This article is published in *Applied Soft Computing* (2014). The presented contents are final as published in the journal. The source code for the numerical analyses can be found at [33].

Global Optimality of Approximate Dynamic Programming and Its Use in Non-convex Function Minimization

Ali Heydari[1] and S. N. Balakrishnan[2]

Abstract: This study investigates the global optimality of approximate dynamic programming (ADP) based solutions using neural networks for optimal control problems with fixed final time. Issues including whether or not the cost function terms and the system dynamics need to be convex functions with respect to their respective inputs are discussed and sufficient conditions for global optimality of the result are derived. Next, a new idea is presented to use ADP with neural networks for optimization of non-convex smooth functions. It is shown that any initial guess leads to direct movement toward the proximity of the global optimum of the function. This behavior is in contrast with gradient based optimization methods in which the movement is guided by the shape of the local level curves. Illustrative examples are provided with single and multi-variable functions that demonstrate the potential of the proposed method.

1. Introduction

In the last two decades, approximate dynamic programming (ADP) has been shown to have a great promise in solving optimal control problems with neural networks (NN) [1-15]. In the ADP framework, the solutions are obtained using a two-network synthesis called adaptive critics (ACs) [2-4]. In the heuristic dynamic programming (HDP) approach with ACs, one network, called the 'critic' network, maps the input states to output the cost-to-go and another network, called the 'action' network, outputs the control with states of the system as its inputs [4,5]. In the dual heuristic programming (DHP) formulation, the action network remains the same as in the HDP, however, the critic network outputs the costates with the current states as inputs [2,6,7]. The computationally effective single network adaptive critics (SNAC) architecture consists of one network only. In [8], the action network was eliminated in a DHP type formulation with control being calculated from the costate values. Similarly, the J-SNAC [9] eliminates the need for the action network in an HDP scheme. Note that the developments in [1-9] are for *infinite-horizon* problems.

The use of ADP for solving *finite-horizon* optimal control problems was considered in [10-15]. Authors of [10] developed a time-varying neurocontroller for solving a scalar problem with state constraints. In [11] a single NN with a single set of weights was proposed which takes the time-to-go as an *input* along with the states and generates the fixed-final-time optimal control for discrete-time nonlinear multi-variable systems. An HDP based scheme for optimal control problems with *soft* and *hard* terminal constraints was presented in [12]. Finite-horizon problems with *unspecified* terminal times were considered in [13-15]. For an extensive literature on adaptive critic based problems, the reader is referred to [16] and the references in the chapters.

Despite much published literature on adaptive critics, there still exists an open question about the nature of optimality of the adaptive critic based results. Are they locally or globally optimal? A major contribution of this study is in proving that the AC based solutions are globally optimal subject to the assumed basis functions. To help with the development of the proof, the ADP based algorithm for solving fixed-final-time problems developed in [11,12] is revisited first. After describing the algorithm, a novel analysis of global optimality of the result is presented. It is shown that with any cost function with a quadratic control penalizing term, the resulting cost-to-go function (sometimes called value function) will be convex versus the control at the current time if the sampling time used for discretization of the original continuous-time system is small enough, and hence, the first order necessary optimality condition [17] will lead to the *global* optimal control. The second major contribution of this paper is in showing that the ADP can be used for function optimization, specifically, optimization of non-convex functions. Finally, through numerical simulations, two examples with varying complexities are presented and the performance of the proposed method is investigated. It is shown that despite the gradient based methods, selecting any initial guess on the minimum and updating the guess using the control resulting from the actor, the states will move *directly* toward the global minimum, passing any possible local minimum in the path.

The rest of this paper is organized as follows: The problem formulation is given in section 2. The ADP-based solution is discussed in section 3. The supporting theorems and analyses are presented in section 4. The use of the method in static function optimization is discussed in section 5, and the conclusions are given in section 6.

2. Problem Formulation

Let the control-affine dynamics of the system be given by

$$\dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)u(t) \tag{1}$$

This work was partially supported by a grant from the National Science Foundation.

[1] Mechanical Engineering Department, South Dakota School of Mines and Technology, (corresponding author, e-mail: ali.heydari@sdsmt.edu).

[2] Mechanical and Aerospace Engineering Department, Missouri University of Science and Technology, (e-mail: bala@mst.edu).

where $f: \mathbb{R}^n \to \mathbb{R}^n$, and $g: \mathbb{R}^n \to \mathbb{R}^{n\times m}$. The state and control vectors are denoted with $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$, respectively, where positive integers n and m denote the dimensions of the respective vectors. The selected cost function, J is fairly general but quadratic in control:

$$J = \psi\left(x(t_f)\right) + \int_{t_0}^{t_f}\left(Q\big(x(t)\big) + u(t)^T R u(t)\right)dt, \tag{2}$$

where positive semi-definite smooth functions $Q: \mathbb{R}^n \to \mathbb{R}$ and $\psi: \mathbb{R}^n \to \mathbb{R}$ penalize the states and positive definite matrix R penalizes the control effort. The initial and final time are denoted with t_0 and t_f, respectively. Discretizing the time horizon to N time steps using sampling time Δt leads to the discrete-time dynamics and cost function as

$$x_{k+1} = \bar{f}(x_k) + \bar{g}(x_k)u_k,\ k \in K, \tag{3}$$

$$J = \psi(x_N) + \textstyle\sum_{k=0}^{N-1}(\bar{Q}(x_k) + u_k^T \bar{R} u_k), \tag{4}$$

where $K \equiv \{0, 1, 2, \dots, N-1\}$, $N \equiv \left(t_f - t_0\right)/\Delta t$, $x_k \equiv x(k\Delta t + t_0)$, $\bar{f}(x) \equiv x + \Delta t f(x)$, $\bar{g}(x) \equiv \Delta t g(x)$, $\bar{Q}(x) \equiv \Delta t Q(x)$, and $\bar{R} \equiv \Delta t R$, if Euler integration is used. The problem is defined as finding a control history $u_k \in \mathbb{R}^m$, $\forall k \in K$, such that cost function (4) is minimized subject to the dynamics given in (3).

Assumption 1: The dynamics of the system do not have finite escape times. Also, the functions $f(x)$ and $g(x)$ are smooth in x.

Remark 1: In order to use ADP, the continuous-time problem is discretized. Moreover, the assumption that discrete-time system (3) is obtained through discretizing a continuous-time problem is utilized in convergence analysis of the algorithm.

3. Approximate Dynamic Programming Based Solution

In this section, an ADP scheme called AC is used for solving the fixed-final-time optimal control problem in terms of the network weights and selected basis functions. The method is adopted from [11,12]. In this scheme, two networks called critic and actor are trained to approximate the optimal cost-to-go and the optimal control, respectively. It should be noted that the optimal cost-to-go, which represents the incurred cost if optimal decisions are made from the current time to the final time, at each instant is a function of the current state, x_k, and the current time, k. Denoting the cost-to-go with $J_k^*(x_k)$, one has

$$J_k^*(x_k) = \psi(x_N) + \textstyle\sum_{\bar{k}=k}^{N-1}\big(\bar{Q}(x_{\bar{k}}) + u_{\bar{k}}^{*T} \bar{R} u_{\bar{k}}^*\big), \tag{5}$$

where u_k^* denotes the optimal control at time k given the current state x_k. The solution to the problem is given by the Bellman equation [18] as

$$J_N^*(x_N) = \psi(x_N), \tag{6}$$

$$J_k^*(x_k) = \bar{Q}(x_k) + u_k^{*T} \bar{R} u_k^* + J_{k+1}^*(x_{k+1}^*),\ k \in K \tag{7}$$

$$u_k^* = \text{argmin}_{u_k \in \mathbb{R}^m}\left(\bar{Q}(x_k) + u_k^T \bar{R} u_k + J_{k+1}^*\big(\bar{f}(x_k) + \bar{g}(x_k)u_k\big)\right),\ k \in K \tag{8}$$

where $x_{k+1}^* = \bar{f}(x_k) + \bar{g}(x_k)u_k^*$. Applying the first order optimality condition [17,4], Eq. (8) leads to

$$u_k^* = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T\ \nabla J_{k+1}^*(x_{k+1}^*),\ k \in K \tag{9}$$

where $\nabla J_{k+1}^*(x_{k+1}^*)$ denotes gradient $\partial J_{k+1}^*(x_{k+1})/\partial x_{k+1}$ evaluated at x_{k+1}^* and formed as a column vector.

An iterative learning scheme can be derived from Bellman equation for finding the solution to the fixed-final-time problem once Eq. (9) is replaced with [12]

$$u_k^{i+1} = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k^i),\ k \in K. \tag{10}$$

Superscript i denotes the index of iteration which starts with an initial guess on u_k^0, $k \in K$. The converged value of u_k^i in (10) is denoted with u_k^* and used in (7). Note that in the dual network AC scheme for finite horizon optimal control, 'iterations' takes place in finding the optimal control, as seen in (10). Once state-control relationship is learned, the optimal cost-to-go is obtained in a 'one-shot' process as given in (7).

