Degradation tolerant control learning for discrete-time affine nonlinear systems

Soha Kanso, Mayank Shekhar Jha, Didier Theilliol

Degradation Tolerant Control Learning for Discrete-Time Affine Nonlinear Systems

Soha KANSO, **Mayank Shekhar JHA**, **Didier THEILLIOL**

CRAN, UMR 7039, CNRS
Université de Lorraine, 54506 Vandoeuvre-lès-Nancy Cedex, France
soha.kanso@univ-lorraine.fr, mayank-shekhar.jha@univ-lorraine.fr,
didier.theilliol@univ-lorraine.fr

Abstract: This paper develops a degradation tolerant optimal control in the framework of Reinforcement Learning (RL). Safety-critical and mission-critical systems require the development of new control designs that maintain system stability and performance specifications but also address incipient degradation. The aim of this work is to decelerate the speed of degradation by minimizing a cost function that includes the rate of evolution of degradation and the performance requirements. The controller is developed for discrete-time nonlinear systems affine in control, where the system's states are affected by a nonlinear degradation. Value iteration (VI) algorithm based approach is developed to find suitable approximations of both optimal control policy and optimal cost, while guaranteeing closed-loop stability and minimization of degradation rate. Offline model-based Adaptive Dynamic Programming (ADP) algorithm is developed and implemented using actor-critic structure which involves training of both actor and critic neural networks (NN). After training the actor NN with the optimal policy, the NN is implemented in real time to generate the input of the system. Simulation example shows the efficiency and feasibility of the algorithm.

Keywords: Adaptive Dynamic Programming, Reinforcement Learning, Actor-Critic Structure, Optimal Control, Fault Tolerant Control, Safety-Critical Systems, Affine Nonlinear Systems.

1. INTRODUCTION

Traditional control system design (Stengel (1994)), (Åström and Wittenmark (1995)) focuses only on the stability and the performance without taking into consideration the effects of aging, fatigue, and damage of the concerned components and without minimizing the risk of failure. However, safety-critical systems (Knight (2002)) arise in several application areas, such as transportation and air-traffic control systems, space systems, nuclear plants and automated industrial processes. The evolution of such complex systems requires the implementation and the development of new control technologies that maintain system stability and performance specifications, and also address incipient fault and graceful performance degradation.

Recent approaches have applied modern control techniques such as adaptive or robust control to address situations where the degree of failure may be unknown. In (Bole et al. (2010)), a fault adaptive control is proposed for incipient fault modes growing to catastrophic failure conditions. The methodology is developed for a finite constrained optimization problem where the model of the system and the degradation is supposed to be known. Moreover, fault tolerant control design (Noura et al. (2009)), (Blanke et al. (2006)) has been developed for various industrial, mission critical and safety critical systems that operate in closed loop, in order to compensate for fault occurrence. More recently new methods are expanded such that useful life of critical systems can be extended. In this context, health aware control has recently become one of the domains where control is being designed, based upon the state of health (SoH) and/or Remaining Useful Life (RUL) prognostics of critical components. Some prominent works have proposed methods to develop control laws that attempt to extend the RUL of component/system such as (Pour et al. (2021), Salazar et al. (2017)). Also, in the framework of model predictive control (MPC), several works were adopted to produce a controller that ensures robustness to particular failures, thus reducing their impact on the system (Brown et al. (2010), Brown et al. (2021)). However, one of the limitations of previous approaches is that it requires a known dynamic of the system. Reinforcement learning has achieved remarkable success for complex systems with unknown dynamics (Lewis and Liu (2012)).

Reinforcement learning refers to agent or actor interacting with its environment and modifying its actions or control policies, based on stimuli received in response to its actions (Buşoniu et al. (2018)). RL theory is well founded on the principles of optimal control theory based on Dynamic Programming (DP). It allows to solve optimization problems using the principle of optimality from DP that provides an essential foundation for understanding RL (Lewis and Vrabie (2009)). In particular, RL can handle optimal control problems for unknown nonlinear systems. Most of the RL methods attempt to achieve the same goal as DP, with less computation and without knowing the model of the environment. The combination of DP and

neural networks (NN) as approximators leads to adaptive dynamic programming (ADP) approaches, introduced by Paul Werbos in 1977 (Werbos (1997)).
In the framework of health aware control design based on RL, very few works have been proposed. (Jha et al. (2019)) presents a RL based approach that is employed to learn an optimal control policy in face of component degradation by using global system transition data and RUL predictions. (Yousefi et al. (2020)) develops a dynamic optimal policy for multi-component systems with individually repairable components using a Q-learning algorithm. To fill in this scientific gap, this paper proposes a learning strategy for degradation tolerant optimal control of nonlinear unstable systems.

