Reinforcement Learning Solution with Costate Approximation for a Flexible Wing Aircraft

Mohammed Abouheaf
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
Email: mohammed.abouheaf@uottawa.ca

Wail Gueaieb
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
Email: wail.gueaieb@uottawa.ca

***Abstract*—An online adaptive learning approach based on costate function approximation is developed to solve an optimal control problem in real time. The proposed approach tackles the main concerns associated with the classical Dual Heuristic Dynamic Programming techniques in uncertain dynamical environments. It employs a policy iteration paradigm along with adaptive critics to implement the adaptive learning solution. The resultant framework does not need or require prior knowledge of the system dynamics, which makes it suitable for systems with high modeling uncertainties. As a proof of concept, the suggested structure is applied for the auto-pilot control of a flexible wing aircraft with unknown dynamics which are continuously varying at each trim speed condition. Numerical simulations showed that the adaptive control technique was able to learn the system's dynamics and regulate its states as desired in a relatively short time.**

I. Introduction

The Dual Heuristic Dynamic Programming approaches employ costate function approximations to solve the Dynamic Programming problems [1]–[4]. The challenges associated with these approaches involve the necessity to know the drift dynamics of the considered systems. This work introduces an online reinforcement learning solution based on the costate function approximation. This approach does not need the drift dynamics of the systems and it is suitable to control systems that cannot be accurately modeled or identified. It is used to solve the challenging control problem of the flexible wing aircraft, where the aerodynamic model of the flexible wing varies continuously *(i.e. the drift dynamics are hard to model)*.

The optimal control problems are formulated as decision processes in the framework of Artificial Intelligence. Dynamic Programming approaches are used to solve the optimal control problems in [1]–[5]. Approximate Dynamic Programming (ADP) approaches are used to find approximate solutions for the Dynamic Programming problems in [2], [5], [6]. These approaches combine knowledge from Dynamic Programming, Reinforcement Learning (RL), and Adaptive Critics [2]–[8]. ADP approaches are used in the cooperative control, computational intelligence, decision making, and applied mathematics [9], [10]. Approximate dynamic Programming (ADP) techniques are developed to solve the optimal control problems online in [2] and offline in [11]. Reinforcement Learning (RL) approaches use different learning processes in dynamic environments [12]. These involve two-step techniques known as Value Iteration (VI) and Policy Iteration (PI) [6]. The Reinforcement Learning solutions are implemented using actor-critic neural networks [6]. Actor-critic neural networks are some forms of Temporal Difference (TD) methods with separate learning structures [13]. The actor approximates the optimal decision and applies it to the dynamic environment and then the quality of this decision is assessed and approximated using a critic structure [6]. Following this assessment, the actor and critic weights are updated [12], [13]. The adaptive critics are used to solve the optimal control problem in real-time in [14]. The optimal control problem finds the necessity optimality conditions and hence the optimal strategies [15]. Reinforcement Learning is used to solve the synchronization control problem online in [16]–[18].

The weight-shift control problem of the flexible wing aircraft depends on the dynamic coupling between the pilot system and the wing system, when the pilot system changes its center of gravity relative to the wing's center of gravity, this creates a pitch/roll control system that uses that shift in the centers of gravity [19]. The observable stability and control features of the flexible wing aircraft are studied using the longitudinal static stability concepts of the conventional aeroplane in [20], [21]. The lateral directional stability margins of the flexible wing aircraft are explained and shown to be larger compared to the conventional aeroplane [19].

The paper is organized as follows: Section 2 demonstrates the formulation of the optimal control problem. Section 3 discusses the challenges associated with implementing the classical Dual Heuristic Programming solutions. Section 4 introduces the proposed adaptive learning solution along with its implementation using the adaptive critics. Section 5 highlights the challenging weight-shift control problem of the flexible wing aircraft along with the simulation results.

II. The Optimal Control Problem

This section highlights the formulation of the optimal control problem for dynamical systems. The optimality conditions are found using the classical Bellman and Hamiltonian dynamics. This mathematical framework is used to motivate the hence-coming reinforcement learning solution.

A. Bellman Equation Formulation

The dynamical equation of the system is given by

$$X_{\ell+1} = A\,X_\ell + B\,u_\ell, \tag{1}$$

where $X_\ell \in R^n$ and $u_\ell \in R^m$ are the states and the control signals respectively, A and B are the drift and the control input matrices respectively, and ℓ is the time-index.

