm{T}}(A^{\mathrm{T}})^{-1}(G^0-Q))^{\mathrm{T}}G^0 \end{aligned}$$

$$\begin{aligned}&\cdot (A - BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^0 - Q)) + Q\\&+(G^0 - Q)^{\mathrm T}(A)^{-1}BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^0 - Q))X_k\\=&\ \frac{1}{2}X_k^{\mathrm T}G^1X_k.\end{aligned}$$

2) If $g^k(X_k) = \frac{1}{2}X_k^{\mathrm T}G^kX_k$, then

$$g^{k+1}(X_k) = \frac{1}{2}X_k^{\mathrm T}G^{k+1}X_k,$$

$$\begin{aligned}&g^{k+1}(X_k)\\&= \frac{1}{2}X_{k+1}^{\mathrm T}G^kX_{k+1} + \frac{1}{2}X_k^{\mathrm T}QX_k\\&+\frac{1}{2}X_k^{\mathrm T}(G^k - Q)^{\mathrm T}(A)^{-1}BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^k - Q)X_k\\&= \frac{1}{2}X_k^{\mathrm T}((A - BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^k - Q))^{\mathrm T}G^k\\&\cdot (A - BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^k - Q)) + Q\\&+(G^k - Q)^{\mathrm T}(A)^{-1}BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^k - Q))X_k\\&= \frac{1}{2}X_k^{\mathrm T}G^{k+1}X_k.\end{aligned}$$

3) $g^1(X_k), g^2(X_k), \dots, g^n(X_k)$ are all quadratic functions of X_k.

By Claim 1, the recursive relation can be written as

$$\begin{aligned}&\frac{1}{2}X_k^{\mathrm T}G^{n+1}X_k\\=&\ \frac{1}{2}X_k^{\mathrm T}((A - BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^n - Q))^{\mathrm T}\\&\cdot G^n(A - BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^n - Q)) + Q\\&+(G^n-Q)^{\mathrm T}(A)^{-1}BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^n-Q))X_k.\end{aligned}\quad \text{(a9)}$$

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

$$\begin{aligned}G^{n+1} =&\ (A - BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^n - Q))^{\mathrm T}\\&\cdot G^n(A - BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^n - Q)) + Q\\&+(G^n-Q)^{\mathrm T}(A)^{-1}BR^{-1}B^{\mathrm T}(A^{\mathrm T})^{-1}(G^n-Q).\end{aligned}\quad \text{(a10)}$$

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

Proof Consider that the mapping between J_k and X_k at each training iteration is $J_k^n = g^n(X_k) = \frac{1}{2}X_k^{\mathrm T}G^nX_k$.

At step $k+1$,

$$J_{k+1}^n = \frac{1}{2}X_{k+1}^{\mathrm T}G^nX_{k+1}. \quad \text{(a11)}$$

Therefore,

$$\lambda_{k+1} = \frac{\partial J_{k+1}}{\partial X_{k+1}} = G^nX_{k+1}. \quad \text{(a12)}$$

Substituting (a12) into (a2), we obtain

$$\begin{aligned}U_k =&\ -R^{-1}B^{\mathrm T}G^nX_{k+1} = -R^{-1}B^{\mathrm T}G^n(AX_k + BU_K)\\=&\ -R^{-1}B^{\mathrm T}G^nAX_k - R^{-1}B^{\mathrm T}G^nBU_k.\end{aligned}\quad \text{(a13)}$$

Moving the last term of (a13) to the LHS, we obtain

$$(I + R^{-1}B^{\mathrm T}G^nB)U_k = -R^{-1}B^{\mathrm T}G^nAX_k,$$

which leads to

$$\begin{aligned}U_k =&\ -(I + R^{-1}B^{\mathrm T}G^nB)R^{-1}B^{\mathrm T}G^nAX_k\\=&\ -(B^{\mathrm T}G^nB + R)^{-1}B^{\mathrm T}G^nAX_k.\end{aligned}\quad \text{(a14)}$$