In the previous work (Kanso et al. (2023)), a linear quadratic regulator (LQR) and tracker (LQT) was designed, over finite and infinite time horizon, for a discrete-time linear system in the presence of a linear degradation. This paper aims to address nonlinear cases. The main contribution of the paper is to learn a control law that decelerates the speed of degradation while maintaining optimal performance and stability of a nonlinear system. This work examines a degradation tolerant approach for discrete-time nonlinear systems, where the states of the system are considered to be affected by a nonlinear degradation. The approach is developed in the framework of model-based RL, by building a quadratic cost function that includes the speed of degradation and by implementing the Actor-Critic structure. The optimal policy and value function are learned offline using Value Iteration (VI) algorithm and the model of the system, then the trained NNs are implemented in real time.

This paper is arranged as follows. Section 2 introduces the problem statement and the issue addressed. Section 3 presents the proposed methodology where the problem is solved in the framework of RL. Section 4 examines the feasibility of the proposed approach using an academic example. And finally, the conclusion summarizes the significant advances and includes plans for future work.

2. PROBLEM FORMULATION

The degradation of a system components affects the performance and the stability of the system. The state of degradation or deterioration, considered as a health indicator, as well affects directly the remaining useful life of the active system, consequently reducing the usability and the productivity of the system. Moreover, the state of health (SoH) is stimulated by the states of the system, and implicitly affected by the action of the controller. Hence the importance of developing an optimal approach for performing a control action that takes into account the performance requirements, the stability and also the SoH of the system.

This paper focuses on discrete-time nonlinear systems affine in control as:

$$x_{k+1} = f(x_k, d_k) + g(x_k, d_k)u_k \tag{1}$$

where $x_k \in R^n$ is the state vector at time k, $u_k \in R^m$ is the vector of control input, and $d_k \in R^l$ is the vector of degradation's state. $f(.,.) \in R^n$ and $g(.,.) \in R^{n\times m}$ are differentiable and $f(0,.) = 0$. The degradation is described by:

$$d_{k+1} = z(x_k, d_k) \tag{2}$$

where $d_k \in R^l$ is the vector of degradation's state and $z(.,.) \in R^l$ is a differentiable function. The evolution of the degradation is implicitly controlled by the input since it is sensitive to the system's states. Some practical examples of the presented formulation are the degradation of the damping coefficient in electromechanical actuator due to bearing wear (Fu et al. (2017)) and the breakdown of the winding insulation caused by the rise of the winding temperature (Brown et al. (2021)).

First, one defines the augmented states vector $X_k = [x_k, d_k]^T \in R^{(n+l)}$ and condenses the evolution equation into:

$$X_{k+1} = \begin{bmatrix} f(X_k) \\ z(X_k) \end{bmatrix} + \begin{bmatrix} g(X_k) \\ 0 \end{bmatrix} u_k = F(X_k) + G(X_k)u_k \tag{3}$$

In this work, the control policy is defined as a feedback controller of the system's and degradation's states:

$$u_k = h(x_k, d_k) = h(X_k) \tag{4}$$

In order to maintain the performance of the system while minimizing the energy and the speed of evolution of degradation, a quadratic utility function is defined by:

$$r(x_k, u_k, \Delta d_k) = x_k^T Q x_k + \Delta d_k^T Q_1 \Delta d_k + u_k^T R u_k \tag{5}$$

where $\Delta d_k = d_{k+1} - d_k$ is the rate of evolution of degradation. Q, Q_1 and R are symmetric positive definite weighting matrices and $|R| \neq 0$. Thus, the reward can be written as:

$$\begin{aligned} r(X_k, u_k) = X_k^T \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} X_k \\ + [z(X_k) - [0\ 1]\, X_k]^T Q_1 [z(X_k) - [0\ 1]\, X_k] \\ + u_k^T R u_k \end{aligned} \tag{6}$$

The matrix Q_1 determines the relative importance or cost associated with the degradation rate term in the overall cost function. By changing the values of Q_1, the weight of Δd_k can be adjust. Thus, higher values of Q_1 would indicate a higher cost or penalty on the degradation speed.