A quadratic performance index P is introduced to assess the quality of the taken control actions such that

$$P = \sum_{\ell=0}^{\infty} U(X_\ell, u_\ell), \tag{2}$$

where U is the cost function and it is given by

$$U(X_\ell, u_\ell) = \frac{1}{2}\left(X_\ell^T\,Q\,X_\ell + u_\ell^T\,R\,u_\ell\right), \tag{3}$$

with $Q \geq 0 \in R^{n\times n}$ and $R > 0 \in R^{m\times m}$ being symmetric weighting matrices.

Let the value function $S(X_\ell)$ be the solution of (2) so that

$$S(X_\ell) = \sum_{i=\ell}^{\infty} U(X_i, u_i). \tag{4}$$

This yields the following Bellman equation

$$S(X_\ell) = \frac{1}{2}\left(X_\ell^T\,Q\,X_\ell + u_\ell^T\,R\,u_\ell\right) + S(X_{\ell+1}). \tag{5}$$

The goal of the optimization problem is to find the optimal value function $S^o(X_\ell)$ and the optimal policy u_ℓ^o to minimize the cost-to-go function (4) or (5). The Bellman's optimality principles yield

$$S^o(X_\ell) = \text{argmin}\underbrace{S(X_\ell)}_{u_\ell}. \tag{6}$$

Thus, the optimal control policy u_ℓ^o is given by

$$u_\ell^o = -R^{-1}\,B^T\,\nabla S^o(X_{\ell+1}), \tag{7}$$

where $\nabla S^o(X_\ell) = \partial S^o(X_\ell)/\partial X_\ell$.

B. Hamiltonian Dynamics

The necessity optimal conditions are derived using means of Lagrange dynamics [15]. The optimization problem generates an optimal set of control actions which minimizes the cost-to-go function (4).

The constraint of the optimization problem $\Psi(X_\ell, \mu_\ell)$ given arbitrary policy μ_ℓ is the system's trajectory at each time-index ℓ such that

$$X_{(\ell+1)} \equiv \Psi\left(X_\ell, \mu_\ell\right). \tag{8}$$

This constraint is associated with a Lagrange multiplier or a costate variable $\theta_{\ell+1}$. Thus, an augmented performance index for the dynamical system (1) on the time interval $[\ell, L]$ is defined so that [15]

$$P^{'} = \Phi(L, X_L) + \sum_{k=l}^{L-1} U(X_k, \mu_k) = \Phi(L, X_L) + \sum_{k=l}^{L-1}\Big(U(X_k, \mu_k) + \theta_{k+1}^{\mathrm{T}}(\Psi(X_k, \mu_k) - X_{(k+1)})\Big), \tag{9}$$

where $\Phi(L, X_L)$ is the boundary function at the final time L.

The Hamiltonian for the dynamical system (1) is given by

$$H(X_k, \theta_{(k+1)}, \mu_k) = \quad \theta_{(k+1)}^{\mathrm{T}} X_{(k+1)} + U(X_k, \mu_k). \tag{10}$$

The performance $P^{'}$ can be rearranged such that

$$P^{'} = \Phi(L, X_L) - \theta_L^{\mathrm{T}} X_L + H(X_l, \theta_{(l+1)}, \mu_l) + \sum_{k=l+1}^{L-1}\Big(H(X_k, \theta_{(k+1)}, \mu_k) - \theta_k^{\mathrm{T}} X_k\Big). \tag{11}$$

The augmented variations in $P^{'}$ are given by

$$\begin{aligned}\partial P^{'} = {} & \Big(\partial\Phi(L, X_L)/\partial X_L - \theta_L\Big)^{\mathrm{T}}\partial X_L \\ & + \big(\partial H\big(X_l, \theta_{(l+1)}, \mu_l\big)/\partial X_l\big)^{\mathrm{T}}\partial X_l \\ & + \big(\partial H\big(X_l, \theta_{(\ell+1)}, \mu_\ell\big)/\partial \mu_\ell\big)^{\mathrm{T}}\partial \mu_\ell \\ & + \sum_{k=l+1}^{L-1}\Bigg(\Big(\partial H\big(X_k, \theta_{(k+1)}, \mu_k\big)/\partial X_k - \theta_k\Big)^{\mathrm{T}}\partial X_k \\ & + \big(\partial H\big(X_k, \theta_{(k+1)}, \mu_k\big)/\partial \mu_k\big)^{\mathrm{T}}\partial \mu_k\Bigg) \\ & + \sum_{k=l+1}^{L}\Big(\partial H\big(X_{(k-1)}, \theta_k, \mu_{(k-1)}\big)/\partial \theta_k - X_k\Big)^{\mathrm{T}}\partial \theta_k,\end{aligned} \tag{12}$$

where $\partial X, \partial\mu$, and $\partial\theta$ are the variations in the states, control inputs, and costate variable respectively.