Denoting the approximated optimal cost-to-go and the approximated optimal control in a feedback form with $J_k(x_k)$ and $u_k(x_k)$, respectively, and selecting linear in the weights NNs, the expressions for the actor (control) and the critic (cost), can be written as

$$u_k(x) = V_k^T \sigma(x),\ k \in K \tag{11}$$

$$J_k(x) = W_k^T \phi(x),\ k \in K \cup \{N\} \tag{12}$$

where $V_k \in \mathbb{R}^{p\times m}$ and $W_k \in \mathbb{R}^q$ are the unknown weights of the actor and the critic networks at time step k, respectively, and the selected smooth basis functions are given by $\sigma: \mathbb{R}^n \to \mathbb{R}^p$ and $\phi: \mathbb{R}^n \to \mathbb{R}^q$ where p and q denote the number of neurons. The idea is using Eqs. (6), (7), and (10) to find the NN weights. Note that, once $J_{k+1}^*(.)$ is known one can use (10) to find $u_k^*(.)$ and then (7) gives $J_k^*(.)$. Therefore, starting with (6) to find $J_N^*(.)$, all the unknowns can be calculated in a backward in time fashion, i.e., from $k = N-1$ to $k = 0$. The learning process for calculating weights V_k and W_k, $\forall k$, is detailed through Algorithm 1. Note that $\nabla\phi(x) \equiv \partial\phi(x)/\partial x$ is formed as a column vector.

Algorithm 1

Step 1: Randomly select n state vectors $x_N^{[j]} \in \Omega, \forall j \in \mathcal{J} \equiv \{1,2,..,n\}$, where n is a selected large positive integer, and Ω denotes a compact subset of $\mathbb{R}^n$ representing the domain of interest.

Step 2: Find W_N such that $W_N^T \phi\left(x_N^{[j]}\right) \cong \psi(x_N^{[j]}), \forall j \in \mathcal{J}$.

Step 3: For $k = N-1$ to $k = 0$ repeat Steps 4 to 9.

{

Step 4: Set $i = 0$ and select a guess on $u_k^{0,[j]} \in \mathbb{R}^m, \forall j \in \mathcal{J}$.

Step 5: Randomly select n state vectors $x_k^{[j]} \in \Omega, \forall j \in \mathcal{J}$.

Step 6: Set $u_k^{i+1,[j]} = -\frac{1}{2}\bar{R}^{-1}\bar{g}\left(x_k^{[j]}\right)^T \nabla\phi\left(\bar{f}\left(x_k^{[j]}\right) + \bar{g}\left(x_k^{[j]}\right)u_k^{i,[j]}\right)^T W_{k+1}, \forall j \in \mathcal{J}$.

Step 7: Set $i = i+1$ and repeat Step 6, until $\left\|u_k^{i+1,[j]} - u_k^{i,[j]}\right\| < \beta, \forall j \in \mathcal{J}$, where $\beta > 0$ is a preset tolerance. Denote the converged value of $u_k^{i,[j]}$ with $u_k^{[j]}, \forall j$.

Step 8: Find V_k such that $V_k^T \sigma\left(x_k^{[j]}\right) \cong u_k^{[j]}, \forall j \in \mathcal{J}$.

Step 9: Find W_k such that $W_k^T \phi\left(x_k^{[j]}\right) \cong \bar{Q}\left(x_k^{[j]}\right) + {u_k^{[j]}}^T \bar{R} u_k^{[j]} + W_{k+1}^T \phi\left(\bar{f}\left(x_k^{[j]}\right) + \bar{g}\left(x_k^{[j]}\right)u_k^{[j]}\right), \forall j \in \mathcal{J}$.

}

In Steps 2, 8, and 9 of Algorithm 1, the method of Least Squares, explained in the Appendix, can be used for finding the unknown weights.

Remark 2: The capability of *uniform approximation* of neural networks [19,20] indicates that once the network is trained for a large enough number of samples distributed evenly throughout the domain of interest, the network is able to approximate the output for any new sample of the domain with a bounded approximation error. This error bound can be made arbitrarily small once the network is rich enough. For the linear-in-weight neural networks selected in this study and the polynomial basis function utilized in the numerical examples, Weierstrass approximation theorem [21] proves a similar uniform approximation capability.

Remark 3: The ADP based solution presented in this section provides (approximate) optimal solution for different initial conditions, as long as the resulting state trajectory lies within the domain on which the networks are trained. Therefore, no retraining is needed every time that a different initial condition is selected for control.

4. Supporting Theorems and Analyses

A. Convergence Analysis

Theorem 1: The iterative relation given by Eq. (10), with any initial guess for $u_k^0 \in \mathbb{R}^m, \forall k \in K$, converges, providing the sampling time Δt selected for discretization of continuous-time dynamics (1) is small enough.

Proof: Let the right hand side of (10) be denoted with function $\mathcal{F}: \mathbb{R}^m \to \mathbb{R}^m$ where

$$\mathcal{F}(u) = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u) \tag{13}$$

The proof is complete if it is shown that the relation given by the successive approximation

$$u^{i+1} = \mathcal{F}(u^i) \tag{14}$$

is a contraction mapping [22]. Since $\mathbb{R}^m$ with 2-norm denoted with $\|.\|$ is a Banach space, the iterations given by (14) converges to some $u_k = \mathcal{F}(u_k)$ if there is a $0 \le \rho < 1$ such that for every u and v in $\mathbb{R}^m$, the following inequality holds

$$\|\mathcal{F}(u) - \mathcal{F}(v)\| \le \rho\|u - v\|. \tag{15}$$

By Eq. (13) one has

$$\|\mathcal{F}(u) - \mathcal{F}(v)\| \le \left\|\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*\left(\bar{f}(x_k) + \bar{g}(x_k)u\right) - \frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*\left(\bar{f}(x_k) + \bar{g}(x_k)v\right)\right\| \tag{16}$$

The optimal cost-to-go function, $J_{k+1}^*(x_{k+1})$, is smooth versus its input, x_{k+1} since functions $\phi(.)$, $Q(.)$, $f(.)$, and $g(.)$ are smooth. By considering Eq. (6), since $\phi(x_N)$ is a smooth function, function $J_N^*(x_N)$, and hence $\nabla J_N^*(x_N)$, are smooth. Smoothness of $\bar{g}(x_{N-1})$ and $\nabla J_N^*(x_N)$ leads to the smoothness of optimal control function $u_{N-1}^*(x_{N-1})$, by (9), which along with the smoothness of $\bar{Q}(x_{N-1})$ lead to a smooth $J_{N-1}^*(x_{N-1})$, by (7). Repeating this argument backward from $N-1$ to $k+1$, it follows that $J_{k+1}^*(x_{k+1})$ is a smooth function. This smoothness leads to the Lipschitz continuity of $\nabla J_{k+1}^*(x_{k+1})$ in the domain of interest

Ω [23]. In other words, there exists some positive real number ρ_J such that for every x_1 and x_2 in Ω, one has $\|\nabla J^*_{k+1}(x_1) - \nabla J^*_{k+1}(x_2)\| \leq \rho_J \|x_1 - x_2\|$. Using this characteristic, inequality (16) can be written as

$$\|\mathcal{F}(u) - \mathcal{F}(v)\| \leq \rho_J \|\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T\| \|\bar{g}(x_k)\| \|(u - v)\| \tag{17}$$

By defining

$$\rho \equiv \rho_J \|\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T\| \|\bar{g}(x_k)\|$$

which is equivalent of

$$\rho \equiv \rho_J \|\frac{1}{2}R^{-1}g(x_k)^T\| \|\Delta t g(x_k)\| \tag{18}$$

one can select the sampling time Δt in discretization of the continuous-time dynamics (1) small enough such that the condition $0 \leq \rho < 1$ is satisfied. Note that the continuity of $g(.)$ in its domain result in being bounded in compact set Ω [24], hence, the state-dependent terms in (18) are upper bounded. This completes the proof of existence of a fixed point, denoted with u_k, and the convergence of u_k^i to u_k, as $i \to \infty$, $\forall u_k^0 \in \mathbb{R}^m$ and $\forall k \in K$, using successive approximation given by (10). ■

In Theorem 1 the role of the sampling time in discretization of a continuous-time system is emphasized. It is worthwhile to discuss this issue in detail. Substituting (12) in optimal control equation (9) and evaluating it at different $x_k^{[j]}$s lead to

$$u_k^{[j]} = -\frac{1}{2}R^{-1}g\left(x_k^{[j]}\right)^T \nabla\phi\left(f\left(x_k^{[j]}\right) + g\left(x_k^{[j]}\right)u_k^{[j]}\right)^T W_{k+1}, \forall j \in \mathcal{J}, \tag{19}$$

which is the same as the iterative equation given in Step 6 of Algorithm 1 except that $u_k^{i,[j]}$ and $u_k^{i+1,[j]}$ on both sides are replaced with the converged value, i.e., $u_k^{[j]}$. Optimal control $u_k^{[j]}$, $\forall k \in K$ and $\forall j \in \mathcal{J}$, can be calculated by solving the set of m nonlinear equations given by (19) for the m unknown elements of $u_k^{[j]}$, without using the iteration given by (10) which is given in terms of the critic network in Steps 6 of Algorithm 1. However, there is no analytical solution to the set of nonlinear equations (19) in general. Therefore, one needs to resort to numerical methods for solving the set of equations. Theorem 1 proves that for any given smooth dynamics and smooth basis functions, if the sampling time is small enough, the iterations given by (10) converge to the solution to the nonlinear equation (19). However, if the sampling time is fixed, certain conditions on the dynamics or the cost function terms need to hold in order for the iterations to converge. These conditions can be easily derived from the proof of Theorem 1, i.e., from Eq. (18).