The notion of optimal behavior is captured by defining a performance index known also as cost function:

$$V_h(X_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(X_i, u_i) \tag{7}$$

with $0 < \gamma \leq 1$ a discount factor that reduces the weight of costs occurring in the future. Note that it is essential to use a discounted performance function for the proposed formulation. This is because if the rate of degradation does not go to zero, then the performance function (7) becomes infinite without the discount factor.
Assumption 1 The dynamical system (1) is controllable on some set $\Omega \subset R^n$ which implies that there exists a control law that asymptotically stabilizes the system on Ω.
Assumption 2 The degradation variable d_k has a maximum value d_{max}, such that if the degradation level at any time k is less than d_{max}, the system remains stable and can be asymptotically stabilized on the set Ω.

Remark 1. It is noted that rate of degradation is not treated like other state variables of the system owing to the fact the degradation mechanism is irreversible in nature. Moreover, the goal is to slow down the degradation and not to eliminate its speed or stabilise it. As such, the augmented systems does not include the rate of degradation as one of the states to avoid stabilisation of the former to an undesired values including negative ones.

Remark 2. Assumption 2 is based on the premise that the effect of degradation on the system dynamics is sufficiently small and it allows us to design a control strategy that guarantees stability and performance for the system under degradation.

To this end, the objective is to find an optimal control input that minimizes the speed of irreversible degradation and extends the operational residual life of the system, while taking into account various factors such as system dynamics and performance criteria.

3. PROPOSED METHODOLOGY

DP approach determines optimal control solutions using Bellman's principle backward-in-time from some desired goal states. This yields offline solution algorithms which are stored and then implemented online, forward-in-time. Using RL, the solution of Hamilton-Jacobi-Bellman equation can be solved-for online in real time. This section presents a novel algorithm based on RL under model based framework, leading to the design of degradation tolerant optimal control.

3.1 Formulation of optimal control problem

The objective of optimal control theory is to find the policy that minimizes the cost function leading to the optimal value:

$$V^*(X_k) = \min_{h(.)} \sum_{i=k}^{\infty} \gamma^{i-k} r(X_i, h(X_i)) \tag{8}$$

Then, the optimal control policy is defined by:

$$h^*(X_k) = \underset{h(.)}{argmin} \sum_{i=k}^{\infty} \gamma^{i-k} r(X_i, h(X_i)) \tag{9}$$

By writing (7) as:

$$V_h(X_k) = r(X_k, u_k) + \gamma \sum_{i=k+1}^{\infty} r(X_i, h(X_i)) \tag{10}$$

It yields to the Bellman equation:

$$V_h(X_k) = r(X_k, u_k) + \gamma V_h(X_{k+1}) \tag{11}$$

According to Bellman's principle of optimality for discrete-time systems, the following Hamilton-Jacobi-Bellman (HJB) is obtained:

$$V_h^*(X_k) = \min_{h(.)} \left(r(X_k, u_k) + \gamma V_h^*(X_{k+1})\right) \tag{12}$$

with the optimal policy as:

$$h^*(X_k) = \underset{h(.)}{argmin} \left(r(X_k, u_k) + \gamma V_h^*(X_{k+1})\right) \tag{13}$$

Since the optimal policy at time k is determined by using the optimal policy at time $k+1$, Bellman's Principle solves the optimal control problem backwards-in-time. It is the basis for Dynamic Programming (DP) algorithms. The optimal control problem is solved by finding the solution of HJB equation (12) for the value function. Then, by substituting the solution in (13) the optimal control is obtained. Due to the nonlinear nature of the HJB equation, it is generally difficult or impossible to find its solution.

3.2 Solving Optimal Control Problem using RL

As referred previously solving the HJB equation is a challenging problem in optimal control and dynamic programming, moreover it can be computationally expensive. To this end, RL has proven to be powerful and effective machine learning technique that allows to handle optimal control problems for nonlinear systems where iterative methods are often used to obtain the solution of Bellman equation indirectly. These methods are classified into two main schemes, namely policy iteration (PI) and value iteration (VI). This paper only focuses on the VI algorithm among others, as dedicated in Algorithm 1 (Lewis and Vrabie (2009)).

Definition 1 A control policy is defined as admissible with respect to (1) on Ω, if it stabilizes (1) on Ω, and yields to a finite cost function.

Algorithm 1 Value Iteration Algorithm

Initialization. Initialize the value function V_0 using (14) and $h_0(X_k)$ with any control policy, not necessarily admissible (see Definition 1).

$$V_0 = X_0^T P_0 X_0 \tag{14}$$

where P_0 is a positive-definite matrix.