The constrained minimum conditions yield

$$X_{(\ell+1)} = \partial H\big(X_\ell, \theta_{(\ell+1)}, u_\ell\big)/\partial\theta_{(\ell+1)}, \tag{13}$$

$$\theta_\ell = \partial H\big(X_\ell, \theta_{(\ell+1)}, u_\ell\big)/\partial X_\ell, \tag{14}$$

$$\partial H\big(X_\ell, \theta_{(\ell+1)}, u_\ell\big)/\partial u_\ell = 0. \tag{15}$$

Equation (14) shows the coupled dynamical behavior of the Lagrange multiplier θ_k and the dynamics $X_{(k+1)}$ leading to the costate equation. The stationarity condition is determined by (15). Thus, the optimal policy is given by

$$u^o = -R^{-1}\,B^T\,\theta_{\ell+1}, \tag{16}$$

The associated boundary conditions are given by

$$\begin{aligned}&\big(\partial H\big(X_\ell, \theta_{(\ell+1)}, u_\ell\big)/\partial X_\ell\big)^{\mathrm{T}}\partial X_\ell = 0, \\ &\big(\partial\Phi(L, X_L)/\partial X_L - \theta_L\big)^{\mathrm{T}}\partial X_L = 0.\end{aligned} \tag{17}$$

The first boundary condition holds as the initial conditions are fixed. On the other hand, for free-final state X_L, the following condition needs to be satisfied

$$\theta_L = \partial\Phi(L, X_L)/\partial X_L. \tag{18}$$

The conditions (13), (14), and (15) are used to propose the RL solution framework as will be explained in the next section.

III. Dual Heuristic Programming: Challenges

The relation between the Hamiltonian function (10) and Bellman equation (5) is of great importance to develop Reinforcement Learning solutions based on the costate function approximations. This section highlights the challenges associated with implementing the Dual Heuristic Programming (DHP) solutions which use value function approximations.

A. Hamilton-Jacobi Equation

The Hamilton-Jacobi (HJ) equation relates the Hamiltonian function (10) and Bellman equation (5) such that

$$S(X_\ell) - S(X_{\ell+1}) = H(X_\ell, \theta_{(\ell+1)}, \mu_\ell) - \theta_{(\ell+1)}^{\mathrm{T}} X_{(\ell+1)}, \quad (19)$$

where the costate variable θ_ℓ is given in terms of the value function $S(X_\ell)$ such that $\theta_\ell = \partial S(X_\ell)/\partial X_\ell$.

Using the optimal action (16) or equivalently (7) yields the Hamilton-Jacobi-Bellman (HJB) equation such that

$$H(X_\ell, \theta_{(\ell+1)}, u_\ell^o) = 0. \quad (20)$$

Thus, the costate equation expressions based on Bellman equation and the Hamiltonian function are given as follows

$$\begin{aligned} \theta_\ell = & \quad A^{\mathrm{T}} \theta_{\ell+1} + Q\, X_\ell, \\ \nabla S(X_\ell) = & \quad A^{\mathrm{T}} \nabla S(X_{\ell+1}) + Q\, X_\ell. \end{aligned} \quad (21)$$

B. RL Solutions Using the Costate Equations

The costate equations (21) are equivalent as implied by the Hamilton-Jacobi (HJ) equation (19) and the Hamilton-Jacobi-Bellman (HJB) equation (20). Reinforcement Learning solutions based on the costate equations (21) are implemented using value and policy iteration techniques listed in Algorithms 1 and 2, respectively.

Algorithm 1: Online Value Iteration Solution (DHP)

1) **Initialization:** *costate* $\nabla S^0(X_\ell)$ *and the action* μ_k^0.
2) **Evaluation:** $\nabla S^{r+1}(..)$

$$\nabla S^{r+1}(X_\ell) = Q\, X_\ell^r + A^{\mathrm{T}} \nabla S^r(X_{\ell+1}), \quad (22)$$

where r *is the iteration index.*
3) **Improving The Policy:**

$$\mu_\ell^{r+1} = -R^{-1} B^{\mathrm{T}} \nabla S^{r+1}(X_{\ell+1}). \quad (23)$$

4) **Termination Criteria:** $\|\nabla S^{r+1}(..) - \nabla S^r(..)\| \leq \varepsilon$, ε is a user defined error threshold.

Algorithm 2: Online Policy Iteration Solution

1) **Initialization:** *costate* $\nabla S^0(X_\ell)$ *and use admissible action* μ_k^0.
2) **Evaluation:** $\nabla S^r(..)$

$$\nabla S^r(X_\ell) = Q\, X_\ell^r + A^{\mathrm{T}} \nabla S^r(X_{\ell+1}), \quad (24)$$

3) **Improving The Policy:**

$$\mu_\ell^{r+1} = -R^{-1} B^{\mathrm{T}} \nabla S^r(X_{\ell+1}). \quad (25)$$

4) **Termination Criteria:** $\|\nabla S^{r+1}(..) - \nabla S^r(..)\| \leq \varepsilon$.

Algorithms 1 and 2 suffer from two main drawbacks. First, the costate equations (22) and (24) depend on the drift dynamics of the system. This means that, the states of the dynamical system need to be physically accessible or identifiable, which may not be possible in some cases, especially with systems of model uncertainties. Second, it is difficult to evaluate S^r using Algorithm 2. This motivates the hence-coming development that will handle these two main drawbacks.

IV. Costate Variable Approximation

The following development motivates the solution of the costate equations or equivalently the costate variable using means of policy iteration without knowing the full dynamics of the system. Indeed, the drift dynamics are not required. The proposed real-time solution uses the HJB equation (20) and it is implemented using means of adaptive critics.

A. Novel Policy Iteration Costate Solution

The Hamilton-Jacobi-Bellman (HJB) expression (20) is used to propose an alternative solution as follows

Algorithm 3: Online Policy Iteration Solution

1) **Initialization:** *costate* $\nabla S^0(X_\ell)$ *and use admissible action* μ_k^0.
2) **Evaluation:** $\nabla S^r(..)$

$$\nabla S^{r\,\mathrm{T}}(X_{(k+1)})\, X_{(\ell+1)}^r = -\frac{1}{2} U(X_\ell^r, \mu_\ell^r), \quad (26)$$

3) **Improving The Policy:**

$$\mu_\ell^{r+1} = -R^{-1} B^{\mathrm{T}} \nabla S^r(X_{\ell+1}). \quad (27)$$

4) **Termination Criteria:** $\|\nabla S^{r+1}(..) - \nabla S^r(..)\| \leq \varepsilon$.

This formulation for the costate variable's approximation enables the use of policy iteration, where the term $\nabla S^{r\,\mathrm{T}}(X_{(k+1)})\, X_{(\ell+1)}^r$ is approximated using means of actor-critic neural networks. Moreover, as per (26), the evaluation of the costate function does not need the drift dynamics, which makes Algorithm 3 more effective than Algorithms 1 and 2.

B. Actor-Critic Neural Networks Implementation

The policy iteration Algorithm 3 operates in a dynamic environment, where the system learns the optimal value function $S^o(X_\ell)$ and the optimal policy μ_ℓ^o in real-time. The actor-critic neural networks are used to implement the solution in real-time. The critic neural network structure is used to approximate the costate variable $\nabla S(..)$. The actor neural structure is used to approximate the optimal control policy (27). The critic component assesses the quality of the taken actions in a dynamic environment.

The value function $S(X_\ell)$ is parameterized by some weight matrix ω_c such that

$$\hat{S}(.|\omega_c) = \frac{1}{2} X_\ell^{\mathrm{T}}\, \omega_c^{\mathrm{T}}\, X_\ell.$$

Thus, the gradient $\nabla S^r(\dots)$ is approximated by

$$\nabla \hat{S}(.|\omega_c) = \omega_c^{\mathrm{T}}\, X_\ell.$$

The action is also parameterized by some weight vector ω_a so that it can be approximated by an actor structure, such that

$$\hat{\mu} = \omega_a^{\mathrm{T}}\, X_\ell.$$

The Hamilton-Jacobi-Bellman (HJB) equation (20) relates the costate approximation to its target value such that

$$-\bar{\omega}_c^T\, \bar{X}_{(\ell+1)} = \frac{1}{2}(X_\ell^{\mathrm{T}} Q X_\ell + \hat{\mu}_\ell^{\mathrm{T}} R \hat{\mu}_\ell), \tag{28}$$

where $\bar{\omega}_c$ and $\bar{X}_{\ell+1}$ are the vector transformations of the critic weights matrix ω_c and its associated states' vector X_ℓ respectively. The entries of the vector $\bar{\omega}_c$ are the upper-triangle elements of the matrix ω_c and the entries of the vector $\bar{X}_{\ell+1}$ are the self and cross product of the entries in the states' vector $\bar{X}_\ell$ in the same order which the weights in $\bar{\omega}_c$ are listed.

Similarly, the optimal control policy relates the action approximation to its target value such that

$$\hat{\mu}_\ell = -\,R^{-1}\, B^{\mathrm{T}}\, \nabla \hat{S}(X_{\ell+1}). \tag{29}$$

The actor and the critic weights are tuned using the following gradient descent approach

$$\omega_a^{(r+1)\mathrm{T}} = \omega_a^{r\mathrm{T}} - \rho_a \left(\left(-\,R^{-1}\, B^{\mathrm{T}}\, \nabla \hat{S}(X_{\ell+1}^r) \right) - \omega_a^{r\mathrm{T}} X_\ell^r \right) X_\ell^{r\,\mathrm{T}}, \tag{30}$$

$$\begin{aligned} \bar{\omega}_c^{(r+1)\mathrm{T}} &= \bar{\omega}_c^{r\mathrm{T}} - \rho_c \left(\Im \left(-\frac{1}{2}(X_\ell^{r\mathrm{T}} Q X_\ell^r + \hat{\mu}_\ell^{r\mathrm{T}} R \hat{\mu}_\ell^r) \right) \right. \\ &- \left. \bar{\omega}_c^{r\mathrm{T}} \Im \left(\bar{X}_{(\ell+1)}^r \right) \right) \Im \left(\bar{X}_{(\ell+1)}^{r\mathrm{T}} \right), \end{aligned} \tag{31}$$

where $\Im(..)$ is an operator that concatenates $n \times (n+1)/2$ terms of the lower (or upper) triangular part of its associated symmetric matrix. ρ_a and ρ_c are the actor and critic learning rates, respectively.

The actor-critic neural network implementation is outlined in Algorithm 4.

Algorithm 4: Actor-Critic Implementation

1) Initialize the actor and critic weights ω_a^0 and ω_c^0.
2) *Outer loop (ℓ iterations)*
   a) Initialize the states X_0^0.
      Inner loop (r iterations)
      - Update the critic weights $\omega_c^r = \omega_c^\ell$.
      - Evaluate $X_{(\ell+1)}^r$ and $\hat{\mu}^r$ using (1) and (29).

      End Inner Loop when $r = n \times (n+1)/2$.
   b) Update the actor and critic weights using (30) and (31).
3) *End outer loop when* $\left\| \hat{S}^{\ell+1}(\dots) - \hat{S}^{\ell}(\dots) \right\| \leq \varepsilon$.

V. Numerical Simulation

The developed algorithm is tested on the challenging problem of controlling a flexible wing aircraft. The high nonlinearity and the continuous deformations in the wing make the control problem of the aircraft difficult to solve using the classical DHP approaches. This is due to the lack of rigorous dynamical models for the flexible wing aircraft. Herein, a model based on experimental and theoretical setups is used to validate the performance of the proposed approach at a trim speed condition [19], [22].

A. Flexible Wing Aircraft Dynamics

The aircraft is a two-body (pilot-wing) system with kinematic constraints [19]. The pilot controls the longitudinal and lateral motions by steering the aircraft's force bar. This creates a relative motion between the wing's and the pilot's centers of gravity, which drives the aircraft. The side and top views for the aircraft's moving frames are shown in Figure 1.

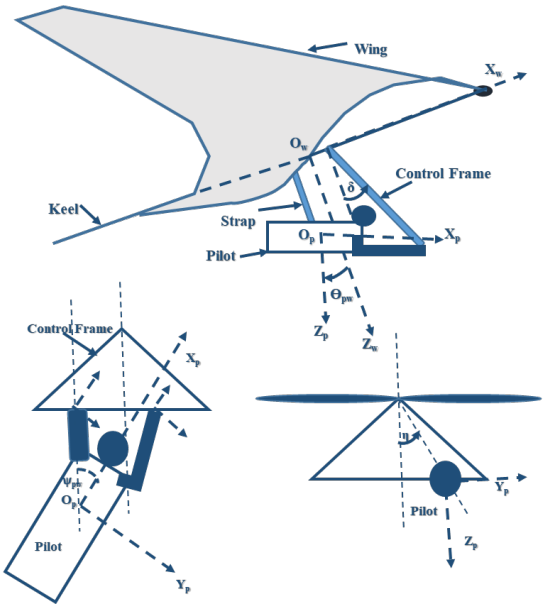

Fig. 1. Longitudinal and Lateral Geometry.

The control problem is divided into two decoupled control problems (longitudinal and lateral control problems). The shift in the center of the gravity is achieved by controlling the longitudinal angle δ and the lateral angle η as shown in Figure 1. The longitudinal control angle δ (is the angle between the hang strap and a line parallel to $O_w Z_w$ at the hang point in the longitudinal frame). The lateral control angle η (is the angle between the hang strap and a line parallel to $O_w Z_w$ at the hang point in the lateral frame).

In the sequel, the linearized small perturbation equations of a semi-experimental study are considered to create the system's measurements (i.e. the values of X_ℓ) [19], [22]. The longitudinal and lateral dynamics are expressed as follows

$$\begin{aligned} X_{(\ell+1)}^{Lo} &= A_d^{Lo}\, X_\ell^{Lo} + B_d^{Lo}\, \mu_\ell^{Lo}, \\ X_{(\ell+1)}^{La} &= A_d^{La}\, X_\ell^{La} + B_d^{La}\, \mu_\ell^{La}, \end{aligned} \tag{32}$$

where the longitudinal and lateral states are $X^{Lo} = \left[\, u_w^T \;\; w_w^T \;\; q_w^T \;\; \theta_w^T \,\right]^T$ and $X^{La} = \left[\, v_w^T \;\; p_w^T \;\; r_w^T \;\; \phi_w^T \;\; \psi_w^T \,\right]^T$ respectively. The wing's attitude angles (roll, pitch, and yaw) are denoted by $\phi_w, \theta_w,$ and ψ_w respectively. The rotational velocities of the wing (roll, pitch, and yaw rates) are denoted by p_w, q_w, and r_w respectively. The transnational velocities of the wing (forward, lateral, and normal velocities) are denoted by u_w, v_w, and w_w respectively. The longitudinal and lateral

control input signals are denoted by $\mu^{Lo} = \delta$ and $\mu^{La} = \eta$ respectively. $A_d^{Lo}, A_d^{La}, B_d^{Lo}, B_d^{La}$ are the longitudinal and lateral drift and control input matrices respectively.

The drift and control input matrices are given as follows

$$A_d^{Lo} = \begin{bmatrix} 0.9982 & 0.0065 & 0.0012 & -0.0971 \\ -0.0139 & 0.9774 & 0.1055 & 0.0136 \\ 0.0027 & -0.0043 & 0.9858 & -0.0002 \\ 0.0000 & -0.0000 & 0.0099 & 1.0000 \end{bmatrix},$$

$$A_d^{La} = \begin{bmatrix} 0.9977 & -0.0028 & -0.1069 & 0.0971 & -0.0131 \\ -0.0131 & 0.8092 & 0.0677 & -0.0007 & 0.0001 \\ 0.0026 & 0.0332 & 0.9802 & 0.0001 & -0.0000 \\ -0.0001 & 0.0090 & 0.0004 & 1.0000 & 0.0000 \\ 0.0000 & 0.0002 & 0.0099 & 0.0000 & 1.0000 \end{bmatrix},$$

$$B_d^{Lo} = \begin{bmatrix} 0.0000 \\ 0.0040 \\ 0.0741 \\ 0.0004 \end{bmatrix}, B_d^{La} = \begin{bmatrix} 0.0002 \\ 0.0324 \\ -0.0036 \\ 0.0002 \\ -0.0000 \end{bmatrix}.$$

B. Simulation Results

The proposed algorithm is used to design an online controller for a flexible wing aircraft. The adaptive learning paradigm in *Algorithm 4* is adopted to control the longitudinal and lateral motions of the system described by (32). The algorithm decides on the optimal control strategy and its respective optimal value function in real-time. This is realized using the actor-critic neural network approximations through a gradient descent tuning approach.

The simulation outcomes involve the actor-critic tuning process, evaluation of the aircraft's asymptotic stability, and the evolution of the closed-loop eigenvalues during the learning process. These performance measures assess the effectiveness of the proposed algorithm.

The learning and simulation parameters are: $\rho_a = \rho_c = 0.01, Q = I_{2\times 2}, R = 0.1, \varepsilon = 10^{-12}$ and sampling time $Ts = 0.01$. The initial states for the lateral and longitudinal motions are given by $X_0^{La} = \begin{bmatrix} 3, & 2, & 0.1571, & 0.1571, & 0.1571 \end{bmatrix}$ and $X_0^{Lo} = \begin{bmatrix} 2, & 1, & 0.1571, & 0.1571 \end{bmatrix}$, respectively.