Substituting (a14) into (a1), we obtain

$$\begin{aligned}X_{k+1} =&\ AX_k + BU_k\\=&\ [A - B(B^{\mathrm T}G^nB + R)^{-1}B^{\mathrm T}G^nA]X_k.\end{aligned}\quad \text{(a15)}$$

Substituting (a12) and (a15) into (a4), we obtain

$$\begin{aligned}&\frac{1}{2}X_k^{\mathrm T}G^{n+1}X_k\\=&\ \frac{1}{2}X_k^{\mathrm T}M^{\mathrm T}G^nMX_k + \frac{1}{2}X_k^{\mathrm T}QX_k\\&+\frac{1}{2}X_k^{\mathrm T}[(B^{\mathrm T}G^nB+R)^{-1}]^{\mathrm T}R(B^{\mathrm T}G^nB+R)^{-1}B^{\mathrm T}G^nAX_k,\end{aligned}\quad \text{(a16)}$$

where $M = A - B(B^{\mathrm T}G^nB + R)^{-1}B^{\mathrm T}G^nA$.

In (a16), every term is premultiplied by $X_k^{\mathrm T}$ and postmultiplied by X_k; since it should hold for all X_k, we obtain

$$\begin{aligned}G^{n+1} =&\ M^{\mathrm T}G^nM + Q + [(B^{\mathrm T}G^nB + R)^{-1}B^{\mathrm T}G^nA]^{\mathrm T}\\&\cdot R(B^{\mathrm T}G^nB + R)^{-1}B^{\mathrm T}G^nA\\=&\ A^{\mathrm T}G^nA - A^{\mathrm T}G^nB(B^{\mathrm T}G^nB + R)^{-1}B^{\mathrm T}G^nA + Q,\end{aligned}\quad \text{(a17)}$$

which leads to

$$\begin{aligned}G^{n+1} =&\ A^{\mathrm T}G^nA - A^{\mathrm T}G^nB(B^{\mathrm T}G^nB + R)^{-1}B^{\mathrm T}G^nA\\&+Q,\end{aligned}\quad \text{(a18)}$$

which is nothing but a discrete-time Riccati equation in G.

Appendix 2

This appendix is derived from the convergence proof by [8]. The difference is that the action network is eliminated in this study.

Consider a discrete nonlinear control-affine system

$$X_{k+1} = f(X_k) + BU_k, \quad \text{(a19)}$$

where X_k is an $n \times 1$ vector, U_k is an $m \times 1$ vector, $f(\cdot)$ can be a nonlinear function of the states, and B is a constant $n \times m$ matrix.

J_k is a function of the network weights $\hat W^k$ and states X_k

$$J_k = (\hat W^k)^{\mathrm T}\phi(X_k), \quad \text{(a20)}$$

$$\lambda_k = \frac{\partial((\hat W^k)^{\mathrm T}\phi(X_k))}{\partial X_k}. \quad \text{(a21)}$$

In the ith iteration, the control is computed as

$$U_k^i = -R^{-1}B^{\mathrm T}A^{-1}(\frac{\partial(\hat W^i)^{\mathrm T}\varphi(X_k)}{\partial X_k} - QX_k), \quad \text{(a22)}$$

where $A \equiv \frac{\partial f(X_k)}{\partial X_k}$.

Cost relationship at stages k and $k+1$ is given by

$$J_k^i = \frac{1}{2}(X_k^{\mathrm T}QX_k + (U_k^i)^{\mathrm T}RU_k^i) + J_{k+1}^i. \quad \text{(a23)}$$

Note that the first term on the right-hand side of (a23) does not need to be quadratic. Substituting (a20) into (a23), we obtain

$$\begin{aligned}(\hat W^{i+1})^{\mathrm T}\phi(X_k) =&\ \frac{1}{2}(X_k^{\mathrm T}QX_k + (U_k^i)^{\mathrm T}RU_k^i)\\&+(\hat W^i)^{\mathrm T}\phi(X_{k+1}).\end{aligned}\quad \text{(a24)}$$