Value Update. Update the value using:

$$V_{j+1}(X_k) = r(X_k, h_j(X_k)) + \gamma V_j(X_{k+1}) \tag{15}$$

Policy Improvement. The control policy is improved by:

$$h_{j+1}(X_k) = \underset{h(.)}{argmin} \left(r(X_k, h(X_k)) + \gamma V_{j+1}(X_{k+1})\right) \tag{16}$$

VI does not find the value corresponding to the current policy, but simply performs a single iteration to that value using (15) with j the step index of VI. The proof of convergence of the VI algorithm in the general nonlinear DT setting was presented in (Al-Tamimi et al. (2008)).
It is difficult to solve equations (15) and (16) for nonlinear systems. Fortunately, NN can be used to approximate V_j and h_j at each iteration. In the following, NN based actor-critic structure is used to implement the VI algorithm, as presented in the next section.

3.3 Actor Critic based implementation

VI algorithm can be implemented using actor-critic structure, where an actor component applies an action or control policy to the system, and a critic component assesses the value of that action and the state resulting from it. The learning mechanism maintained by the Actor-Critic structure (Sutton and Barto (2018)) has two steps:

- policy evaluation, executed by the critic and performed by observing the results of applying current control policy on the environment;
- policy improvement, performed by the actor.

The combination of DP, NN, and actor-critic structure results in the ADP algorithms. This latter includes three NNs: two critics NNs and one actor NN. The structural diagram of the ADP algorithm is shown in Fig. 1.

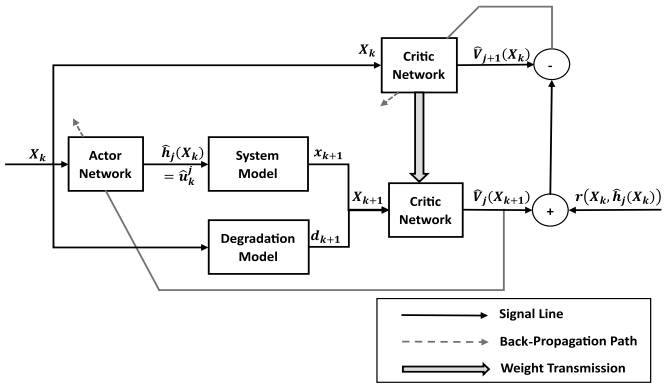

Fig. 1. The structural diagram of ADP algorithm

The first critic NN approximates the relationship between the value function $\hat{V}_{j+1}(X_k)$ and the states vector X_k. While, the second critic NN approximates the relation between the value function $\hat{V}_j(X_{k+1})$, at the previous iteration j, and the states X_{k+1} at instant $k+1$. The actor NN approximates the relationship between control vector $\hat{h}_j(X_k)$ and the states vector X_k. In this paper, the model of the system and the model of the degradation are assumed to be available. Otherwise, the dynamics can be approximated by using available input–output data to train a NN (Liu et al. (2017)).

The output of the critic and actor two-layer networks is described by:

$$\hat{V}_j(X_k) = W_c^{(j)T}\Phi(Y_c^{(j)T}X_k) \tag{17}$$

$$\hat{h}_j(X_k) = W_a^{(j)T}\sigma(Y_a^{(j)T}X_k) \tag{18}$$

where $\Phi = [\Phi_1, ..., \Phi_{N_c}]$ is the vector of hidden-layer activation functions of the critic with $\Phi(.) = sigmoid(.)$ and N_c is the number of hidden-layer neurons. $\sigma = [\sigma_1, ..., \sigma_{N_a}]$ is the vector of hidden-layer activation functions of the actor with $\sigma(.) = tanh(.)$ and N_a is the number of hidden-layer neurons. $W_c^{(j)}$ and $Y_c^{(j)}$ are the critic NN weights, and $W_a^{(j)}$ and $Y_a^{(j)}$ are the actor NN weights. The weights of the critic are obtained from NN training during the j-th iteration, and the weights of the actor are trained only at the final iteration. The Gradient Descent algorithm is used to tune the weights of NNs.
A summary of the VI-based ADP algorithm for optimal control is provided in Algorithm 2.

The developed method has been extended in a tracking control framework also. In the following section, a tracking control design is presented where the dynamics of the system are required.

3.4 Optimal Tracking Control Using ADP

The above VI-based ADP approach can be applied to solve infinite horizon optimal tracking control problem. The objective is to find an optimal control policy $h^*(.)$ that regulates the state x_k to follow a specified trajectory ref_k

Algorithm 2 Value Iteration ADP algorithm

Step 1. Initialize the weights of critic and actor neural networks, and the parameters i^a_{max}, i^c_{max}, ϵ_a, ϵ_c, j_{max}, ξ_c, ξ_a, Q, Q_1, R. ($\xi_a > \xi_c$)

Step 2. Set the iteration index $j = 0$ and $P_0 = \beta I_n = 0$.

Step 3. Choose randomly an array of p state vector $\{x_k^1, x_k^2, ..., x_k^p\}$, and p degradation's state vector $\{d_k^1, d_k^2, ..., d_k^p\}$ to obtain an array of dimension $p \times (n+l)$. Initialize V_0 using (14). Train the critic network using the data until the given accuracy ϵ_c, or the maximum number of iterations i^c_{max} is reached.

Step 4. If $j = 0$, Initialise randomly the vector of initial policy $\{\hat{h}_0(X_k^1), \hat{h}_0(X_k^2), ...\hat{h}_0(X_k^p)\}$ otherwise go to Step.5.

Step 5. Using $\{h_j(X_k^1), h_j(X_k^2), ..., h_j(X_k^p)\}$, compute the output of the system $\{x_{k+1}^1, x_{k+1}^2, ..., x_{k+1}^p\}$ and $\{d_{k+1}^1, d_{k+1}^2, ..., d_{k+1}^p\}$.

Step 6. Compute the output of the critic network $\{\hat{V}_j(X_{k+1}^1), \hat{V}_j(X_{k+1}^2), ..., \hat{V}_j(X_{k+1}^p)\}$.

Step 7. Compute the target of the critic network training $\{V_{j+1}(X_k^1), V_{j+1}(X_k^2), ..., V_{j+1}(X_k^p)\}$ by (15). Train the critic network until the given accuracy ϵ_c, or the maximum number of iterations i^c_{max} is reached.

Step 8. Compute the target of action network $\{h_{j+1}(X_k^1), h_{j+1}(X_k^2), ..., h_{j+1}(X_k^p)\}$ by using:

$$\begin{aligned} h_{j+1}(X_k) &= -\frac{\gamma}{2}R^{-1}G^T\frac{\partial \hat{V}_{j+1}(X_{k+1})}{\partial X_{k+1}} \\ &= -\frac{\gamma}{2}R^{-1}g^T\frac{\partial \hat{V}_{j+1}(X_{k+1})}{\partial x_{k+1}} \end{aligned} \tag{19}$$

Step 9. If $j = j_{max}$ or $|V_{j+1}(X_k^s) - V_j(X_k^s)| \leq \xi_a$ with $s = \{1, 2, ..., p\}$, train the actor NN until the given accuracy ϵ_a, or the maximum number of iterations i^a_{max} is reached; otherwise go to Step 10.

Step 10. Set the iteration index $j = j + 1$. If $j > j_{max}$ or $|V_{j+1}(X_k^s) - V_j(X_k^s)| \leq \xi_c$, go to Step 11; otherwise, go to Step 5.

Step 11. Compute the output of the actor NN $\{\hat{h}_j(X_k^1), \hat{h}_j(X_k^2), ..., \hat{h}_j(X_k^p)\}$. Obtain the final near optimal control law $u^*(.) = \hat{h}_j(.)$, and stop the algorithm.

$\in R^n$. Consider that there exists a feedback control $u_{b,k}$, known by desired control (Liu et al. (2017)), satisfying the following equation :

$$u_{b,k} = g^+(ref_k)[ref_{k+1} - f(ref_k)] \tag{20}$$

where $g^+(ref_k) = (g(ref_k)^T g(ref_k))^{-1} g(ref_k)^T$ is the generalized inverse of $g(ref_k)$ with $g(ref_k)^+ g(ref_k) = I$. By posing $\epsilon_k = x_k - ref_k$ and $\mu_k = u_k - u_{b,k}$, the utility function becomes:

$$r(\epsilon_k, \mu_k, \Delta d_k) = \epsilon_k^T Q \epsilon_k + \mu_k^T R \mu_k + \Delta d_k^T Q_1 \Delta d_k \tag{21}$$

And the objective becomes to find μ_k that minimizes the cost function leading to the optimal value:

$$V^*(\epsilon_k, d_k) = \min_{\mu_k} \sum_{i=k}^{\infty} \gamma^{i-k} r(\epsilon_i, \mu_i, \Delta d_i) \tag{22}$$

To find the solution of the above equation, the iterative ADP algorithm will iterate between value function update (23) and the policy improvement (24):

$$V_{j+1}(\epsilon_k, d_k) = r(\epsilon_k, h_j(\epsilon_k, d_k), \Delta d_k) + \gamma V_j(\epsilon_{k+1}, d_{k+1}) \tag{23}$$

$$h_{j+1}(\epsilon_k, d_k) = \underset{\mu_k}{argmin}\ (r(\epsilon_k, \mu_k, \Delta d_k) + \gamma V_{j+1}(\epsilon_{k+1}, d_{k+1})) \tag{24}$$

The ADP algorithm described above in equations (22-24) is essentially the same as in Algorithm 2.
In the following, the NNs trained for regulation are used for tracking but the main difference is that the inputs of NNs (x_k, d_k) are replaced by (ϵ_k, d_k) and the output of the actor becomes μ_k. To verify the effectiveness of the developed control schemes, an infinite horizon tracker is implemented on an academic example in the next section.

4. SIMULATION RESULTS AND DISCUSSION

Consider the following unstable nonlinear system with affine form:

$$x_{k+1} = \begin{bmatrix} 0.4x_{2,k} \\ 0.3x_{1,k} + x_{2,k} + 0.1d_k x_{2,k} \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} u_k$$

The dynamic of evolution of degradation is described by the following equation:

$$d_{k+1} = 0.01x_{2,k}^2|d_k| + d_k$$

The prior knowledge of system dynamics is assumed to be available. The weighting matrices are chosen as:

$$Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \text{ and } R = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

The three-layers NNs has been chosen as critic network and actor network with the structures of 3–10–1 and 3–10–2 respectively, and $\gamma = 0.98$. The initial weights of NNs are initialized to zero. A random array of data has been generated to train the NNs, the state variables are generated in the range of [-0.7, 0.7] and the degradation state in the range of [0.1, 0.43]. The number of data generated is $p = 2000$. For each iteration j, the critic NN is trained for $i^c_{max} = 4000$ steps under the learning rate 1×10^{-4} so that the approximation error limit 10^{-6} is reached. The actor NN is trained for $i^a_{max} = 6000$ steps under the learning rate 1×10^{-5}, only at the final iteration j_{max} or when the value function converges to the optimal value. After implementing the outer-loop iteration for $j_{max} = 20$ times, the convergence of the value function is observed. If $V_0(x_k) \leq V_1(x_k)$ holds for all x_k, the value function sequence V_j is a monotonically increasing sequence, $V_{j+1}(x_k) \geq V_j(x_k)$, $\forall x_k$, $\forall j \geq 0$ (Liu et al. (2017)). In our case, $V_0(x_k)$ is fixed to zero. The 3-D plot of approximate value function at $j = [1, 3, 20]$ is given in Fig. 2, this latter shows that $V_{20} > V_3 > V_1$. The convergence process of the value function is given in Fig. 3 which also affirms the monotonic behaviour of V_j.
For $x_0 = [0, 0]$ and $d_0 = 1 \times 10^{-1}$, the corresponding state trajectories are displayed in Fig. 4. The curve in green represent the trajectory of the states without taking into consideration the rate of evolution of degradation in the reward function, and the NNs are with the structures of 2–10–1 and 2–10–2, i.e., only the states of the system are used to train the NNs. In this case, the states track the reference with a null steady state error, and the value of the degradation at the final instant of time is equal to 0.4208 (Fig. 5). In order to decelerate the speed and the final value of degradation, Δd_k is integrated in the cost function, and the state of degradation d_k is considered as one of the NNs inputs, in addition to the states of the

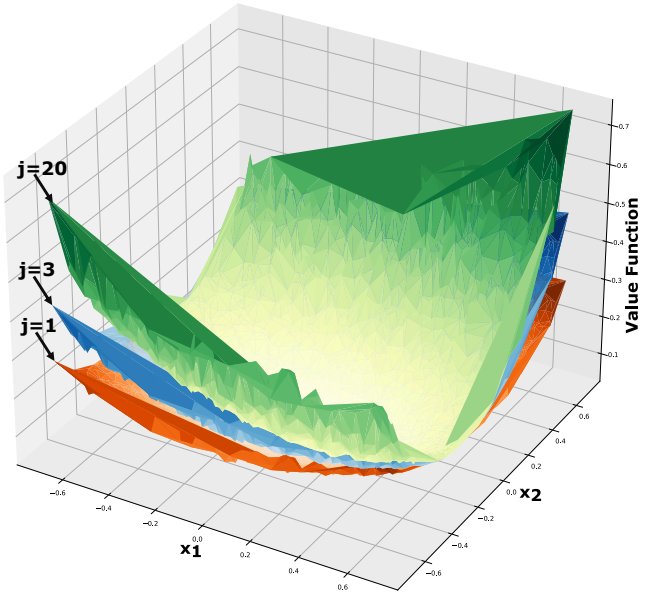

Fig. 2. 3-D plot of approximate value function (17) at $j = [1, 3, 20]$ for $Q_1 = 2 \times 10^4$

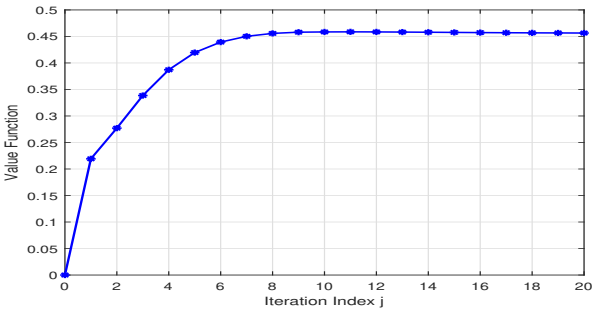

Fig. 3. Convergence of the value function for $Q_1 = 2 \times 10^4$ at $x = [0.6605, 0.5784]^T$ and $d = 0.1036$

system. For $Q_1 = 2 \times 10^4$ and $Q_1 = 2.6 \times 10^4$, the red and blue curves are respectively obtained. From Fig. 4, it can be seen that by including Δd_k and d_k in the learning and the structure of the NNs, the steady state error increases progressively by raising the value of Q_1 in order to decrease the speed of degradation.
Table 1 presents the mean square error (MSE) between the x_1 and the reference for the three cases (No delta, $Q_1 = 2 \times 10^4$ and $Q_1 = 2.6 \times 10^4$) where each period is 100 time steps.:

$$MSE = \frac{1}{N}\sum_{k=1}^{N} ||x_k - ref_k||^2$$

The Table 1 shows that for Case 1, MSE decreases progressively with respect to time which implies that the controller is focusing only on the performance of the systems ignoring the degradation. While in Case 2 and 3, the MSE increases progressively with respect to time, which confirms the results obtained in Fig. 4 and shows that the controller is trying to compromise between the performance and the speed of degradation. By increasing the value of Q_1, the controller will prioritize reducing the rate of evolution of degradation over the performance of the system (21).
Fig. 5 shows the evolution of degradation for the different

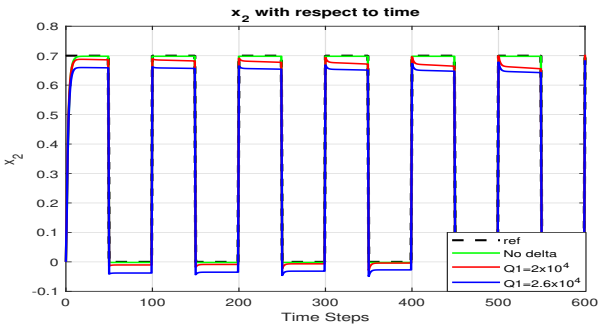

Fig. 4. Evolution of x_2 with respect to time

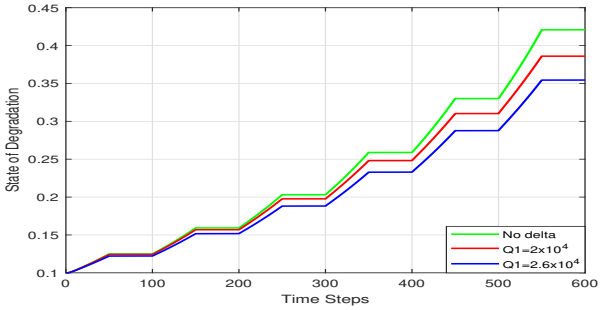

Fig. 5. Variation of degradation with respect to time

	No delta	$Q_1 = 2 \times 10^4$	$Q_1 = 2.6 \times 10^4$
N_1	0.0054	0.0056	0.0054
N_2	1.5134×10^{-9}	2.3758×10^{-6}	6.7253×10^{-5}
N_3	1.0128×10^{-9}	1.0256×10^{-5}	8.2514×10^{-5}
N_4	7.1224×10^{-10}	2.7914×10^{-5}	1.042×10^{-4}
N_5	3.1450×10^{-10}	6.1742×10^{-5}	1.3519×10^{-4}
N_6	4.9005×10^{-11}	1.2128×10^{-4}	1.7958×10^{-4}

Table 1. Mean Squared Error between x_1 and ref_k per period N

cases. At $k = 600$, the value of degradation for $Q_1 = 2 \times 10^4$ is higher than for $Q_1 = 2.6 \times 10^4$, and both values are less than the value obtained when the degradation was not considered in the cost function and the learning.

5. CONCLUSION

This paper proposes a learning approach towards the design of degradation tolerant control, for nonlinear discrete time systems affine in control. The problem is formulated over an infinite horizon in the framework of reinforcement learning. The proposed methodology is effective in learning control law that leads to reduction in degradation rate of the system whilst maintaining the closed loop stability under the assumptions of admissibility. The approach was examined for three cases. The first case is when the rate of evolution of degradation is not included in the reward function and also the degradation data are not considered as input of neural networks. In the second and third cases, the degradation is integrated in the cost function and also supposed as one of the NN's inputs in addition to the state of the system. By tuning the weighting coefficient matrix Q_1, the rate of evolution of degradation can be decreased thus preventing the system's breakdown. Future works will focus on implementing the developed algorithm in real time, so the NNs are trained online based on the outputs of the systems without need of the system's model.

REFERENCES

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

Åström, K.J. and Wittenmark, B. (1995). *Adaptive control*. Addison-Wesley.

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., and Schröder, J. (2006). *Diagnosis and fault-tolerant control*, volume 2. Springer.

Bole, B.M., Brown, D.W., Pei, H.L., Goebel, K., Tang, L., and Vachtsevanos, G. (2010). Fault adaptive control of overactuated systems using prognostic estimation. In *Annual Conference of the PHM Society*, volume 2.

Brown, D., Bole, B., and Vachtsevanos, G. (2010). A prognostics enhanced reconfigurable control architecture. 1061–1066. doi: 10.1109/MED.2010.5547651.

Brown, D.W., Georgoulas, G., Bole, B., Pei, H.L., Orchard, M., Tang, L., Saha, B., Saxena, A., Goebel, K., and Vachtsevanos, G. (2021). Prognostics enhanced reconfigurable control of electro-mechanical actuators. *Annual Conference of the PHM Society*.

Buşoniu, L., de Bruin, T., Tolić, D., Kober, J., and Palunko, I. (2018). Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46, 8–28.

Fu, J., Maré, J.C., and Fu, Y. (2017). Modelling and simulation of flight control electromechanical actuators with special focus on model architecting, multidisciplinary effects and power flows. *Chinese Journal of Aeronautics*, 30(1), 47–65.

Jha, M.S., Weber, P., Theilliol, D., Ponsart, J.C., and Maquin, D. (2019). A reinforcement learning approach to health aware control strategy. *27th Mediterranean Conference on Control and Automation (MED)*, 171–176. URL `10.1109/MED.2019.8798548`.

Kanso, S., Jha, M.S., and Theilliol, D. (2023). Degradation tolerant optimal control design for linear discrete-times systems. In *International Conference on Diagnostics of Processes and Systems*, 398–409. Springer.

Knight, J.C. (2002). Safety critical systems: challenges and directions. In *Proceedings of the 24th international conference on software engineering*, 547–550.

Lewis, F.L. and Liu, D. (2012). *Reinforcement learning and approximate dynamic programming for feedback control*.

Lewis, F.L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9, 32–50. doi: 10.1109/MCAS.2009.933854.

Liu, D., Wei, Q., Wang, D., Yang, X., and Li, H. (2017). Adaptive dynamic programming with applications in optimal control. URL `http://www.springer.com/series/1412`.

Noura, H., Theilliol, D., Ponsart, J.C., and Chamseddine, A. (2009). *Fault-tolerant control systems: Design and practical applications*. Springer Science & Business Media.

Pour, F.K., Theilliol, D., Puig, V., and Cembrano, G. (2021). Health-aware control design based on remaining useful life estimation for autonomous racing vehicle. *ISA Transactions*, 113, 196–209. doi: 10.1016/j.isatra.2020.03.032.

Salazar, J.C., Weber, P., Nejjari, F., Sarrate, R., and Theilliol, D. (2017). System reliability aware model predictive control framework. *Reliability Engineering and System Safety*, 167, 663–672. doi:10.1016/j.ress.2017.04.012.

Stengel, R.F. (1994). *Optimal control and estimation*. Courier Corporation.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement Learning: An Introduction*.

Werbos, P.J. (1997). Advanced forecasting methods for global crisis warning and models of intelligence.

Yousefi, N., Tsianikas, S., and Coit, D.W. (2020). Reinforcement learning for dynamic condition-based maintenance of a system with individually repairable components. *Quality Engineering*, 32(3), 388–408.