The online tuning of the actor-critic weights for the longitudinal and lateral motions is shown to converge, as illustrated by Figure 2. The actor-critic weights converge in approximately 10 iterations of the inner loop of *(Algorithm 4)*. This reveals how the gradient descent technique succeeded in tuning the weights towards a convergent region. The dynamics and control input signals are shown to be asymptotically stable, as illustrated in Figure 3. The longitudinal and lateral states decay to zero as well as the input control angles. Figure 4 demonstrates the powerful properties of this adaptive learning technique. It shows the evolution of the closed-loop poles during the learning process. The algorithm quickly overpasses the system's sluggish modes and asymptotically stabilizes the originally unstable modes of the aircraft. It rapidly steers the initially unstable poles of the system towards the inside of the unit circle (stability region) after a few learning iterations.

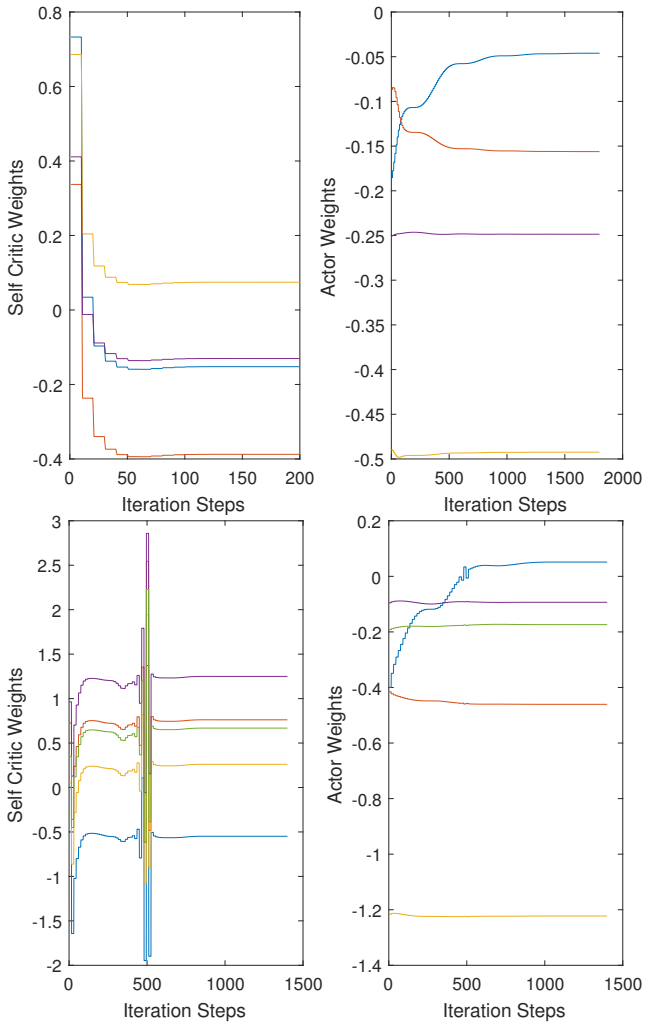

Fig. 2. Longitudinal and Lateral Actor-Critic Tuning.

VI. Conclusion

The paper introduced a novel online adaptive learning approach that combines costate variable approximation and policy iteration techniques to solve an optimal control problem for dynamic systems. It is implemented in real-time using means of adaptive critics and neural networks. This approach does not need to know the drift dynamics of the considered systems, which makes it suitable for non-identifiable or physically inaccessible systems. This represents a major advantage in comparison to the classical Dual Heuristic Dynamic Programming approaches. The proposed adaptive learning structure is employed to solve the challenging control problem of a flexible wing aircraft. Simulation results showed satisfactory convergence and transient characteristics of the proposed control scheme.

References

[1] R. Howard, *Dynamic Programming and Markov Processes*, ser. Four volumes. Cambridge. MA: MIT Press, 1960.