Equation (a24) is linear in $\hat W^{i+1}$ with m unknowns, where m is number of elements of vector $\phi(X_k)$. Taking the transpose of (a24), and selecting m sets of states X_k called $X_k^{(1)}$ to $X_k^{(m)}$, it ends up with m equations with m unknowns:

$$\begin{cases}\phi(X_k^{(1)})^{\mathrm T}\hat W^{i+1} = \Psi_k^{(1)} + \phi(X_{k+1}^{(1)})^{\mathrm T}\hat W^i,\\ \quad\vdots\\ \phi(X_k^{(m)})^{\mathrm T}\hat W^{i+1} = \Psi_k^{(m)} + \phi(X_{k+1}^{(m)})^{\mathrm T}\hat W^i,\end{cases}\quad \text{(a25)}$$

where

$$\Psi_k^{(j)} = \frac{1}{2}((X_k^{(j)})^{\mathrm T}QX_k^{(j)} + (U_k^{i,(j)})^{\mathrm T}RU_k^{i,(j)})\quad \text{for } j = 1, 2, \dots, m,$$

$$U_k^{i,(j)} = -R^{-1}B^{\mathrm T}A^{-1}(\frac{\partial(\hat W^i)^{\mathrm T}\varphi(X_k^{(j)})}{\partial X_k^{(j)}} - QX_k^{(j)}), \quad \text{(a26)}$$

$$X_{k+1}^{(j)} = f(X_k^{(j)}) + BU_k^{i,(j)}. \quad \text{(a27)}$$

Equations (a25) can be rewritten as

$$\Phi(X_k)\hat{W}^{i+1} = \mathrm{RHS}(X_k, \hat{W}^i), \tag{a28}$$

where the RHS is the $m \times 1$ vector composed of all the right-hand side of (a25), and the $m \times m$ matrix $\Phi(X_k)$ is given by

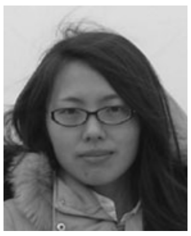
$$\Phi(X_k) = \begin{bmatrix} \phi(X_k^{(1)})^{\mathrm{T}} \\ \vdots \\ \phi(X_k^{(m)})^{\mathrm{T}} \end{bmatrix}, \ X_k = \begin{bmatrix} (X_k^{(1)})^{\mathrm{T}} \\ \vdots \\ (X_k^{(m)})^{\mathrm{T}} \end{bmatrix}. \tag{a29}$$

By using (a29) in (a28) leads to a recursive relationship for the network weights as

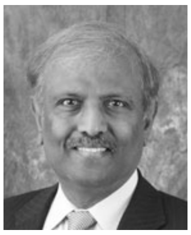
$$\hat{W}^{i+1} = \Phi(X_k)^{-1}\mathrm{RHS}(X_k, \hat{W}^i). \tag{a30}$$

For the inverse $\Phi(X_k)^{-1}$ to exist, $X_k^{(j)}$'s should not be identical, and the elements of vector $\Phi(X_k)$ should be linearly independent. One can select more than the 'm' minimum required sets of states and formulate a recursive relationship with the overdefined system of equations. In fact, it is advisable to use a large number of datasets since it will better enable the network to capture the long-term behavior or the system evolution for many different initial conditions and may help with faster convergence. In this case, the unique solution of the least-squares minimization problem is simply

$$\hat{W}^{i+1} = (\Phi(X_k)^{\mathrm{T}}\Phi(X_k))^{-1}\Phi(X_k)^{\mathrm{T}}\mathrm{RHS}(X_k, \hat{W}^i). \tag{a31}$$



Jie DING is currently working toward the Ph.D. degree at the Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology. E-mail: jdr5c@mail.mst.edu.



S. N. BALAKRISHNAN received his Ph.D. degree from the University of Texas, Austin. He is currently a curators' professor of Aerospace Engineering, Missouri University of Science and Technology. His research interests include neural networks, optimal control, and large-scale and impulse systems. His papers from the development of techniques in these areas include applications to missiles, spacecraft, aircraft, robotics, temperature, and animal population control. E-mail: bala@mst.edu.