In the solution to linear problems with quadratic cost function terms, a similar issue is observed. To see this, one may consider the optimal control equation (9). Let cost-to-go function, for the linear problem, be $J_k = \frac{1}{2}x_k^T P_k x_k$ for some $P_k \in \mathbb{R}^{n\times n}$. Consequently, the optimal control equation (9) becomes

$$u_k = -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}x_{k+1} = -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}(\bar{A}x_k + \bar{B}u_k), \tag{20}$$

where the continuous problem given by $\dot{x}(t) = Ax(t) + Bu(t)$ and $J = x\left(t_f\right)^T Sx\left(t_f\right) + \int_{t_0}^{t_f}\left(x(t)^T Qx(t) + u(t)^T Ru(t)\right)dt$ is discretized to $x_{k+1} = \bar{A}x_k + \bar{B}u_k$ and $J = x_N^T S x_N + \sum_{k=0}^{N-1}(x_k^T \bar{Q} x_k + u_k^T \bar{R} u_k)$. Similar to (19), unknown u_k exists in both sides of equation (20). However, equation (20) is linear and the analytical solution can be calculated as

$$u_k = -(2\bar{R} + \bar{B}^T P_{k+1}\bar{B})^{-1}\bar{B}^T P_{k+1}\bar{A}x_k. \tag{21}$$

If solution (21) was not available, one could use the following iterations, starting with any initial guess u_k^0, to find u_k

$$u_k^{i+1} = -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}(\bar{A}x_k + \bar{B}u_k^i)\ . \tag{22}$$

Following the idea presented in proof of Theorem 1, it is straightforward to show that (22) is a contraction mapping if the sampling time used for discretization of the original continuous-time linear problem is small enough. Hence u_k^i converges to the solution of (20). Another way of investigating the effect of the sampling time in the convergence of Eq. (22) is considering the evolution of u_k^i during the *iterations* as the evolution of the state vector of a discrete-time system versus *time*. In other words, one can look at Eq. (22) as a discrete-time system with the state vector at 'time' i being denotes with u_k^i and the constant control to the system being x_k. Then Eq. (22) for every given k can be written as

$$u_k^{i+1} = \mathcal{A}u_k^i + \mathcal{B}x_k \tag{23}$$

where the superscripts denote the 'time' index and

$$\mathcal{A} \equiv -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}\bar{B} = -\frac{1}{2}\Delta t R^{-1}B^T P_{k+1}B \tag{24}$$

and

$$\mathcal{B} \equiv -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}\bar{A}.$$

Selecting any bounded u_k^0, the system given in (23) is stable for a bounded x_k and converges to a steady state u_k as $i \to \infty$, providing the eigenvalues of $\mathcal{A}$ are inside the unit circle around the origin [25]. Considering (24), it is straight forward to show that for any given R, B, A, Q, and S, (where the last three matrices affect the equation through P_{k+1}), selecting a small enough sampling time Δt, the eigenvalues of $\mathcal{A}$ can be made arbitrarily small and brought into the unit circle and hence, the system can be made stable. This stability leads to the convergence of iterations given by (22).

B. Global Optimality Analysis

The objective is finding the 'global' optimal control, not a local optimum. Let the cost-to-go given state x_k, time k, and utilizing *any* control u_k at the current time and the *optimal* control for the rest of the time steps be denoted with $J_k^*(x_k, u_k)$. In other words,

$$J_k^*(x_k, u_k) = \bar{Q}(x_k) + u_k^T \bar{R} u_k + J_{k+1}^*\big(\bar{f}(x_k) + \bar{g}(x_k)u_k\big). \tag{25}$$

This cost-to-go should be differentiated from the 'optimal' cost-to-go, which is only a function of time and current state, as given in Eq. (5). In other words, $J_k^*(x_k) = \min_{u_k \in \mathbb{R}^m}\big(J_k^*(x_k, u_k)\big)$. The global optimal control is given by

$$\begin{aligned} u_k^* &= \operatorname{argmin}_{u_k \in \mathbb{R}^m}\big(J_k^*(x_k, u_k)\big) \\ &= \operatorname{argmin}_{u_k \in \mathbb{R}^m}\Big(\bar{Q}(x_k) + u_k^T \bar{R} u_k + J_{k+1}^*\big(\bar{f}(x_k) + \bar{g}(x_k)u_k\big)\Big) \\ &= \operatorname{argmin}_{u_k \in \mathbb{R}^m}\Big(u_k^T \bar{R} u_k + J_{k+1}^*\big(\bar{f}(x_k) + \bar{g}(x_k)u_k\big)\Big) \end{aligned} \tag{26}$$

As shown in the proof of Theorem 1, the optimal cost-to-go function $J_{k+1}^*(x_{k+1})$ is smooth versus x_{k+1}, therefore, the smoothness of $J_k^*(x_k, u_k)$ with respect to u_k follows from (25). If function $J_k^*(x_k, u_k)$ is 'convex' in u_k, then its global optimum could be found by comparing its value at its sole critical point as well as at the boundaries. Smoothness of the function and its unboundedness at the boundaries of $\mathbb{R}^m$ leads to the only candidate being the point at which the gradient vanishes. Therefore, if the function is convex, the global optimum is given by

$$\partial J_k^*(x_k, u_k)/\partial u_k = 0 \implies u_k^* = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(x_{k+1}^*) \tag{27}$$

However, if the function is not convex, Eq. (27) which the Adaptive Critics are based on, can lead to a 'local' minimum. Considering (26), even though the first term subject to minimization is convex in u_k, the second term, i.e., $J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k)$, may not be convex in u_k. In other words, satisfying Eq. (27) is only a 'necessary' condition for global optimality [17]. This fact leads to the problem that there might be more than one u_k which satisfy (27). For example, the iteration given by (10) may converge to some u_k which satisfies (27), but it is not the global optimum.

Considering the result given in Theorem 1, this concern is addressed. Note that, once it is shown that (10) is a contraction mapping in Theorem 1, the 'uniqueness' of the converged value follows [22]. In other words, if (10) is a contraction mapping, there cannot be any other u_k which satisfy (27), except the one which can be found using the iteration given by (10). This result might be unexpected because of claiming the global minimum without assuming convexity of $J_{k+1}^*(x_{k+1}, .)$ or even assuming convexity of cost function terms $Q(.)$ or $\psi(.)$. It should be noted that even if the cost-function terms are convex in their arguments, because of the arbitrary dynamics of the system, the cost-to-go may not be convex in control. Interested reader are referred to Mangasarian Sufficient Condition Theorem [26] which requires the cost function terms as well as the dynamics functions to be convex in x and u, and also requires that the resulting optimal costates being non-negative for the entire horizon, in order to conclude the 'global' optimality of the solution to the optimal control problem. As for ADP, the authors of [27] claim that the ADP is susceptible in getting stuck in local optimums.

Note that we are interested in minimization of $F(u_k) \equiv u_k^T \bar{R} u_k + J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k)$. Theorem 2 proves that, providing a certain condition holds, function $F(u_k)$ is convex. Therefore, Eq. (27) gives the global optimal solution. For this theorem, the following Lemma is required. Before proceeding to the lemma, it should be noted that function $F: \mathbb{R}^m \to \mathbb{R}$ is *convex* in convex space $\bar{\Omega}$ if $F_{uu}(u) \equiv \partial^2 F(u)/\partial u^2 \geq 0$, $\forall u \in \bar{\Omega}$. If $F_{uu}(u) > 0$, $\forall u \in \bar{\Omega}$, then the function is called *strictly convex*. Moreover, if there exists some positive real number c such that $F_{uu}(u) \geq cI$, $\forall u \in \bar{\Omega}$, then the functions is called *strongly convex*, where the $m \times m$ identity matrix is denoted with I, cf. [28], and notations $F_{uu}(u, a) \geq 0$ and $F_{uu}(u, a) > 0$, respectively, mean that $F_{uu}(u, a)$ is positive semi-definite and positive-definite.

Lemma 1: Define $F(u, a) \equiv a f_1(u) + f_2(au, a)$, for twice differentiable functions $f_1: \mathbb{R}^m \to \mathbb{R}$ and $f_2: \mathbb{R}^m \times \mathbb{R}_+ \to \mathbb{R}$ with respect to all of the inputs, where a is a positive real number and $\mathbb{R}_+$ denotes the set of nonnegative reals. Assume $f_1(.)$ is a strongly convex function in a convex set Ω_1 containing the origin. Then, there exists some positive real number a_0 which for any $a \in (0, a_0]$ function $F(x, a)$ is strictly convex with respect to x in any compact and convex set $\Omega_2 \subset \Omega_1$ containing the origin.

Proof: Because of the twice differentiability of $f_1(.)$ and $f_2(., a)$, function $F(., a)$ is also twice differentiable. If there exists some a_0 which for every $a \in (0, a_0]$ one has $F_{uu}(u, a) > 0$ for every u in every convex subset of Ω_1, then $F(., a)$ is strictly convex in that domain and the proof is complete [28]. By definition, one has

$$F_{uu}(u, a) = a f_{1_{uu}}(u) + a^2 f_{2_{uu}}(au, a). \tag{28}$$

The strong convexity of $f_1(.)$ leads to the existence of some positive real number c_1 such that

$$f_{1_{uu}}(u) - c_1 I > 0, \forall u \in \Omega_1. \tag{29}$$

On the other hand, $f_{2_{uu}}(.,.)$ is bounded in every compact and convex set $\Omega_2 \times [0, a_1] \subset \Omega_1 \times \mathbb{R}_+$ containing origin, for every selected $a_1 \in \mathbb{R}_+$. The reason is the continuity of $f_{2_{uu}}(.,.)$ [24], which is resulted from its twice differentiability. Therefore, denoting the smallest eigenvalue with $\lambda(.)$, there exists some positive real number c_2 for the selected a_1 such that

$$\inf_{\substack{u\in\Omega_2 \\ a\in[0,a_1]}} \lambda\left(f_{2_{uu}}(au, a)\right) \geq -c_2. \tag{30}$$

On the other hand, inequality (29) leads to

$$\inf_{u\in\Omega_2} \lambda\left(f_{1_{uu}}(u)\right) > c_1. \tag{31}$$

Considering (28), (30) and (31), there always exists some $a_0 \in (0, a_1]$ which for every $a \in (0, a_0]$ one has $F_{uu}(u, a) > 0$, $\forall u \in \Omega_2$. The reason is, because of the symmetricity of matrices $f_{1_{uu}}$ and $f_{2_{uu}}$, Eq. (28) leads to [29]

$$\lambda(F_{uu}(u, a)) \geq a\lambda\left(f_{1_{uu}}(u)\right) + a^2\lambda\big(f_{2_{uu}}(au, a)\big), \tag{32}$$

for every selected a and u. Inequality (32) along with (30) and (31) lead to

$$\lambda(F_{uu}(u, a)) > ac_1 - a^2c_2, \tag{33}$$

in $\Omega_2 \times [0, a_1]$, i.e., for every $u \in \Omega_2$ and $a \in [0, a_1]$. Therefore, selecting $a_0 = \min(c_1/c_2, a_1)$, one has

$$\lambda\big(F_{uu}(u, a)\big) > 0, \tag{34}$$

in $\Omega_2 \times [0, a_0]$. This shows that $F_{uu}(u, a) > 0$, $\forall u \in \Omega_2$, and $\forall a \in (0, a_0]$, hence, proves the lemma. ■

In order to better understand the result given in Lemma 1, the following example is helpful.

Example 1: Let $f_1(u) = u^2$ and $f_2(u, a) = 2.1333u^4 + 0.9333u^3 - 2.1333u^2 - 0.9333u + a$. Fig. 1 depicts the shapes of functions $f_1(.)$ and $f_2(.,1)$. As seen in the figure, $f_1(.)$ is a convex function, but, $f_2(.,1)$ is non-convex. Let $F(u, a) \equiv af_1(u) + f_2(au, a)$. Fig. 2 presents the shape of the resulting $F(u, a)$ for the three values of $a = 10$, 1, and 0.1. This figure shows that as a becomes smaller, function $F(u, a)$ changes toward becoming a convex function. For example, $F(u, 10)$ is non-convex while $F(u, 0.1)$ is a convex function. This behavior is compatible with the result given in Lemma 1.

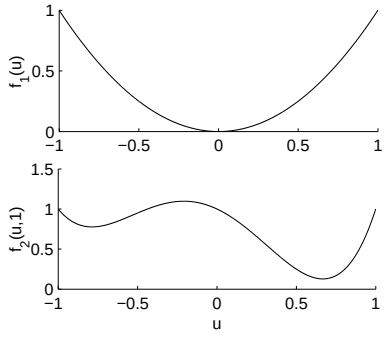

Fig. 1: Functions $f_1(u)$ and $f_2(u, 1)$.

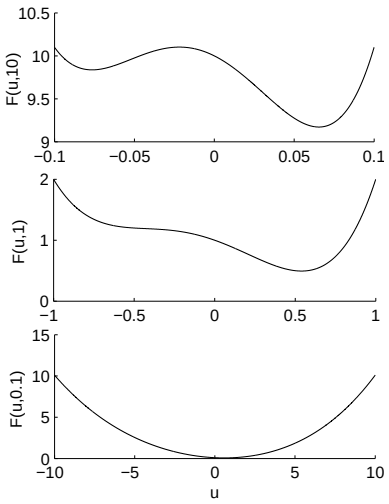

Fig. 2: Function $F(u, a)$ for $a = 10$, 1, and 0.1.

Theorem 2: There exists some sampling time Δt_0 for discretization of the original continuous problem, that selecting any sampling time smaller than that leads to the strict convexity of the cost-to-go function given in (25) with respect to the control in any compact and convex subset of $\mathbb{R}^m$.

Proof: Rewriting $F(u_k) \equiv u_k^T \bar{R} u_k + J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k)$ in terms of the parameters given in the continuous-time problem (before discretization) one has

$$F(u_k, \Delta t) \equiv \Delta t u_k^T R u_k + J_{k+1}^*(x_k + \Delta t f(x_k) + \Delta t g(x_k)u_k). \tag{35}$$

Defining $f_1(u) \equiv u^T R u$ and $f_2(u, a) \equiv J_{k+1}^*(x_k + af(x_k) + g(x_k)u)$, one has

$$F(u_k, \Delta t) \equiv \Delta t f_1(u) + f_2(\Delta t u, \Delta t). \tag{36}$$

The strong convexity and twice differentiability of $f_1(.)$ follows from $f_{1_{uu}}(u) = R > 0$. The smoothness of $J_{k+1}^*(.)$, derived in the proof of Theorem 1, leads to the twice differentiability of $f_2(.,.)$ with respect to both its inputs. Comparing $F(u_k, \Delta t)$ defined in (36) with $F(x, a)$ defined in Lemma 1, Δt has the same role that a has in Lemma 1. Therefore, there exists some small

sampling time, which for any sampling time smaller than that, the cost-to-go is strictly convex with respect to the control in any selected compact and convex subset of $\mathbb{R}^m$. ■

While the focus in this study is on optimal control problems with finite-horizon, it can be seen that the result given in Theorem 2 holds for the use of ADP for infinite horizon problems as well.

In order to better understand the effect of sampling time in making the cost-to-go a convex function with respect to the control, the following example is helpful. In this example, terminal cost $\psi(.)$ is selected as a non-convex function, and the relation between the global minimum of $\psi(.)$ and that of the optimal control problem is investigated.

Example 2: Let the fixed final time cost function with the horizon equal to Δt be given by

$$J = c\psi\big(x(\Delta t)\big) + \int_0^{\Delta t} u(t)^T u(t)\, dt \tag{37}$$

for some positive real number c. The discretized cost function will be

$$J = c\psi(x_1) + \Delta t u_0^T u_0. \tag{38}$$

Assume the dynamics be given by the single integrator $\dot{x} = u$ discretized to $x_1 = x_0 + \Delta t u_0$. For simplicity assume that x_k and u_k are scalar. Let $c\psi(x)$ be the non-convex function plotted in Fig. 3.

Let the selected initial condition be x_0 shown in the figure. As seen, the global minimum of $\psi(.)$ is at x^*, while the selected x_0 is on a downslope toward a local minimum. Denote the cost-to-go of applying no control and staying at x_0 with J_0. Assume that the optimal solution, with corresponding optimal cost-to-go of J^*, leads to moving toward the left, i.e., toward the global minimum of $\psi(.)$. The cost-to-go difference denoted with ΔJ and defined as $J^* - J_0$, needs to be negative, otherwise J^* is not optimal. However, looking at Fig. 3, it can be seen that unless x_1, i.e., the terminal point, is on the left side of $\bar{x}$, ΔJ will not be negative. The reason is, if x_1 is somewhere between x_0 and $\bar{x}$, then we have spent some control cost and have ended up at a place with more terminal cost compared to staying at x_0 and spending no control cost. But, in order to end up at some point on the left side of $\bar{x}$, we need a control which satisfies $|u_0| > a/\Delta t$, because of the selected dynamics, where $a \equiv |x_0 - \bar{x}|$. In this case, the control cost will be $\Delta t u_0^2 > (a/\Delta t)^2 \Delta t$. But, the best thing which can be done in terms of having less terminal cost is reaching point x^*, with the reward equal to b, given in the figure. Therefore, in moving toward left one has

$$\Delta J > (a/\Delta t)^2 \Delta t - b = a^2/\Delta t - b. \tag{39}$$

As $\Delta t \to 0$, regardless of how deep the global optimum is, i.e., how large parameter c in the cost function is, which results in large b, there always exists some Δt which for any sampling time smaller than that, one has $\Delta J > 0$. Therefore, the global optimal solution to the cost function is not toward the left of x_0.

The global optimal solution is moving toward the right side of x_0. The reason is, $\partial\psi(x_0)/\partial x_0 = -d < 0$, for some positive d. By definition, if x_1 is selected close to x_0, one has $\partial\psi(x_0)/\partial x_0 \cong -\bar{b}/\bar{a}$. Therefore, $\bar{b} \cong d\bar{a}$. Denoting the cost-to-go of moving toward the right by $\bar{J}^*$, the cost difference $\Delta J \equiv \bar{J}^* - J_0$ has to be negative in order to move toward right, for example to the point denoted with x_1. The control for this move is $\bar{a}/\Delta t$, and the control cost is $(\bar{a}/\Delta t)^2 \Delta t = \bar{a}^2/\Delta t$. Hence, $\Delta J = \bar{a}^2/\Delta t - \bar{b}$. Considering $\bar{b} \cong d\bar{a}$, one has

$$\Delta J = \bar{a}^2/\Delta t - d\bar{a}. \tag{40}$$

Looking at (40), regardless of how small Δt is, there always exists some small $\bar{a}$ for which one has $\Delta J < 0$. Such $\bar{a}$ is given by $\bar{a} < \Delta t d$. Therefore, the global minimum of the *optimal control problem* is moving toward the right of point x_0, despite the fact that the global minimum of the *terminal cost term* is on the left of x_0.

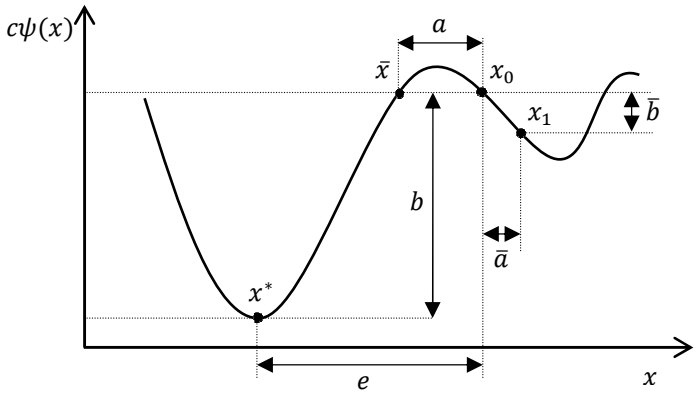

Fig. 3: Non-convex function $c\psi(x)$ versus x

It should be noted that in here, we limited the horizon to be only one time-step. Assuming a constant sampling time Δt and increasing the number of time-steps N, the results change toward coinciding the global optimum of the optimal control problem

with that of the non-convex $\psi(x)$. To see this, let the sampling time be small but, constant and the number of time-steps N be the design parameter. The cost function will be

$$J = c\psi(x_N) + \Delta t \sum_{k=0}^{N-1} u_k^T u_k \tag{41}$$

Considering the distance between x_0 and x^*, one policy to get to x^* from x_0 is applying constant control $\bar{u} = -e/(N\Delta t)$ for N time steps to have $x_N = x^*$, based on the selected dynamics. The control cost for such $\bar{u}$ will be

$$\Delta t \sum_{k=0}^{N-1} \bar{u}_k^T \bar{u}_k = e^2/(N\Delta t) \tag{42}$$

Therefore, denoting the cost-to-go of going from x_0 to x^* in N times steps using constant control $\bar{u}$ with J^*, one has

$$\Delta J \equiv J^* - J_0 = e^2/(N\Delta t) - b. \tag{43}$$

Considering Eq. (43), for any selected Δt, there always exists some large enough N for which, ΔJ is negative. As $N \to \infty$, one has $\Delta J \to -b$. Note that going to any other $x_N \neq x^*$, denoting $\psi(x_N) - \psi(x_0)$ with h, regardless of what constant or non-constant control is being selected, the cost-to-go difference (the cost-to-go of going to x_N minus J_0) will be, at the best case scenario, equal to $-h$. Considering the fact that regardless of how small or large $c > 0$ is, one has $b > h$, the 'global' optimal solution to the optimal control problem will be having $x_N = x^*$, as $N \to \infty$. This result confirms that the global optimum of the optimal control problem coincides with the global optimum of the terminal cost term $\psi(.)$ once the number of time steps goes to infinity.

The important fact which leads to this result, is the control cost being in a quadratic form, therefore, we can always select a larger number of time-steps in order to split the control and end up with less control cost. Note that a control equal to $\bar{u}$ leading to the control cost $\Delta t\bar{u}^2$, once split to N equal parts will have the *total* control cost of $N\Delta t(\bar{u}/N)^2 = \Delta t\bar{u}^2/N$. Therefore, for a fixed Δt, as $N \to \infty$ the control cost vanishes.

Finally, it should be noted that for the NNs trained based on Algorithm 1 to provide the global optimal solution, beside the global optimality of the input-target training pairs, another condition is also required. The condition is that the network training law needs to avoid getting stuck in local minimums, in learning the mapping between the given input-target pairs. One way of fulfilling this requirement is using linear in the weight NNs, as in this study, and using the method of Least Squares (explained in the Appendix) for finding the weight. Note that lease squares problems are convex [28], hence, their solution is the global optimum.

5. Non-convex Function Optimization

One of the applications of the global optimality results given in this study is using ADP for finding the global optimum of smooth but possibly non-convex functions. In other words, the 'optimal control' tool can be used for 'convex or non-convex function optimization'. Considering nonlinear programming based optimization methods [17,18], for optimizing function $\psi(x)$, one selects an initial guess, denoted with x_0, and uses the update rule

$$x_{k+1} = x_k + \tau u_k, \tag{44}$$

where $\tau \in \mathbb{R}\backslash\{0\}$ is the update rate. Parameter u_k is calculated based on the gradient/Hessian of the function subject to minimization at point x_k in gradient based methods of optimization. Update rule (44) should be repeated until the minimum is reached. It is well-known that nonlinear programming based methods are susceptible to getting stuck in local minima. Therefore, they are suitable for optimization of smooth convex functions, not for general smooth functions. An alternative is using evolutionary algorithms [30] in which finding the global minimum depends on the selected initial community/samples.

Looking at (44) it resembles the discretized version of single integrator dynamics $\dot{x} = u$, which is

$$x_{k+1} = x_k + \Delta t u_k. \tag{45}$$

Selecting cost function

$$J = c\psi\left(x(t_f)\right) + \frac{1}{2}\int_{t_0}^{t_f} u(t)^T u(t)\, dt, \tag{46}$$

which is discretized to

$$J = c\psi(x_N) + \frac{\Delta t}{2}\sum_{k=0}^{N-1} u_k^T u_k, \tag{47}$$

minimizing the cost function for large c results in approximately optimizing $\psi(.)$. In other words, solving the optimal control problem defined by cost function (47) subject to dynamics (45) for large $c > 0$ results in the (approximate) global minimum of function $\psi(.)$, given by the state at the final time, i.e., x_N. Therefore, the ADP method described earlier may be used for minimization of non-convex smooth functions.

Remark 4: Even-though it is more intuitive to assume the terminal state penalizing term in the cost function being positive semi-definite in the ADP problems, they are not required to be so for the ADP to provide the optimal solution. Hence, in the development in this section, no assumption on positive semi-definiteness of function $\psi(.)$ is made.

The simplicity of the dynamics given in (45) and the lack of presence of state penalizing term $Q(x)$ in cost function (47) provide an interesting feature for the optimal control problem. Defining the costate vector $\lambda_k \equiv \nabla J_k^*(x_k)$, the costate equation,

resulting from taking the partial derivative of the cost-to-go recursive equation given by (6) and (7) with respect to x, is given by [11]

$$\lambda_N = \partial\psi(x_N)/\partial x_N, \tag{48}$$

$$\lambda_k = \partial\bar{Q}(x_k)/\partial x_k + A_k^T\lambda_{k+1},\ k \in K, \tag{49}$$

where $A_k \equiv \partial x_{k+1}/\partial x_k$. The optimal control will then be given by

$$u_k^* = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T\lambda_{k+1},\ k \in K. \tag{50}$$

Considering dynamics (45) and cost function (47) for the problem at hand, the costate equation simplifies to

$$\lambda_k = \lambda = \partial\psi(x_N)/\partial x_N, \forall k \in K \cup \{N\}, \tag{51}$$

in other words, the costate vector will be a constant vector. Therefore, once the optimal costate vector is found, the optimal control also will be constant and given by $u_k^* = u^* = -\lambda$. This results in a considerable computational simplicity, because, the approximate global minimum x_N, can then be simply calculated as

$$x_N = x_0 - N\Delta t\lambda. \tag{52}$$

It should be noted that the fact that the resulting costate history is constant throughout the horizon does not imply that the costate vector is not dependent on the current states or the current time, which both vary along the horizon. Denoting the costate vector with $\lambda_k(x_k)$, the effect of the change in x_k could be cancelled with the change in k leading to a constant $\lambda_k(x_k)$ along the horizon. Considering the continuous-time problem, whose costate vector may be denoted with $\lambda(x,t)$, this concept can be explained better. Note that

$$\dot{\lambda}(x,t) = \frac{\partial\lambda(x,t)}{\partial x}\dot{x} + \frac{\partial\lambda(x,t)}{\partial t}. \tag{53}$$

Therefore, having $\dot{\lambda}(x,t) = 0$ does not necessarily lead to each one of the terms in the right hand side of (53) being zero separately.

At the global minimum of a smooth function whose minimum is not at the boundaries of its domain, the gradient has to be zero. Therefore, if the global minimum is given by x_N, it is needed to have $\partial\psi(x_N)/\partial x_N = 0$. Considering (51), this condition leads to $\lambda = 0$, and therefore $u^* = 0$ and $x_N = x_0$, which is of course not the desired solution. Solving the defined optimal control problem will not give the solution of $u^* = 0$, because it obviously does not optimize the selected cost function, unless the selected initial condition x_0 coincides with the global optimum of the cost function. In general, the optimal control solution gives a non-zero control, which means that $\lambda = \partial\psi(x_N)/\partial x_N \neq 0$. Therefore, this method will not give the exact global optimum of $\psi(.)$, however, it will provide an approximation of the global optimum. The reason is, looking at (52) parameter N is a design parameter. Selecting a fixed and finite initial condition x_0, the method will provide a finite x_N, otherwise the solution is not the optimal solution to the control problem. However, N can be selected arbitrarily large. Therefore, the following property holds

$$\lim_{N\to\infty}\lambda = 0. \tag{54}$$

In other words, as the number of time steps, denoted with N grows for a given sampling time Δt, the optimal costate vector λ has to converge to zero, in order to cancel the effect of the growth of N and to lead to a finite x_N. The global optimality of the ADP result along with the property given in (54) show that if the number of time steps goes to infinity, x_N will converge to a point which has two features: 1) it globally optimizes the selected cost function, 2) function $\psi(.)$ at this point has the slope of zero, that is $\partial\psi(x)/\partial x = 0$. Selecting large $c > 0$, the only point which has these two features is the global optimum of $\psi(.)$.

Once the optimal control problem is solved, looking at the value of λ provides us with an insight into 'how optimal the result will be'. As discussed above, as λ goes to zero, the result will be closer to the global minimum of $\psi(.)$. It should be noted that this method is not limited to the case of one-dimensional x, and can be used for $x \in \mathbb{R}^n$, i.e., optimizing multi-variable functions. However, it is applicable to the problems which the global optimum is finite and an estimate of the domain of interest, containing the global optimum exists, in order to be used in Algorithm 1 for training the networks.

Further analyzing the property of constant costate vector, given in Eq. (51), provides some interesting result given in Theorem 3.

Theorem 3: The optimal cost-to-go function $J_k^*(x)$, $\forall k \in K$, resulting from minimizing cost function (47) subject to dynamics (45) is a strongly convex function with respect to x in the domain of interest. More specifically, $J_k^*(x)$ is a paraboloid with a unique minimum.

Proof: Considering (52) and the fact that $\lambda = \nabla J_k^*(x)$, $\forall k$, one has

$$\nabla J_k^*(x) = (x - x_N)/((N-k)\Delta t)\,, \forall k \in K, \forall x \in \Omega. \tag{55}$$

Calculating the gradient of (55) leads to the Hessian of $J_k^*(.)$

$$J_{k_{xx}}^*(x) = I/((N-k)\Delta t) > 0, \forall k \in K, \forall x \in \Omega. \tag{56}$$

The abovementioned relation proves that $J_k^*(.)$ is strongly convex in Ω [28]. Eq. (55) provides the shape of function $J_k^*(.)$. It says that function $J_k^*(.)$ is such that its gradient at each point x is proportional to the distance between x and x_N. Also, it says that the function has a unique point with zero gradient located at x_N. Such a function is a paraboloid centered at x_N. It can also be

observed by integrating Eq. (55). From the convexity of $J_k^*(x)$ it follows that the paraboloid opens upward, i.e., point x_N is its 'minimum'. ■

Note that the convexity result given in Theorem 3, for the problem defined in this section, does not assume any condition on the size of the selected sampling time, despite the result given in Theorem 2. However, in order to use the ADP based algorithm discussed in this study for solving the problem, the condition of small enough sampling time is required to guarantee the convergence of the algorithm (Theorem 1).

6. Numerical Analysis

In order to numerically analyze the global optimality of ADP scheme and its use in static minimization, a simple way is selecting a cost function with a non-convex terminal cost term and evaluating the performance of the ADP in providing the global optimum. To this end, two separate examples are selected; a single variable example and a multi-variable benchmark example, namely Rosenbrock/Banana function. The source code for the numerical analysis can be found at [33].

A. Optimizing a Single Variable Function

The function subject to minimization is a single variable non-convex function which has a local minimum as well as a global minimum. The function is

$$\psi(x) = 2.1333x^4 + 0.9333x^3 - 2.1333x^2 - 0.9333x + 1.$$

The function is plotted in Fig. 4. The global minimum is at $x = 0.67$, while the local minimum is at $x = -0.79$. The cost function given in Eq. (46) and discretized to (47) is selected with $c = 100$, in order to put a heavy weight on minimization of $\psi(.)$ as opposed to the control cost. The selected dynamics is of form (45) and the sampling time is selected as $\Delta t = 0.0001$, where $t_0 = 0$ and $t_f = 1\ s$, therefore, $N = 10^4$.

The basis functions are selected as polynomials given below

$$\phi(x) = [1, x, x^2, \dots, x^7],\ \sigma(x) = [1, x, x^2, \dots, x^6].$$

The least squares, as explained in the Appendix, is carried out over 100 points from the domain of interest selected as $-2 \le x \le 2$. Fig. 5 shows the history of the converged weights of the critic network. Once the training is done, the approximated optimal cost-to-go function $J_0(x)$ is given by the critic network as

$$J_0(x) = W_0^T \phi(x) = -0.4276 \times 10^{-10}x^7 - 0.1554 \times 10^{-8}x^6 - 0.4842 \times 10^{-7}x^5 - 0.5592 \times 10^{-5}x^4 - 0.5019 \times 10^{-4}x^3 + 0.5050\,x^2 - 0.6973\,x + 21.77.$$

Neglecting terms with coefficients of order 10^{-5} and less, the approximated $J_0(x)$ is a parabola, which confirms the results given in Theorem 3. This result can also be observed through looking at the shape of $J_0(x)$ as plotted versus x in Fig. 6, without neglecting any term. The roots of the derivative of $J_0(x)$ turned out to be 0.6905, 71.223, 19.681±78.812i, -71.211±45.904i, where $i \equiv \sqrt{-1}$. Neglecting the roots which are out of the problem domain and the imaginary roots, the remained root is 0.6905. Hence, the minimum of $J_0(x)$ is very close to the global optimum of $\psi(x)$, as expected.

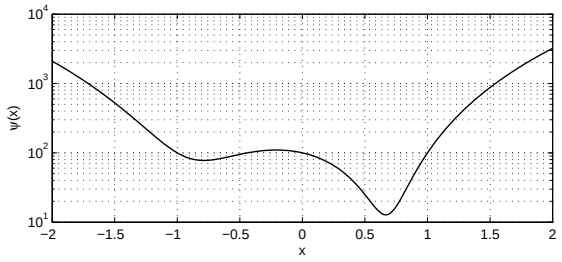

Fig. 4: Function subject to minimization, $\psi(x)$ versus x.

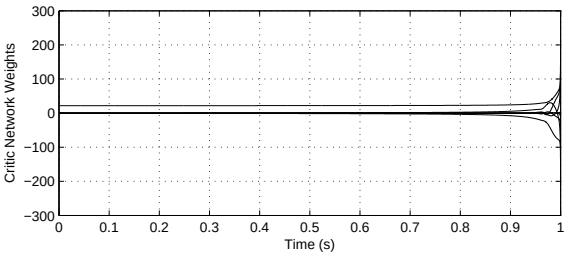

Fig. 5: Weight of the critic network versus time.

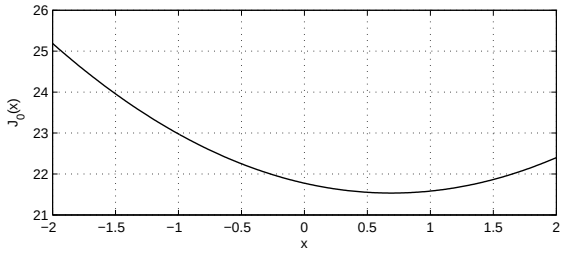

Fig. 6: Optimal cost-to-go versus x.

Five different initial conditions $x_0 \in \{-2,-1,0,1,2\}$ are selected and the dynamics in (45) is simulated for each one while the control is provided by the same trained actor network. Note that per Remark 3, no retraining is needed for different initial conditions. The resulting state and control histories are presented in Figs. 7 and 8, respectively. As seen in Fig. 7, the ADP scheme has resulted in $x_N \cong 0.700$ for different initial conditions, which is very close to the global optimum of $\psi(x)$, that is 0.67. It should be noted that some of the initial conditions are selected to be on the left side of the local minimum of $\psi(x)$. It shows that the method does not go towards the local minimum. The controller, for such initial conditions, has passed the local minimum and has reached to the proximity of the global minimum.

Considering the applied control histories given in Fig. 8, the controls have been constant in the majority of the time as expected based on Eq. (51). However, some anomalies are observed at the end of the horizon, which could be due to the numerical errors of using NNs with finite number of basis functions. Instead of propagating the initial conditions x_0s to the terminal points x_Ns using the dynamics given in (45), one can use the relation given in (52) to fine x_N in one shot. Doing so for the selected five initial conditions leads to the terminal points x_N varying between 0.678 and 0.717 which still are very close to the global optimum.

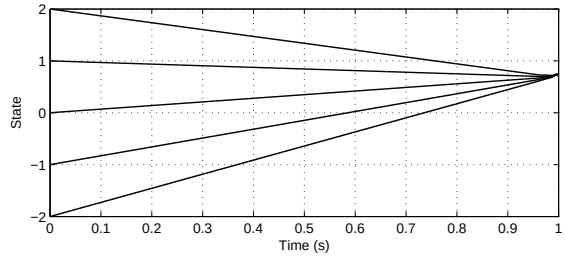

Fig. 7: State histories for different initial conditions $x_0 \in \{-2,-1,0,1,2\}$

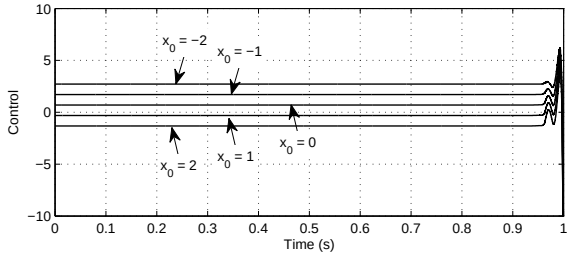

Fig. 8: Control histories for different initial conditions $x_0 \in \{-2,-1,0,1,2\}$

An interesting feature of developed ADP scheme is providing optimal solution for every shorter horizon as well, without needing to retrain the networks. This is due to Bellman's principle of optimality [18] which says that every optimal policy for a given horizon remains optimal for the shorter horizons as well. Let the shorter horizon be $t_f = 0.5$ s. Note that having the optimal control for $t \in [0,1)$, the optimal control for the horizon of $t \in [0,0.5)$ is given by utilizing the network weights corresponding to the *last* half of the original horizon. In other words, if W_0 to W_{10000} give the optimal critic weighs for $t \in [0,1)$, then, W_{5000} to W_{10000} give the optimal weighs for $t \in [0,0.5)$. Simulating the same five initial conditions using the shorter horizon of $t_f = 0.5$ s leads to the state and control histories given in Figs. 9 and 10, respectively. The resulting terminal points are still 0.700, close to the global optimum of $\psi(x)$. Comparing the applied control histories given in Fig. 10 for the shorter horizon, $t_f = 0.5$ s, with the applied control histories resulted from the longer horizon of $t_f = 1$ s, given in Fig. 8, it can be observed that the actor has smartly applied almost twice larger controls in order to get to point x_N in half the time that the controls given in Fig. 8 were applied.

The plot of the optimal cost-to-go function, for the time-to-go of 0.5 s, which is given by $W_{5000}^{T}\phi(x)$, is supper-imposed on the plot of the same function for the time-to-go of 1 s, which is given by $W_{0}^{T}\phi(x)$, in Fig. 11. The minimum of the new parabola has slightly moved, from 0.690 in $t_f = 1$ s case to 0.695 in $t_f = 0.5$ s case. This observation is compatible with the theoretical result given in section 5, i.e., as N becomes larger, the results get closer to the global minimum of $\psi(.)$. Considering the mathematical formula for the new cost-to-go function, given below, shows that the new parabola has almost twice the slope that the previous parabola had, in order to generate twice larger controls for the shorter horizon.

$$W_{5000}^{T}\phi(x) \cong 1.0203x^2 - 1.4086x + 22.0202, \text{ for } t_f = 0.5 \text{ s}.$$

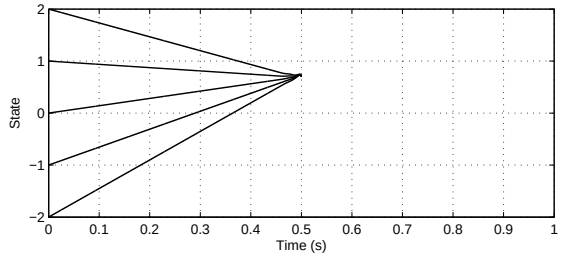

Fig. 9: State histories for different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\}$, with $t_f = 0.5$.

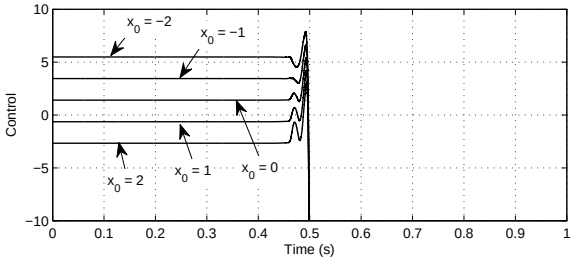

Fig. 10: Control histories for different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\}$, with $t_f = 0.5$.

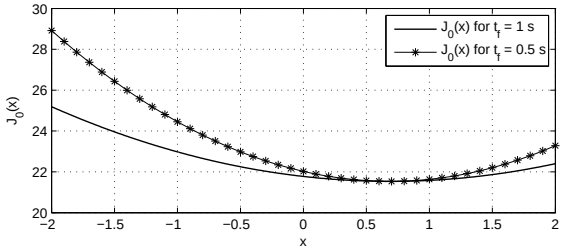

Fig. 11: Optimal cost-to-go versus x, for different t_fs.

It is noteworthy that as discussed in Example 2 in section 4, if the horizon is shortened too much without refining the sampling time, the method may fail to lead to the global optimum of $\psi(.)$. This is showed in Fig. 12, where, the very short horizon of $t_f = 0.01$ s with the sampling time used before, is utilized for propagating the five initial conditions. As seen in this figure, some of the initial conditions are led to the proximity of the global minimum of $\psi(.)$, while, some others are deviated toward the local minimum of the function at $x = -0.79$. Re-training the network for the short horizon of $t_f = 0.01$ s but with smaller sampling time was seen to solve this issue and present the same behavior observed in Figs. 7 and 9.

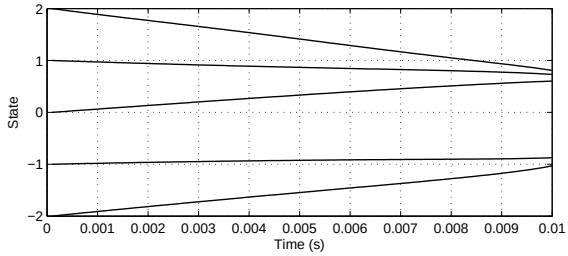

Fig. 12: State histories for different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\}$, with $t_f = 0.01$.

B. Optimizing a Multi-variable Function

For the second example, a benchmark optimization problem is selected, namely, the Rosenbrock (Banana) function [31].

$$\psi(X,Y) = (1-X)^2 + (Y-X^2)^2$$

The Rosenbrock function is a non-convex function with two independent variables (See Fig. 13). It has a global minimum at $(X,Y) = (1,1)$. Defining the state vector as $x \equiv [X,Y]^T$, the ADP scheme is utilized for solving the optimal control problem with the cost function given in (47) and the discretized dynamics (45). The design parameters are selected as $c = 5$, $t_0 = 0$, $t_f = 10$ s, $\Delta t = 0.005$, hence, $N = 2000$. Moreover, the basis functions are selected as

$$\phi(x) = [1, X, Y, XY, X^2, Y^2, \dots, X^8, Y^8]$$
$$\sigma(x) = [1, X, Y, XY, X^2, Y^2, \dots, X^7, Y^7]$$

The least squares method is done using 100 randomly selected points from the domain of interest $\Omega \equiv [-2,2] \times [-2,2]$, which is a typical region selected for this benchmark problem [32]. Once the network training is carried out, the approximated optimal cost-to-go function $J_0(X,Y)$ is observed to be given by the paraboloid

$$J_0(X,Y) = W_0^T \phi(x) \cong 0.0498X^2 + 0.0482Y^2 - 0.0012XY - 0.0972X - 0.1002Y - 0.1107,$$

where the higher order terms with the coefficient less than 10^{-5} are skipped. Function $J_0(X,Y)$, without neglecting the higher order terms, is plotted in Fig. 14. Finding the roots of the jacobian of $J_0(X,Y)$, which leads to the extrema of $J_0(X,Y)$, gives the only real root of $(X,Y) = (0.988, 1.048)$. It is seen that the sole minimum of the cost-to-go function is very close to the global minimum of the Rosenbrock function.

In order to analyze the performance of the developed method in global minimization of the cost function, 25 different initial conditions $x_0 \in \{-2,-1,0,1,2\} \times \{-2,-1,0,1,2\}$ are selected and separately simulated using the controls provided by the same trained actor network (see Remark 3). The resulting state trajectories are presented in Fig. 15. In this figure, the level curves of the Rosenbrock function are also plotted. It can be seen that each one of the initial conditions has been directly led toward the global minimum, regardless of the orientation of the level curves at the points. This behavior shows the advantage of the method over, for example, steepest descent method, in which, the states move in the direction that is perpendicular to the local level curves [18,17].

The state trajectories are super-imposed on the level curves of the optimal cost-to-go function $J_0(x_0)$ in Fig. 16. In this figure, however, one can see that the directions of movement of the trajectories are (almost) perpendicular to the local level curves of the cost-to-go function. This shows the analogy between the developed method and the steepest descent method, with the difference that the ADP algorithm follows the gradient of the cost-to-go function, not that of $\psi(.)$. It should be noted that each u_k, for different ks, will be selected to be perpendicular to the level curves of the respective $J_k(.)$, while, only those of $J_0(.)$ are plotted in Fig. 16. However, the level sets of the rest of $J_k(.)$s also were observed to have the same shape, i.e., circles/ellipses centered at a point close to the global minimum point of $\psi(.)$, as expected based on Theorem 3.

7. Conclusions

The performance of approximate dynamic programming in finding the global optimal solution to the fixed-final-time control problem was investigated. A sufficient condition for global optimality of the result, regardless of the convexity or non-convexity of the functions representing the dynamics of the system or the state penalizing terms in the cost function, was derived. Moreover, an idea was presented in converting a static function optimization to an optimal control problem and using ADP for approximating the global minimum of the selected convex or non-convex function. Numerical results showed that the proposed method results in a trajectory which directly goes to the proximity of the global minimum, regardless of the shape of the local level curves. This is a promising feature which differentiates the method from many nonlinear programming based optimization methods.

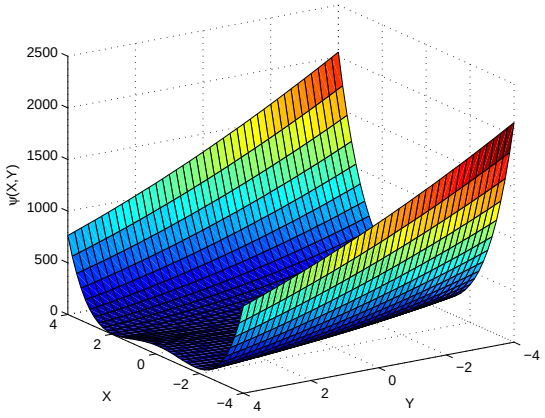

Fig. 13: Rosenbrock function subject to minimization versus the inputs X and Y.

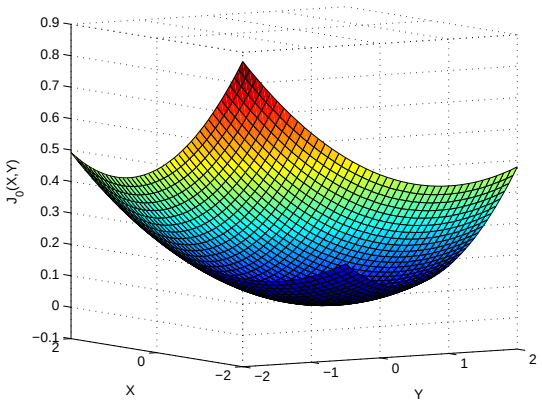

Fig. 14: Optimal cost-to-go versus X and Y.

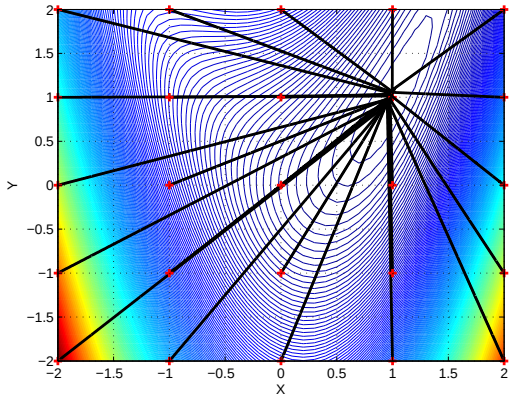

Fig. 15: Level curves of the Rosenbrock function and state trajectories for different initial conditions $x_0 \in \{-2,-1,0,1,2\} \times \{-2,-1,0,1,2\}$. The red plus signs denote the initial point of the respective trajectory.

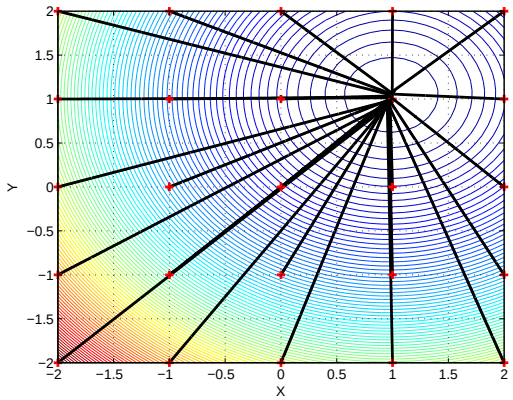

Fig. 16: Level curves of the optimal cost-to-go function and state trajectories for different initial conditions $x_0 \in \{-2,-1,0,1,2\} \times \{-2,-1,0,1,2\}$. The red plus signs denote the initial point of the respective trajectory.

Appendix

In Steps 2, 8, and 9 of Algorithm 1, the least squares method can be used for calculating V_k and W_k. For example, considering Step 8 of the algorithm, the objective is finding V_k such that it solves

$$\begin{cases} {V_k^j}^T \sigma\left(x_k^{[1]}\right) = u_k^{[1]} \\ {V_k^j}^T \sigma\left(x_k^{[2]}\right) = u_k^{[2]} \\ \vdots \\ {V_k^j}^T \sigma\left(x_k^{[n]}\right) = u_k^{[n]} \end{cases}. \tag{57}$$

Define

$$\boldsymbol{\sigma} \equiv \left[\sigma\left(x_k^{[1]}\right) \;\; \sigma\left(x_k^{[2]}\right) \;\; \ldots \;\; \sigma\left(x_k^{[n]}\right)\right],$$

$$\boldsymbol{u} \equiv \left[u_k^{[1]} \;\; u_k^{[2]} \;\; \ldots \;\; u_k^{[n]}\right].$$

Using the method of least squares, the solution to system of linear equations (57) is given by

$$V_k = (\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\boldsymbol{u}^T \tag{58}$$

Note that for the inverse of matrix $(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)$, which is a $p \times p$ matrix, to exist, one needs the basis functions $\sigma(.)$ to be linearly independent and n to be greater than or equal to the number of the basis functions.

Reference:

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

[2] S. N. Balakrishnan, and V. Biega, "Adaptive-critic based neural networks for aircraft optimal control", *Journal of Guidance, Control & Dynamics*, Vol. 19, No. 4, 1996, pp. 893-898.

[3] D. V. Prokhorov, and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Networks*, Vol. 8, No. 5, 1997, pp. 997-1007.

[4] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Trans. Systems, Man, and Cybernetics*-Part B, Vol. 38, 2008, pp. 943-949.

[5] T. Dierks, B. T. Thumati, and S. Jagannathan, "Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence," *Neural Networks*, vol. 22, pp. 851-860, 2009.

[6] S. Ferrari, and R. F. Stengel, "Online adaptive critic flight control," *Journal of Guidance, Control and Dynamics*, vol. 27 (5), pp. 777-786, 2004.

[7] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13 (3), pp. 764-773, 2002.

[8] R. Padhi, N. Unnikrishnan, X. Wang , and S. N. Balakrishnan, "A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems," *Neural Networks*, vol. 19, pp.1648–1660, 2006.

[9] J. Ding and S. N. Balakrishnan, "An online nonlinear optimal controller synthesis for aircraft with model uncertainties," in Proc. *AIAA Guidance, Navigation, and Control Conference*, 2010.

[10] D. Han and S. N. Balakrishnan, "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 4, 2002, pp. 481-489.

[11] A. Heydari and S. N. Balakrishnan, "Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 24 (1), 2013, pp. 145-157.

[12] A. Heydari and S. N. Balakrishnan, "Fixed-final-time Optimal Control of Nonlinear Systems with Terminal Constraints," *Neural Networks*, vol. 48, pp. 61-71, 2013.

[13] F. Wang, N. Jin, D. Liu, and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw.*, vol. 22 (1), pp. 24-36, 2011.

[14] R. Song and H. Zhang, "The finite-horizon optimal control for a class of time-delay affine nonlinear system," *Neural Comput & Applic*, DOI 10.1007/s00521-011-0706-3, 2011.

[15] Q. Wei, and D. Liu, "An iterative ϵ-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state," *Neural Networks*, vol. 32, pp. 236-244, 2012.

[16] F. L. Lewis and D. Liu, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, John Wiley & Sons, New Jersey, 2012.

[17] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999, pp. 1-187.

[18] D. E. Kirk*, Optimal control theory: an introduction*. New York: Dover Publications, 2004, pp. 54,75,332.

[19] K. Homik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[20] J. G. Attali, and G. Pages, "Approximations of Functions by a Multilayer Perceptron: a New Approach," *Neural Networks*, vol. 10 (6), pp. 1069-1081, 1997.

[21] M. Stone, and P. Goldbart, *Mathematics for Physics - A Guided Tour for Graduate Students*, Cambridge, England, Cambridge University Press, p. 70, 2009.

[22] H. Khalil, *Nonlinear Systems*, 3rd ed., Prentice Hall, New Jersey, 2002, pp.653-656.

[23] J. E. Marsden, T. Ratiu, and R. Abraham, *Manifolds, Tensor Analysis, and Applications*, 3rd ed. Springer-Verlag Publishing Co., New York, 2001, p. 73.

[24] W. F. Trench, *Introduction to Real Analysis*, available online at http://ramanujan.math.trinity.edu/wtrench/texts/TRENCH_REAL_ANALYSIS.PDF, 2012, pp. 313.

[25] C.-T. Chen, *Linear System, Theory and Design*, Oxford University Press, New York, 3rd Edition, 1999, p. 131.

[26] B. Chachuat, *Nonlinear and Dynamic Optimization: From Theory to Practice*, EPFL, 2007, pp.120-121.

[27] T. T. Shannon, and G. G. Lendaris, "A new hybrid critic-training method for approximate dynamic programming," Proceedings of *International Society for the System Sciences*, ISSS'2000. Available online at: web.pdx.edu/~tads/papers/20060.pdf.

[28] S. Boyd, and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2009, pp.4-7,71,459.

[29] J. R. Magnus, and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*, 3rd ed. John Wiley & Sons, 2007, p. 231.

[30] A. Tiwari, R. Roy, G. Jared, and O. Munaux, "Evolutionary-based techniques for real-life optimisation: development and testing," *Applied Soft Computing*, vol. 1 (4), 2002, pp. 301-329.

[31] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," *The Computer Journal*, vol. 3, pp 175-184, 1960.

[32] S. Siva Sathya, M.V. Radhika, "Convergence of nomadic genetic algorithm on benchmark mathematical functions," *Applied Soft Computing*, vol. 13, pp. 2759-2766, 2013.

[33] Online: (http://webpages.sdsmt.edu/~aheydari/Research/SourceCodes/GlobalOptimality_SourceCode.html).