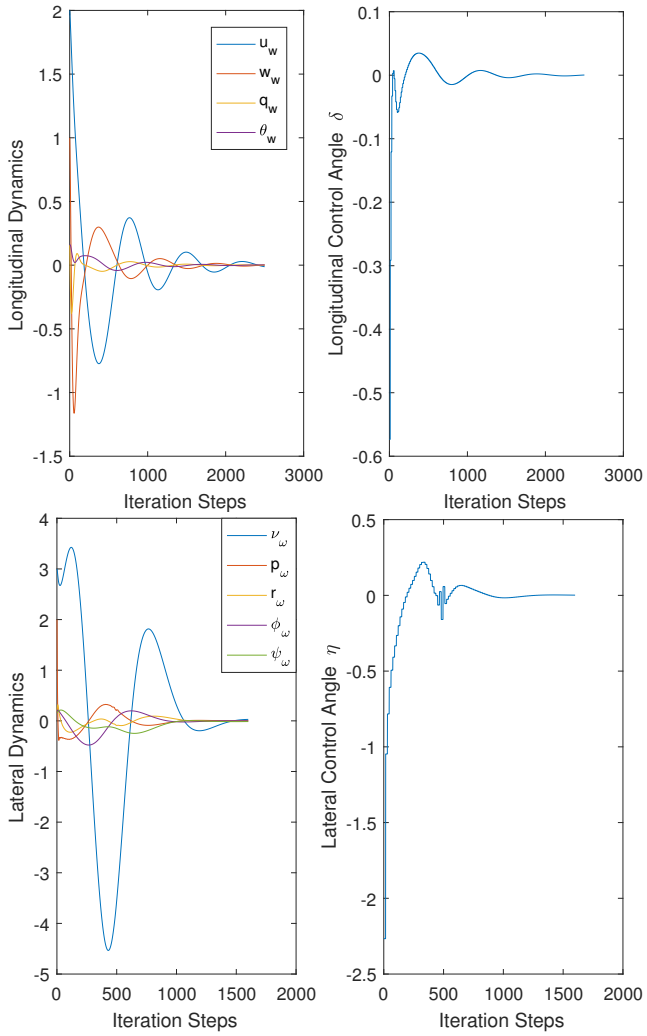

Fig. 3. Longitudinal and Lateral Dynamics and Control Signals (angles are in rad).

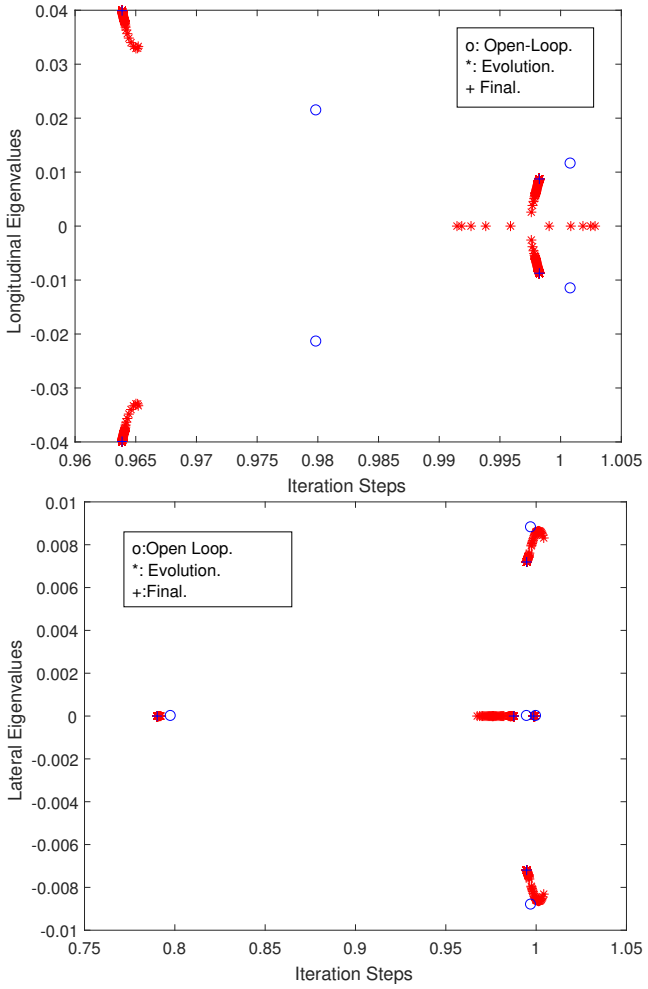

Fig. 4. Longitudinal and Lateral Eigenvalues Evolution During Learning.

[2] P. Werbos, *Approximate dynamic programming for real-time control and neural modeling. Handbook of Intelligent Control*. New York: Van Nostrand Reinhold: D.A. White and D.A. Sofge, 1992.

[3] M. