Revisiting Approximate Dynamic Programming and its Convergence

Ali Heydari

Abstract: Value iteration based approximate/adaptive dynamic programming (ADP) as an approximate solution to infinite-horizon optimal control problems with deterministic dynamics and continuous state and action spaces is investigated. The learning iterations are decomposed into an outer loop and an inner loop. A relatively simple proof for the convergence of the outer-loop iterations to the optimal solution is provided using a novel idea with some new features. It presents an analogy between the value function during the iterations and the value function of a fixed-final-time optimal control problem. The inner loop is utilized to avoid the need for solving a set of nonlinear equations or a nonlinear optimization problem numerically, at each iteration of ADP for the policy update. Sufficient conditions for the uniqueness of the solution to the policy update equation and for the convergence of the inner-loop iterations to the solution are obtained. Afterwards, the results are formulated as a learning algorithm for training a neuro-controller or creating a look-up table to be used for optimal control of nonlinear systems with different initial conditions. Finally, some of the features of the investigated method are numerically analyzed.

1- Introduction

Approximate dynamic programming has shown great promises [1-13] in circumventing the problem of curse of dimensionality existing in the dynamic programming [14,15] approach to solving optimal control problems. Using the framework of Adaptive Critic (AC) for approximating the optimal cost-to-go/costate and the optimal control was observed to provide accurate approximations of the optimal solutions to real-world nonlinear problems in different disciplines from flight to turbogenerator control [4,8,9,10]. ACs are typically composed of two neural networks (NN) [15]; a) the critic which approximates the mapping between the current state of the system and the optimal cost-to-go (called value function) in Heuristic Dynamic Programming (HDP) [6,7,16] or the optimal costate vector in Dual Heuristic Programming (DHP) [4,8,10], and b) the actor which approximates the mapping between the state of the system and the optimal control. Therefore, the solution will be in a feedback form. ADP can also be implemented using a single network [17] which provides significant computational advantages over the dual network framework. There are two different approaches in implementing the iterations of the ADP; policy iteration (PI) and value iteration (VI) [3,18]. The advantage of PI is the stabilizing feature of the solution during the learning iterations, which makes the method attractive for online learning and control. However, PI requires an initial stabilizable control to start the iterations with. This restrictive requirement makes it impracticable in many nonlinear systems, especially when the internal dynamics of the system is unknown. The VI framework, however, does not need a stabilizing initial guess and can be initialized arbitrarily. Moreover, if VI is implemented online, no knowledge of the internal dynamics of the system is required [6]. It will not require the model of the input gain matrix as well in case of utilizing Action Dependent HDP (ADHDP) [15] or Q-Learning [2].

The applications of ADP to different problems are extensively investigated, including the recent activities in using ADP for continuous-time problems [19,20], finite-horizon problems [21-23], hybrid problems [24,25], etc. However, the convergence of the iterative algorithms corresponding to ADP is not well investigated yet. Ref. [26] analyzed the convergence for the case of linear systems and quadratic cost function terms. A rigorous convergence proof was developed for general nonlinear control-affine systems under VI in [6], assuming the policy update equation can be solved exactly at each iteration. The idea developed in [6] was later adapted by [27-30] for convergence proof of problems with some differences, including tracking, constrained control, non-affine dynamics, and finite-horizon cost function. As for PI, a convergence analysis was recently presented in [31].

In this study, an innovative idea is presented for the proof of convergence of the VI-based ADP algorithms, including HDP and DHP, in solving infinite-horizon optimal control problems. The idea is establishing an analogy between the parameters subject to iteration in the VI-based ADP algorithm and the optimal solution to a finite-horizon optimal control problem with a fixed final time. It is shown that the parameters under the approximation at each iteration are identical to the optimal solution to a finite-horizon optimal control problem with a horizon equal to the iteration index. Moreover, it is shown that the solution to the finite-horizon optimal control problem converges to the solution to the respective infinite-horizon problem as the horizon extends to infinity. Using these characteristics, it is proved that the VI converges to the optimal solution of the infinite-horizon problem at hand.

Another contribution of this study is decomposing the learning iterations to an outer loop and an inner loop and providing the proof of convergence of the inner loop, beside that of the outer loop. While the outer loop is the same as in traditional implementations of VI, the inner loop is suggested to remedy the problem of having a nonlinear equation, namely the policy update equation, which needs to be solved numerically for updating the control vector at each iteration of VI-based algorithms. Typically, nonlinear programming based methods are utilized to solve this equation or the respective nonlinear optimization problem [5,6]. In here, however, a successive approximation based algorithm is suggested and sufficient conditions for its

1Assistant Professor of Mechanical Engineering, South Dakota School of Mines and Technology (e-mail: ali.heydari@sdsmt.edu).
the unique fixed point using any initial guess on the unknown parameter is presented. Finally, a new learning algorithm is proposed that does not require the weight update of the actor at each iteration.

Comparing the results presented in this study with the existing convergence proofs including Ref. [6], there are several differences, listed here. 1) The idea and the line of proof given here are completely different from [6] and relatively simpler. 2) The new convergence proof provides some ideas on the required number of iterations for convergence, namely, it is equal to the number of time steps in the horizon using which, the value function of the respective finite-horizon problem converges to the value function of the infinite-horizon problem at hand. Moreover, it provides an understanding of the characteristics of the (immature) value function under iteration since it is identical to the (mature) value function of a finite-horizon problem. Such understanding is very useful especially for stability analysis of applications with online learning. 3) The new convergence result admits the case of non-zero initial guess on the parameter subject to iterations while the results presented in [6] are for zero initial guess. 4) Considering the idea behind the convergence proof of HDP presented in here, the convergence of DHP also follows, as included in this study. 5) The result presented in [6] assumes the policy update equation can be strictly solved at each iteration. This condition is met for linear systems, however, it is not generally possible for nonlinear systems to solve the policy update equation analytically and as proposed in [6], numerical gradient based methods are suggested to be used for nonlinear systems. In this study, however, the inner loop is introduced for solving the policy update equation through successive approximations and the sufficient conditions for convergence and uniqueness of the result are investigated. 6) In the learning algorithm presented in this study, one only needs to update the weights of the critic in the learning iterations and no actor weight update is needed until the end of the algorithm once the value function is converged. This leads to a considerable computational efficiency, especially considering the fact that the prescribed weight update of the actor in [6] is supposed to be done at each learning iteration after the respective gradient based numerical solution to the policy update equation is converged.

The rest of this study is organized as follows. The problem definition is given in Section 2. The HDP based solution is discussed in Section 3 and its convergence analyses are presented in Section 4. The learning algorithm for implementing HDP is summarized in Section 5. The extension of the convergence result to DHP is discussed in Section 6. Some numerical analyses are presented in Section 7 and followed by concluding remarks in Section 8.

2- Problem Formulation

The infinite-horizon optimal control problem subject to this study is formulated as finding control sequence \( \{u_0,u_1,u_2, \ldots \} \), denoted by \( \{u_\kappa\}_{\kappa=0}^{\infty} \), that minimizes the cost function given by

\[
J = \sum_{\kappa=0}^{\infty} (Q(x_\kappa) + u_\kappa^T R u_\kappa),
\]

subject to the discrete-time nonlinear control-affine dynamics represented by

\[
x_{\kappa+1} = f(x_\kappa) + g(x_\kappa)u_\kappa, \quad k \in \{0, 1, 2, \ldots \},
\]

with the initial condition of \( x_0 \). The state and the control vectors are given by \( x_\kappa \in \mathbb{R}^n \) and \( u_\kappa \in \mathbb{R}^m \), respectively, where positive integers \( n \) and \( m \), respectively, denote the dimensions of the state and the control vectors. Moreover, positive semi-definite smooth function \( Q: \mathbb{R}^n \rightarrow \mathbb{R}_+^n \) penalizes the states, positive definite matrix \( R \in \mathbb{R}^{m \times m} \) penalizes the control effort, smooth function \( f: \mathbb{R}^n \rightarrow \mathbb{R}_+^n \) represents the internal dynamics of the system, and smooth function \( g: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m} \) is the matrix-valued input gain function. The set of non-negative reals is denoted with \( \mathbb{R}_+ \) and the subscripts on the state and the control vectors denote the discrete time index.

3- HDP-based Solution

Considering cost function (1), the incurred cost, or cost-to-go, resulting from propagating the states from the given state of \( x_\kappa \) at the current time to infinity, using dynamics (2) and control sequence \( \{u_\kappa\}_{\kappa=0}^{\infty} \), may be denoted with \( V(x_\kappa, \{u_\kappa\}_{\kappa=0}^{\infty}) \). In other words,

\[
V(x_\kappa, \{u_\kappa\}_{\kappa=0}^{\infty}) = \sum_{\kappa=0}^{\infty} (Q(x_\kappa) + u_\kappa^T R u_\kappa),
\]

where in the summation in the right hand side of (3) one has \( x_\kappa = f(x_{\kappa-1}) + g(x_{\kappa-1})u_{\kappa-1} \), \( \forall \kappa \in \{k+1, k+2, \ldots \} \). From (3), one has

\[
V(x_\kappa, \{u_\kappa\}_{\kappa=0}^{\infty}) = Q(x_\kappa) + u_\kappa^T R u_\kappa + V(x_{\kappa+1}, \{u_\kappa\}_{\kappa=0}^{\infty})
\]

\[
= Q(x_\kappa) + u_\kappa^T R u_\kappa + V(f(x_\kappa) + g(x_\kappa)u_\kappa, \{u_\kappa\}_{\kappa=0}^{\infty}).
\]

Let the cost-to-go resulting from propagating the states from the current state \( x_\kappa \) using the optimal control sequence \( \{u_\kappa^*\}_{\kappa=0}^{\infty} \) be denoted with \( V^*(x_\kappa) \) and called the value function or the optimal cost-to-go function. Function \( V^*(x_\kappa) \) is uniquely defined for the given \( x_\kappa \), assuming the uniqueness of the optimal control sequence. Considering the recursive relation given by (4), Bellman equation [14] gives

\[
V^*(x_\kappa) = \inf_{\{u_\kappa\}_{\kappa=0}^{\infty}} \left( V(x_\kappa, \{u_\kappa\}_{\kappa=0}^{\infty}) \right)
\]

\[
= \inf_{u_\kappa \in \mathbb{R}^m} \left( Q(x_\kappa) + u_\kappa^T R u_\kappa + V^*(f(x_\kappa) + g(x_\kappa)u_\kappa) \right), \quad \forall x_\kappa \in \mathbb{R}^n,
\]

\[
u^*(x_\kappa) = \arg \inf_{u_\kappa \in \mathbb{R}^m} \left( Q(x_\kappa) + u_\kappa^T R u_\kappa + V^*(f(x_\kappa) + g(x_\kappa)u_\kappa) \right), \quad \forall x_\kappa \in \mathbb{R}^n,
\]
which lead to
\[ u^*(x) = -\frac{1}{2} R^{-1} g(x)^T \nabla V^*(f(x) + g(x)u^*(x)), \forall x \in \mathbb{R}^n, \] \[ V^*(x) = Q(x) + u^*(x)^T R u^*(x) + V^*(f(x) + g(x)u^*(x)), \forall x \in \mathbb{R}^n, \]
where \( \nabla V(x) \) defined as \( \partial V(x) / \partial x \) is formed as a column vector. Bellman equation gives the optimal solution to the problem, however, the so called curse of dimensionality \([14,15]\) leads to the mathematical intractability of the approach for many nonlinear problems.

The VI-based ADP proposes the solution to the optimal control problem to be calculated through an iterative fashion. More specifically, selecting the HDP approach, the value function and the optimal control are approximated in closed forms, i.e., as functions of the current state. These approximations are conducted within a compact set representing the domain of interest, to be selected based on the system/problem and denoted with \( \Omega \). Denoting the iteration index by superscript \( i \), the approximations of \( u^*(x) \) and \( V^*(x) \) at the \( i \)th iteration are, respectively, denoted with \( u^i(x) \) and \( V^i(x) \). The iteration starts with selecting a guess on \( V^0(x) \), \( \forall x \in \Omega \). Afterwards, one iterates through the value update equation
\[ V^{i+1}(x) = Q(x) + u^i(x)^T R u^i(x) + V^i(f(x) + g(x)u^i(x)), \forall x \in \Omega, \]
for \( i = 0, 1, 2, \ldots \) until the iterations converge. Note that at each iteration of the value update equation, one needs to solve the so called policy update equation given below, to find \( u^i(\cdot) \) based on \( V^i(\cdot) \).
\[ u^i(x) = \arg \min_{u \in \mathbb{R}^n} \left( Q(x) + u^T R u + V^i(f(x) + g(x)u) \right) = -\frac{1}{2} R^{-1} g(x)^T \nabla V^i(f(x) + g(x)u^i(x)), \forall x \in \Omega, \]
It is straightforward to see the analogy between the value and policy update equations (7) and (8) and Bellman equation (5) and (6).

### 4- Convergence Analysis

The fact that HDP is an iterative method, as seen through equations (7) and (8), gives rise to the question that if the iterations converge. Theorem 1 proves the convergence of the VI-based HDP to the optimal solution to the problem. The proof of the theorem, presented in this paper, is based on establishing an analogy between the iteration index in the VI algorithm and the solution to a corresponding finite-horizon problem. Therefore, a brief review of the finite-horizon problem and its solution is presented first.

Let the finite-horizon optimal control problem with a fixed final time, given by \( N \), be defined as minimizing cost function \( J^N \) defined as
\[ J^N = \psi(x_N) + \sum_{k=0}^{N-1} (Q(x_k) + u_k^T R u_k), \]
such that the state equation (2), where positive semi-definite smooth function \( \psi : \mathbb{R}^n \to \mathbb{R}_+ \) penalizes the terminal states and \( Q(\cdot) \) and \( R \) are the same as in infinite-horizon cost function (1). Let the control at time \( k \) that minimizes \( J^N \) subject to (2) for the given current state \( x_k \) be denoted with \( v^{*,N-k}(x_k) \) and the resulting (finite-horizon) value function at time \( k \) and state \( x_k \) be denoted with \( V^{*,N-k}(x_k) \), i.e.,
\[ V^{*,N-k}(x_k) = \psi(x_N) + \sum_{k=0}^{N-1} (Q(x_k) + v^{*,N-k}(x_k)^T R v^{*,N-k}(x_k)), \]
where \( x_0 = f(x_{k-1}) + g(x_{k-1})v^{*,N-(k+1)}(x_{k-1}), \forall k \in [0, k, k + 2, \ldots, N] \), in the summation in the right hand side of (10). In other words, the states are propagated using the respective optimal control vectors. It is important to note that in finite-horizon optimal control, both the optimal control and the value function depend on a) the current state \( x_k \) and b) the time-to-go \((N - k)\) \([22,30]\).

The value function and the optimal solution to this finite-horizon problem are given by Bellman equation \([14]\)
\[ V^{*,0}(x) = \psi(x), \forall x \in \Omega, \]
\[ v^{*,N-k}(x) = \arg \min_{u \in \mathbb{R}^n} \left( Q(x) + u^T R u + V^{*,N-(k+1)}(f(x) + g(x)u) \right), \forall x \in \Omega, k = 0, 1, \ldots, N - 1, \]
\[ V^{*,N-k}(x) = Q(x) + v^{*,N-k}(x)^T R v^{*,N-k}(x) + V^{*,N-(k+1)}(f(x) + g(x)v^{*,N-k}(x)), \forall x \in \Omega, k = 0, 1, \ldots, N - 1. \]
More specifically, Eq. (11) gives \( V^{*,0}(\cdot) \), which once used in (12) for \( k = N - 1 \) gives \( v^{*,N}(\cdot) \) and then Eq. (13) leads to \( V^{*,N}(\cdot) \). Repeating this process the value function and the optimal control for the entire horizon can be found in a backward fashion, from \( k = N - 1 \) to \( k = 0 \) \([22]\). Therefore, no VI is required for finding the value function and solving the finite-horizon problem, assuming the policy update equation (12) can be solved analytically. The important observation is the fact that, considering equations (11)- (13), if the optimal solution for the last \( N \) steps of a finite-horizon problem is known in a closed form, then the solution to the problem with the longer horizon of \((N + 1)\) steps can be calculated directly. In other words, if the value function at time \( k = 0 \) with the finite horizon of \( N \), i.e., \( V^{*,N}(x) \), is available, then the optimal control and the value function at time \( k = 0 \) for the problem with the longer horizon of \((N + 1)\) can be calculated as
\[ v^{*,N+1}(x) = \arg \min_{u \in \mathbb{R}^n} \left( Q(x) + u^T R u + V^{*,N}(f(x) + g(x)u) \right), \forall x \in \Omega. \]
\[ V^{*,N+1}(x) = Q(x) + v^{*,N+1}(x)^T R v^{*,N+1}(x) + V^{*,N}(f(x) + g(x)v^{*,N+1}(x)), \forall x \in \Omega. \]
Considering these observations, the convergence theorem is presented next.

**Theorem 1**: If the nonlinear system given by (2) is controllable, then the VI-based HDP given by Eqs. (7) and (8) converges to the optimal solution, using any initial guess $V^0(x)$ such that $V^0(\cdot)$ is smooth and $0 \leq V^0(x) \leq Q(x)$, $\forall x \in \Omega$, e.g. $V^0(x) = Q(x)$ or $V^0(x) = 0$, $\forall x \in \Omega$.

**Proof**: The first iteration of HDP using (7) and (8) leads to

$$u^0(x) = \arg\min_{u \in \mathbb{R}^m}(Q(x) + u^T R u + V^0(f(x) + g(x)u)), \forall x \in \Omega,$$

$$V^1(x) = Q(x) + u^0(x)^T R u^0(x) + V^0(f(x) + g(x)u^0(x)), \forall x \in \Omega.$$  

Selecting $\psi(\cdot)$ in the finite-horizon control problem of minimizing $J^1$ equal to $V^0(\cdot)$, i.e.,

$$\psi(x) = V^0(x), \forall x \in \Omega,$$  

and considering (16), vector $u^0(x)$ is identical to the solution to $J^1$ subject to (2), i.e.,

$$u^0(x) = v^{s_1}(x), \forall x \in \Omega.$$  

Moreover, based on (17), $V^1(x)$ is identical to the value function for the problem of minimizing $J^1$, i.e.,

$$V^1(x) = V^{s_1}(x), \forall x \in \Omega.$$  

Now, assume that for some $i$ one has

$$V^i(x) = V^{s_i}(x), \forall x \in \Omega.$$  

Conducting the $i$th iteration of HDP leads to Eqs. (7) and (8). Considering (21), it follows from (8) and (14) that

$$u^i(x) = v^{s_{i+1}}(x), \forall x \in \Omega.$$  

Moreover, from Eq. (7) and (15), considering (21) and (22), one has

$$V^{i+1}(x) = V^{s_{i+1}}(x), \forall x \in \Omega.$$  

Since (19) and (20) hold and the assumption given by (21) leads to (22) and (23), the relations given by (22) and (23) hold for all $i$s, by mathematical induction.

So far it is proved that the parameters subject to iteration in the VI-based ADP at the $i$th iteration are identical to the solution of a finite-horizon problem with the fixed final time of $i$. What remains to show is the proof of convergence of the solution of the finite-horizon problem to that of the infinite-horizon problem at hand. To this goal, let $i < j$, where $i$ and $j$ are any two positive integers. Then

$$V^{s_i}(x_0) \leq V^0(x_i) + \sum_{k=0}^{i-1} \left( Q(x_k) + v^{s_j-k}(x_k)^T R v^{s_j-k}(x_k) \right), \forall x_0 \in \Omega,$$

where in the summation in the right hand side of (24) one has $x_k = f(x_{k-1}) + g(x_{k-1})v^{s_j-(k-1)}(x_{k-1})$, $\forall k \in \{1, 2, \ldots, i\}$. Relation (24) holds, otherwise, instead of $\{v^{s_k-k}(x_k)\}_{k=0}^{i-1}$, control sequence $\{v^{s_j-k}(x_k)\}_{k=0}^{i-1}$ will be the optimal solution to $J^i$. Moreover, for every selected $x_0 \in \Omega$, propagating the initial state $x_0$ to $x_i$ using control sequence $\{v^{s_j-k}(x_k)\}_{k=0}^{i-1}$, one has

$$V^0(x_i) \leq V^0(x_j) + \sum_{k=0}^{i-1} \left( Q(x_k) + v^{s_j-k}(x_k)^T R v^{s_j-k}(x_k) \right),$$

where $x_k = f(x_{k-1}) + g(x_{k-1})v^{s_j-(k-1)}(x_{k-1})$, $\forall k \in \{1, 2, \ldots, i\}$, in the summation in the right hand side of (25). Relation (25) holds because $0 \leq V^0(x_i) \leq Q(x_i), \forall x_i \in \Omega$, and the fact that $Q(x_i)$ along with some non-negative terms exist in the right hand side of (25). Adding $\sum_{k=0}^{i-1} \left( Q(x_k) + v^{s_j-k}(x_k)^T R v^{s_j-k}(x_k) \right)$, where $x_k = f(x_{k-1}) + g(x_{k-1})v^{s_j-(k-1)}(x_{k-1})$, $\forall k \in \{1, 2, \ldots, i-1\}$, to both sides of (25) and considering the fact that $x_i$ is calculated using the optimal control sequence for the problem with horizon $j$ leads to

$$V^0(x_i) + \sum_{k=0}^{i-1} \left( Q(x_k) + v^{s_j-k}(x_k)^T R v^{s_j-k}(x_k) \right) \leq V^{s_j}(x_i), \forall x_i \in \Omega,$$

by definition of $V^{s_j}(x_0)$, i.e., by Eq. (10). From (24) and (26) it follows that $V^{s_i}(x) \leq V^{s_j}(x), \forall x \in \Omega$. In other words, sequence $\{V^{s_k}(x)\}_{k=0}^{\infty}$ is a (pointwise) non-decreasing sequence. The controllability of the system leads to the fact that there exists a stabilizing control policy for every selected initial condition $x_0 \in \Omega$ using which cost function $J$ remains finite [32]. The existence of such a control policy leads to the upper boundedness of sequence $\{V^{s_k}(x)\}_{k=0}^{\infty}$. The reason is, if this sequence is not upper bounded, then the applied control history which results in an unbounded cost-to-go is not optimal, compared to the existing stabilizing control policy. The boundedness of $\{V^{s_k}(x)\}_{k=0}^{\infty}$ and its non-decreasing feature lead to its convergence, because every non-decreasing and upper bounded sequence converges [35]. Let the converged value, which is the least upper bound to the sequence also, be denoted with $V^{*\infty}$. The upper boundness of sequence $\{V^{s_k}(x)\}_{k=0}^{\infty}$ leads to the fact that utilizing the optimal control sequence $\{v^{s_k}(x)\}_{k=0}^{N-1}$, one has

$$Q(x_N) \to 0 \text{ as } N \to \infty,$$

otherwise the cost-to-go becomes unbounded, because of the summation of infinite number of positive terms which do not converge to zero. Moreover, due to $0 \leq V^N(x) \leq Q(x), \forall x \in \Omega$, and (27) one has

$$V^0(x_N) \to 0 \text{ as } N \to \infty.$$
Considering (28) and comparing (9) with (1), it follows that \( J^N \to J_\infty \) as \( N \to \infty \). Therefore, by definition one has \( V^{*,\infty} = V^* \), otherwise, the smaller value between \( V^{*,\infty} \) and \( V^* \) will be the optimal cost-to-go to the infinite-horizon problem, and also, the least upper bound to sequence \( \{V^{*,k}(x)\}_{k=0}^\infty \). Therefore,

\[
V^{*,N}(x) \to V^*(x) \quad \text{as} \quad N \to \infty, \forall x \in \Omega, \tag{29}
\]

\[
v^{*,N}(x) \to u^*(x) \quad \text{as} \quad N \to \infty, \forall x \in \Omega. \tag{30}
\]

Relations (29) and (30) along with (22) and (23), which hold for all \( k \), complete the proof of the convergence of VI-based HDP to the optimal solution.

The approach presented for proving the convergence of VI-based ADP through Theorem 1 is new and unlike [6], which is the approach utilized in many research papers including [27-30]. Besides a simpler line of proof and admitting a more general initial guess \( V^0(\cdot) \), an important feature of this proof is providing a more intuitive idea on the iteration process and the (immature) value function during the training iterations. For example, once it is shown that the (immature) value function under iterations is identical to the (mature/perfect) value function of a finite-horizon problem with the horizon corresponding to the iteration index, the problem of finding the required number of iterations for the convergence of the VI algorithm simplifies to finding the required horizon for convergence of the solution of the finite-horizon problem to the solution of the respective infinite-horizon problem. Considering this analogy, one can qualitatively, and sometime quantitatively, predict the required number of iterations for the convergence of the VI. This feature is numerically analyzed in Section 7.

Having proved the convergence of the HDP iterations, a new concern emerges in implementation of the method. The problem is the existence of term \( u^i(x) \) in both sides of the policy update equation, i.e., Eq. (8). Hence, in order to calculate \( u^i(x) \) one needs to solve Eq. (8) for every given \( x \). Generally, this equation is a set of \( m \) nonlinear equations with the \( m \) unknown elements of control vector \( u^i(x) \). The remedy suggested in [6] for this issue is using gradient based numerical methods, including Newton method or Levenberg–Marquardt method, for finding the unknown \( u^i(x) \) through (8). In other words, at each iteration of (7), one needs to solve one set of nonlinear equations to find \( u^i(x) \). Instead of this process, another set of iterations are suggested in this study for finding \( u^i(x) \) and its convergence proof is given. This new set of iterations, indexed by the second superscript \( j \), is given below

\[
u^{i,j+1}(x) = -\frac{1}{2} R^{-1} g(x)^T \nabla V^i(f(x) + g(x) u^{i,j}(x)), \quad \forall x \in \Omega. \tag{31}
\]

In other words, for calculating \( u^i(x) \) to be used in (7), one can select a guess on \( u^{i,0}(x) \) and iterate through (31), by calculating \( u^{i,j+1}(x) \) using \( u^{i,j}(x) \). Once these iterations converge (Theorem 2), the result, denoted with \( u^i(x) \), can be used in (7). Therefore, there will be an outer-loop iteration given by (7) and an inner-loop iteration given by (31). Theorem 2 provides the sufficient condition for the convergence of the inner-loop iterations given by (31).

**Theorem 2**: The iterations given by Eq. (31) converge using any initial guess on \( u^{i,0}(x) \in R^m \), providing the selected \( V^0(\cdot) \) as well as functions \( Q(\cdot), f(\cdot), \) and \( g(\cdot) \) are smooth versus their inputs and one of the following conditions holds:

1) The norm of matrix \( R^{-1} \) is small enough.

2) The norm of matrix-valued function \( g(x) \) is small enough, \( \forall x \in \Omega \).

**Proof**: Let the right hand side of (31) be denoted with function \( \mathcal{F} : R^m \rightarrow R^m \) where

\[
\mathcal{F}(u) = -\frac{1}{2} R^{-1} g(x)^T \nabla V^i(f(x) + g(x) u). \tag{32}
\]

If it can be shown that the relation given by the successive approximation

\[
u^{i,j+1} = \mathcal{F}(u^{i,j}) \tag{33}
\]

is a contraction mapping, then the convergence to the unique fixed point follows [33]. Since \( R^m \) with 2-norm denoted with \( \|\cdot\| \) is a Banach space, the iterations given by (33) converges to some \( u = \mathcal{F}(u) \) if there is a \( 0 \leq \rho < 1 \) such that for every \( u \) and \( v \) in \( R^m \), the following inequality holds

\[
\|\mathcal{F}(u) - \mathcal{F}(v)\| \leq \rho \|u - v\|. \tag{34}
\]

From Eq. (32) one has

\[
\|\mathcal{F}(u) - \mathcal{F}(v)\| \leq \left\| \frac{1}{2} R^{-1} g(x)^T \nabla V^i(f(x) + g(x) u) - \frac{1}{2} R^{-1} g(x)^T \nabla V^i(f(x) + g(x) v) \right\| \tag{35}
\]

It can be seen that function \( V^i(x) \) is smooth versus its input, \( x \), for all \( i \), since functions \( V^0(\cdot), Q(\cdot), f(\cdot), \) and \( g(\cdot) \) are smooth. Considering Eq. (7), since \( V^0(x) \) is a smooth function, function \( V^i(x) \), and hence \( \nabla V^i \) are smooth. Smoothness of \( g(x) \) and \( \nabla V^i \) leads to the smoothness of function \( u^i(x) \), by (8), which along with the smoothness of \( Q(x) \) and \( V^i(x) \) lead to a smooth \( V^i(x) \), by (7). Repeating this argument from 1 to \( i \), it follows that \( V^i(x) \) is a smooth function. This smoothness leads to the Lipschitz continuity of \( \nabla V^i \) in the compact domain of interest \( \Omega \) [34]. In other words, there exists some positive real number \( \rho_i \) such that for every \( x \) and \( y \) in \( \Omega \), one has \( \|\nabla V^i(x) - \nabla V^i(y)\| \leq \rho_i \|x - y\| \). Using this characteristic, inequality (35) can be written as

\[
\|\mathcal{F}(u) - \mathcal{F}(v)\| \leq \frac{1}{2} \rho_i \|R^{-1}\| \|g(x)\| \|u - v\| \tag{36}
\]
By defining
\[ \rho \equiv \frac{1}{2} \rho_1 \| R^{-1} \| \| g(x) \|^2 \] (37)
inequality (36) leads to (34). Note that the continuity of \( g(x) \) in its domain result in its norm being bounded in compact set \( \Omega \) [35], hence, the state-dependent term in (37) is upper bounded. Either having a large enough \( R \), which leads to a small enough \( \| R^{-1} \| \), or a small enough \( \| g(x) \| \), \( \forall x \in \Omega \), leads to \( 0 \leq \rho < 1 \). This completes the proof of existence of a unique fixed point, denoted with \( u^i \), and the convergence of \( u^{i,j} \) to \( u^i \), as \( j \to \infty \), \( \forall u^{i,0} \in \mathbb{R}^m \), using the successive approximations given by (31). □

As seen in Theorem 2, the convergence of (31) requires certain conditions either on control penalizing matrix \( R \), or on input gain matrix \( g(x) \). In practice, the design objectives may not let the designer to change \( R \) such that the convergence is assured. In this case, it can be seen that if the discrete-time dynamics given by (2) is obtained through discretization of a continuous-time dynamics, then utilizing small enough sampling time, the second sufficient condition for the convergence can always be met, i.e., a small enough sampling time leads to the desired characteristic of small enough \( \| g(x) \| \). To see this, assume the original continuous-time dynamics with the respective cost function be given by
\[ \dot{x}(t) = f(x(t)) + g(x(t))u(t), \]
\[ J_c = \int_0^\infty \tilde{Q}(x(t)) + u(t)^T \tilde{R}u(t)dt. \] (39)
Let the sampling time be denoted with \( \Delta t \). Using Euler integration scheme, the continuous-time problem can be discretized to (1) and (2), where \( R = \Delta t \tilde{R} \), \( \tilde{Q}(x) = \Delta t \tilde{Q}(x) \), \( f(x) = x + \Delta t \tilde{f}(x) \), \( g(x) = \Delta t \tilde{g}(x) \), \( x_k = x(k\Delta t) \), and \( u_k = u(k\Delta t) \). Representing \( \rho \) given in (37) in terms of the continuous-time problem, one has
\[ \rho = \frac{1}{2} \rho_1 \Delta t \| R^{-1} \| \| \tilde{g}(x) \|^2. \] (40)
As seen in (40), for every selected \( \tilde{R} \) and every smooth (hence, bounded in \( \Omega \)) input gain matrix \( \tilde{g}(x) \), there exists a non-zero sampling time using which one has \( \sup_{x \in \Omega} \rho \leq 1 \). Utilizing any sampling time smaller than that sampling time, leads to the desired convergence.

Note that, as \( \Delta t \) is refined, function \( V^i(x) \) and hence coefficient \( \rho_i \) will change in the right hand side of (40). However, it can be seen that \( V^i(x) \) and its gradient with respect to \( x \) converge to some respective smooth functions as \( \Delta t \to 0 \), hence, \( \rho_i \), will remain bounded as \( \Delta t \to 0 \). To see this, one should note that \( V^i(x) \), as shown in the proof of Theorem 1, is identical to the optimal cost-to-go for the problem of minimizing \( J^i \), given in (9), subject to (2), i.e., \( V^i(x) = V^{i,i}(x) \). On the other hand, as \( \Delta t \to 0 \), discrete-time cost function (9) converges to the respective continuous-time cost function
\[ J_c^{i,f} = V^0(x(t_f)) + \int_0^{t_f} \tilde{Q}(x(t)) + u(t)^T \tilde{R}u(t)dt. \] (41)
where \( t_f = i \Delta t \). Denoting the cost-to-go resulting from solving the problem of minimizing \( J_c^{i,f} \) subject to (38) with \( V_c^{*,i,f}(x) \), one has \( V^{*,i}(x) \to V_c^{*,i,f}(x) \), and hence, \( V^i(x) \to V_c^{*,i,f}(x) \), as \( \Delta t \to 0 \). The smoothness of \( \tilde{f}(\cdot) \), \( \tilde{g}(\cdot) \), and \( \tilde{Q}(\cdot) \) leads to the smoothness of the control and hence, the smoothness of \( V_c^{*,i,f}(x) \) by composition. This smoothness property leads to the smoothness of \( \partial V_c^{*,i,f}(x)/\partial x \), and hence, its Lipschitz continuity [34]. Denoting the Lipschitz coefficient of \( \partial V_c^{*,i,f}(x)/\partial x \) by \( \rho_i^{*,f} \), one has
\[ \lim_{\Delta t \to 0} \rho_i = \rho_i^{*,f}, \] (42)
because of \( V^i(x) \to V_c^{*,i,f}(x) \), as \( \Delta t \to 0 \). Eq. (42) guarantees the boundedness of \( \rho_i \) as \( \Delta t \) is decreased for the purpose of ending up with \( 0 \leq \rho < 1 \).

Before concluding the section, it is worthwhile to look at the linear case to clarify the need for the prescribed inner-loop iterations. Assuming linear system \( x_{k+1} = Ax_k + Bu_k \) and quadratic cost function \( J = \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k) \), the value function is approximated by \( V^i(x_k) = x_k^T P^i x_k \) for some \( P^i \in \mathbb{R}^{n \times n} \) subject to iteration. Considering this assumption, optimal control equation (5) reads
\[ u^i(x_k) = -\frac{1}{2} R^{-1} B^T P^i x_{k+1} = -\frac{1}{2} R^{-1} B^T P^i \left( Ax_k + B u^i(x_k) \right), \] (43)
which is similar to (8), unknown \( u^i(x_k) \) exists in both sides of the equation. However, policy update equation (43) is linear and the analytical solution can be calculated, that is
\[ u^i(x_k) = -(2R + B^T P^i B)^{-1} B^T P^i A x_k, \] (44)
which is a familiar equation in LQR when \( P^i \) is replaced with the optimal \( P \) obtained from the Riccati equation. If solution (44) was not available, one could use the following (inner-loop) iterations, starting with any initial guess \( u^{i,0} \), to find \( u^i \)
\[ u^{i,j+1}(x_k) = -\frac{1}{2} R^{-1} B^T P^i \left( A x_k + B u^{i,j}(x_k) \right). \] (45)
Following the idea presented in proof of Theorem 2, it is straightforward to show that (45) is a contraction mapping, and hence \( u^{i,j+1}(x_k) \) converges to the solution of (43), providing \( \| B \| \) is small enough. But, if the original continuous time is given by
\( \dot{x}(t) = Ax(t) + Bu(t) \), then, \( B = \Delta t \bar{B} \). Hence, a small enough \( \Delta t \) leads to a small enough \( \|B\| \). Therefore, as long as one can solve the set of \( m \) equations (43) in the linear problem, or the nonlinear equation (8) in the general nonlinear case, no inner-loop iteration and hence, no condition on the sampling time is required.

**Remark 1**: Unlike the approach of resorting to nonlinear programming for finding \( u^i(x) \) through (8) [6], whose success is dependent on the initial guess on the unknown parameter and is susceptible of getting stuck in a local minimum, the result of inner-loop iterations (31) does not depend on the initial guess on \( u^i(x) \) and will not get stuck in a local minimum. The reason is the contraction mapping basis of the successive approximations given by (31). In other words, once it is shown that (31) is a contraction mapping, as done in Theorem 2, the uniqueness of the fixed point and the convergence to the fixed point using any initial guess on the unknown parameter are guaranteed [33].

**Remark 2**: Except the problems with inherent hybrid/discrete nature, all of the real-world dynamical systems are representable by continuous-time state equations. Discretizing the state equations of such systems using a small enough sampling time, the condition of the input gain function having a small enough norm can always be met, when the input gain function is a continuous function. Hence, this (sufficient) condition does not refrain one from applying the proposed method to such problems.

**Remark 3**: As seen in (40) and (42), one needs the knowledge about the Lipschitz constant of the value function in order to calculate the suitable sampling time ahead of the training phase. This knowledge may not be generally available. However, Theorem 2 proves that selecting a small enough sampling time, e.g., through trial and error, the convergence is guaranteed. Hence, one can start the iterations given by (31) and in case of divergence decrease the sampling time.

### 5- Implementation of HDP Using Neural Networks

For implementation of the VI-based HDP scheme, one may use two NNs, called the *actor* and the *critic*, as function approximators for approximating the mapping between input \( x \) and outputs \( u^i(x) \) and \( V^i(x) \), respectively [15]. Considering the proposed inner-loop and the outer-loop iterations of the HDP method, the training algorithm may be summarized as given in Algorithm 1. Interested readers are referred to [37] for a source code implementing this algorithm.

**Algorithm 1**: HDP algorithm with inner loop and outer loop for training actor and critic networks

1. Randomly select \( n \) sample state vectors \( x^{[i]} \in \Omega \), \( \forall j \in J \equiv \{1,2,..,n\} \), where \( n \) is a selected large positive integer, and \( \Omega \) denotes a compact subset of \( \mathbb{R}^n \) representing the domain of interest.
2. Set \( i = 0 \).
3. Initialize constant scalars \( V^i(x^{[i]}), \forall j \in J \), e.g., \( V^i(x^{[i]}) = 0 \).
4. Train the critic network using input-target pairs \( \{x^{[i]}, V^i(x^{[i]})\}, \forall j \in J \).
5. Set \( j = 0 \).
6. Initialize constant vectors \( u^j(x^{[i]}), \forall j \in J \), e.g., randomly.
7. Calculate \( u^{j+1}(x^{[i]}), \forall j \in J \), using Eq. (31). Gradient \( \nabla V^j \) may be calculated using the critic network, to be used in (31).
8. If \( \|u^{j+1}(x^{[i]}) - u^j(x^{[i]})\| < \beta_u \), \( \forall j \in J \), where \( \beta_u > 0 \) is a small preset tolerance, then set \( u^i(x^{[i]}) = u^{j+1}(x^{[i]}), \forall j \in J \), and proceed to the next step, otherwise set \( j = j + 1 \) and go back to Step 7.
9. Calculate \( V^{i+1}(x^{[i]}), \forall j \in J \), using Eq. (7).
10. If \( \|V^{i+1}(x^{[i]}) - V^i(x^{[i]})\| < \beta_V \), \( \forall j \in J \), where \( \beta_V > 0 \) is a small preset tolerance, then proceed to the next step, otherwise set \( i = i + 1 \) and go back to Step 4.
11. Train the actor network using input-target pairs \( \{x^{[i]}, u^i(x^{[i]})\}, \forall j \in J \).

**Remark 4**: Algorithm 1 is composed of two separate loops. The inner loop is composed of Steps 7 and 8 and the outer loop is composed of Steps 4 to 10.

**Remark 5**: It can be seen that the critic network needs to be updated at each iteration of the outer loop, while the actor is trained only once at the end of the algorithm. In other words, for implementation of the iterative relation (31) one does not need to train the actor network at each iteration of the inner loop or the outer loop. The iteration can be done using the selected *pointwise* values for function \( u^j(\cdot) \) at different \( x \)s through (31) until the pointwise values of \( u^j(\cdot) \)s converge. Note that Theorem 2 proves the convergence of these iterations. Afterwards, the converged \( u^j(\cdot) \) values, denoted with \( u^j(\cdot) \), will be used in (7). Once the outer-loop iterations given by (7) converge, the final pointwise values of \( u^i(\cdot) \)s will be used for training the actor. However, the critic network is different and needs to be re-trained at each iteration of (7). The reason is the evaluation of the \( V^{i+1}(\cdot) \) at \( x \) in the left hand side of (7), while, \( V^i(\cdot) \) is evaluated at \( f(x) + g(x)u^i(x) \) in the right hand side of equation (7). Also, \( V^i(\cdot) \) is evaluated at \( f(x) + g(x)u^{i-1}(x) \) in the right hand side of equation (31). Note that \( f(x) + g(x)u^i(x) \), in equation (7), is subject to change after each set of iterations of the inner loop, due to the change in \( u^i(\cdot) \). Also, \( f(x) + g(x)u^{i-1}(x) \), in
Note that, the optimal control is a function of this derivative, as seen in (5) which in terms of horizon problem is given in the proof of Theorem 1. Considering these results, the convergence of the DHP iterations to the hold for all $o x$.

**Remark 6**: Another option in implementation of the HDP scheme is using look-up tables for the approximation of $u^*(\cdot)$ and $V^*(\cdot)$, instead of using NNs for learning the mapping. In this case, there will be no network training and interpolations between the available values of $V^*(\cdot)$ in the look-up table can be used for evaluating the right hand sides of (7) and (31). Once the iterations converge, one needs to interpolate between the available values for the control at different $x$s to calculate the feedback control in online implementation.

**Remark 7**: Algorithm 1 is presented for training the networks in a batch form, suitable for offline training. However, the algorithm for sequential learning, which is more suitable for online learning, can be derived similarly.

## 6- DHP-based Solution and Its Convergence

Another approach for implementation of ADP is dual heuristic programming (DHP) [5]. In DHP instead of the value function, the derivative of the value function with respect to $x$, called the costate vector and denoted with $\lambda^*(x)$, is approximated. Note that, the optimal control is a function of this derivative, as seen in (5) which in terms of $\lambda^*(x)$ is given by

$$u^*(x) = -\frac{1}{2} R^{-1} g(x)^T \lambda^*(f(x) + g(x) u^*(x)), \forall x \in \mathbb{R}^n,$$

hence, having the costate vector, one can calculate the optimal control. Taking the derivative of equation (6) leads to the costate equation

$$\lambda^*(x) = \partial Q(x)/\partial x + A^T (x, u^*(x)) \lambda^*(f(x) + g(x) u^*(x)), \forall x \in \Omega,$$

where $A(x, u) \equiv \partial(f(x) + g(x) u)/\partial x$. The DHP is carried out through initializing $\lambda^0(x), \forall x \in \Omega$, and using the following iterations

$$u^i(x) = -\frac{1}{2} R^{-1} g(x)^T \lambda^i \left( f(x) + g(x) u^i(x) \right), \forall x \in \Omega,$$

$$\lambda^{i+1}(x) = \partial Q(x)/\partial x + A^T (x, u^i(x)) \lambda^i \left( f(x) + g(x) u^i(x) \right), \forall x \in \Omega.$$

It is straightforward to extend the idea utilized in the proof of Theorem 1 to prove the convergence of the DHP scheme. To shed some lights on the line of proof, one may consider the optimal solution to finite-horizon cost function (9), with $\psi(.) = V^0(.)$, given below in terms of costates [30]

$$\lambda^0(x) = \partial V^0(x)/\partial x, \forall x \in \Omega,$$

$$v^*(x) = -\frac{1}{2} R^{-1} g(x)^T \lambda^*(f(x) + g(x) v^*(x)), \forall x \in \Omega, k = 0, 1, ..., N - 1,$$

$$\lambda^{N-k}(x) = \partial Q(x)/\partial x + A^T (x, v^*(x)) \lambda^{N-k} \left( f(x) + g(x) v^*(x) \right), \forall x \in \Omega, k = 0, 1, ..., N - 1,$$

where $\lambda^{N-k}(x) \equiv \partial V^0(x)/\partial x$. If $\lambda^0(x)$ is selected such that $\lambda^0(x) = \partial V^0(x)/\partial x$ for some smooth $V^0(x)$ which satisfies $0 \leq V^0(x) \leq Q(x), \forall x$, e.g. $\lambda^0(0) = 0, \forall x \in \Omega$, using the same line of proof given in the proof of Theorem 1, it can be seen that relations (22) and

$$\lambda^{i+1}(x) = \lambda^{i+1}(x), \forall x \in \Omega.$$

hold for all $i$ in DHP. Moreover, the proof that the solution to the finite-horizon problem converges to the solution to infinite-horizon problem is given in the proof of Theorem 1. Considering these results, the convergence of the DHP iterations to the optimal solution follows.

As for the inner loop, the nonlinear equation given in (48) can be replaced with the inner loop iterations

$$u^{l,i+1}(x) = -\frac{1}{2} R^{-1} g(x)^T \lambda^i \left( f(x) + g(x) u^{l,i}(x) \right), \forall x \in \Omega,$$

similar to the HDP case. Using the idea presented in Theorem 2, it is also straightforward to show that assuming the conditions given in Theorem 2 hold, equation (54) is a contraction mapping, hence, it converges to a unique fixed point.

## 7- Numerical Analysis

In order to numerically analyze the performance of the ADP based controller discussed in this study, two examples are selected; a benchmark second order problem and a third order real world problem. The source codes (in MATLAB®) for the numerical examples presented in this paper are available at [37]. For implementation of Algorithm 1, the linear-in-weight NN structures given by

$$V^*(x) \equiv W_c^T \phi(x)$$

$$u^*(x) \equiv W_a^T \sigma(x)$$

are selected for approximating the desired nonlinear mappings in both examples. Functions $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^{n_c}$ and $\sigma: \mathbb{R}^n \rightarrow \mathbb{R}^{n_a}$ are the selected basis functions and positive integers $n_c$ and $n_a$ denote the number of neurons in the critic and the actor networks, respectively. Unknown matrices $W_c \in \mathbb{R}^{n_c}$ and $W_a \in \mathbb{R}^{n_a \times n_m}$ denote the network weights for the critic and the actor, respectively.
It should be noted that one can use multi-layer NNs for improving the approximation capability, compared with the linear-in-weight NNs utilized in this study.

Example 1:

The selected system for the first example is a nonlinear benchmark system, namely Van der Pol’s oscillator, with the dynamics of

\[
\dot{x}_1 = x_2
\]
\[
\dot{x}_2 = (1 - x_1^2)x_2 - x_1 + u
\]

where \(x_i, i = 1, 2\), denotes the state vector elements. The cost function given in (39) with terms

\[
\tilde{Q}(x) = 100(x_1^2 + x_2^2), \quad \tilde{R} = 0.5,
\]

are used for the simulation. The sampling time of \(\Delta t = 0.02\ s\) is selected for discretization of the continuous-time problem.

An important step in the neurocontroller design is the selection of the basis functions. Weierstrass approximation theorem [36] proves that any continuous function on a closed and bounded interval can be uniformly approximated by polynomials to any degree of accuracy. Therefore, basis functions \(\phi(.)\) are selected as polynomials made up of different combinations of the state elements up to the fourth order, which leads to \(n_c = 14\). Moreover, \(\sigma(.)\) is selected as the same type of polynomials up to the third order, leading to \(n_a = 9\). The network is trained for the domain of \(\Omega = [-2.5, 2.5] \times [-2.5, 2.5]\). In implementation of Algorithm 1 the parameters subject to iterations are initialized at zero and 50 random sample states are selected, i.e., \(n = 50\). As for the network training method, the least squares approach is used for finding the weights in Steps 4 and 11 of Algorithm 1 [22].

Fig. 1 shows the evolution of the weights of the critic versus the outer-loop iterations, as well as the evolution of \(u^{l/-}(.)\)s (at the last outer-loop iteration) versus the inner-loop iterations. As seen in this figure, the inner-loop iterations converged in almost 5 iterations. The outer-loop iterations, however, took around 150 iterations to converge. Note that, as explained in Algorithm 1, each single outer-loop iteration requires conducting a set of inner-loop iterations until they converge. In order to analyze the effect of the sampling time in the convergence speed, a smaller sampling time, namely \(\Delta t = 0.005\ s\) is selected and the evolution of the weights in the respective training of the networks is shown in Fig. 2. Based on the result given in the proof of Theorem 2, a smaller sampling time which leads to a smaller \(\|g(x)\|\), should increase the speed of convergence of the inner loop, due to leading to a smaller contraction mapping coefficient \(\rho\) given in (40). Considering the evolution of \(\{u^{l/-}(.)\}\)s at the inner-loop iterations, given in Fig. 2, they have converged much faster, in around 2 iterations. However, the convergence of the outer-loop iterations has slowed down, compared to the case of \(\Delta t = 0.02\ s\).

The convergence proof presented for Theorem 1 provides some insights into the required number of outer-loop iterations for convergence, namely, it is equal to the required horizon for the convergence of the solution of the finite-horizon problem to the solution of the respective infinite-horizon problem. As shown in the proof of Theorem 1, and as expected by mathematical intuition, the solution to a finite-horizon problem converges to the respective infinite-horizon problem as the horizon extends beyond a certain value. Let this “certain value” be called convergence horizon. Based on Fig. 1, the convergence horizon (for the given original continuous-time problem) is approximately \(150\Delta t = 3\ s\), since the iterations converged about \(i = 150\). Moreover, as long as the sampling times are selected small enough such that the discretized problem accurately approximates the original continuous-time problem, it is expected that the resulting state trajectories, control histories, and hence, value functions be almost the same for different small sampling times. Therefore, utilizing another small sampling time should lead to the same convergence horizon, by its definition. Considering Fig. 2, the iterations converged after almost 600 iterations, which leads to the convergence horizon of \(600\Delta t = 3\ s\), based on the utilized sampling time of \(\Delta t = 0.005\ s\). As seen, the calculated convergence horizons turned out to be identical. This feature can be used for predicting the required number of iterations for convergence of the VI-Based ADP. As an example, selecting a new sampling time of \(\Delta t = 0.002\ s\) is expected for the VIs to converge after \(3/\Delta t = 1500\) iterations. To evaluate the accuracy of the prediction, the training is done based on the new sampling time and the results are presented in Fig. 3. It is seen that the predicted number of iterations for the convergence of outer-loop iterations turned out to be accurate.

As for optimality analysis of the numerical results, once the networks are trained, the actor is used for controlling initial condition \(x_0 = [2, 1.5]^{T}\) for 5 seconds with the sampling time of \(\Delta t = 0.02\ s\). It should be noted that the initial condition is selected with elements larger than one in order for the nonlinear part of the system to be dominant over the linear part, i.e., to be able to analyze the capability of the controller in controlling systems with non-negligible nonlinearities. The resulting state histories are depicted in Fig. 4. As seen in the figure, the controller has been able to regulate the states to the origin. In order to check the optimality of the result, an open loop optimal solution is calculated through direct method of optimization and the results are superimposed on the ADP results in Fig. 4. The comparison shows that the ADP has been able to provide a very good approximation of the optimal solution. To further analyze the optimality and the versatility of the ADP based solution, the cost-to-go resulting from simulating different initial conditions using the same trained network along with the optimal cost-to-gos resulting from the open loop numerical method calculated through separate runs for each single initial condition, are listed in Table 1. The values listed in the table show that the same trained network has been able to provide very accurate approximations of the optimal solutions for different initial conditions.
Figure 1. Evolution of the unknown parameters versus the outer-loop and inner-loop iterations, for the case of $\Delta t = 0.02$ s.

Figure 2. Evolution of the unknown parameters versus the outer-loop and inner-loop iterations, for the case of $\Delta t = 0.005$ s.

Figure 3. Evolution of the unknown parameters versus the outer-loop and inner-loop iterations, for the case of $\Delta t = 0.002$ s.

Figure 4. State trajectory resulting from ADP versus the optimal trajectory, for Example 1 with initial conditions of $[2, 1.5]^T$.

Table 1: Cost-to-go comparison between the ADP solution and the optimal open loop solution.

<table>
<thead>
<tr>
<th>Initial Condition</th>
<th>Cost-to-go of optimal open loop sol.</th>
<th>Cost-to-go of ADP sol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[-2, 1]^T$</td>
<td>414.37</td>
<td>414.45</td>
</tr>
<tr>
<td>$[1, -2]^T$</td>
<td>113.59</td>
<td>113.66</td>
</tr>
<tr>
<td>$[2, 0]^T$</td>
<td>437.28</td>
<td>437.34</td>
</tr>
<tr>
<td>$[1.5, 0]^T$</td>
<td>245.55</td>
<td>245.59</td>
</tr>
<tr>
<td>$[1.5, 1]^T$</td>
<td>276.62</td>
<td>276.66</td>
</tr>
<tr>
<td>$[1.5, 2]^T$</td>
<td>323.77</td>
<td>323.808</td>
</tr>
<tr>
<td>$[2, 1.5]^T$</td>
<td>498.92</td>
<td>498.99</td>
</tr>
</tbody>
</table>

Example 2:
As the second example, a practical problem is selected to show the applicability of the method to real world problems. The selected problem is detumbling of a rigid spacecraft [38]. A rigid spacecraft is tumbling with some initial angular velocities about each one of its three perpendicular axes. The controller needs to regulate the angular velocities to zero. The equations of motion are given by

$$\dot{x} = I^{-1}(I\dot{x} \times x + u),$$
where \( x \in \mathbb{R}^3 \), \( I \in \mathbb{R}^{3 \times 3} \), and \( u \in \mathbb{R}^3 \) are the rigid body angular velocities, the moment of inertia of the rigid body, and the applied mechanical torque, respectively. Sign ‘\( \times \)’ denotes matrix cross product. The moment of inertia is selected as \( I = \text{diag}(86.24, 85.07, 113.59) \text{ kgm}^2 \). Cost function (39) with the terms given below is selected

\[
\bar{Q}(x) = 10000x^T x, \quad \bar{R} = \text{diag}(2,2,2).
\]

Let the vector whose elements are all the non-repeating polynomials made up through multiplying the elements of vector \( X \) by those of vector \( Y \) be denoted with \( X \otimes Y \). In this example the following basis functions are used:

\[
\phi(x) = \left[ (x \otimes x)^T, (x \otimes (x \otimes x))^T \right]^T, \\
\sigma(x) = [x^T, (x \otimes x)^T]^T,
\]

hence, \( n_e = 16 \) and \( n_a = 9 \).

Selecting the sampling time of \( \Delta t = 0.01 \) s the continuous dynamics is discretized. In implementing Algorithm 1, 500 random states are selected where \( \Omega = \{ x \in \mathbb{R}^3 : -2 \leq x_i \leq 2, i = 1,2,3 \} \). The learning iterations were observed to converge in around 500 outer-loop iterations. Once the networks are trained, they are used for controlling the IC of \( x_0 = [-0.4,0.8,2]^T \text{ rad/s} \), as simulated in Ref. [38]. The resulting state histories are given in Fig. 5. For comparison, the optimal open loop solution, calculated using direct method of optimization, is also plotted in these figures. As seen in Fig. 5, the ADP solution has done a nice job in providing a solution which is very close to the optimal numerical solution. Comparing the ADP solution with the open loop optimal solution it should be noted that the ADP solution has the advantage of providing feedback solutions to different initial conditions, while, each time that the initial condition changes, the open loop numerical solution loses its validity and a new solution needs to be calculated.

![Figure 5](image-url)  
Figure 5. State trajectories resulting from ADP versus the optimal trajectory, for Example 2 with initial conditions of \([-0.4,0.8,2]^T\).

8- Conclusions

The problem of convergence analysis of ADP was revisited and the training algorithm was decomposed into inner-loop and outer-loop iterations. The convergence of the outer-loop to the optimal solution was proved in a new way, through establishing an analogy between the iterations and the optimal solution to a finite-horizon problem. The convergence of the inner loop, however, was proved by showing that it is a contraction mapping. Finally, the convergence results were numerically analyzed in two nonlinear problems.

Reference


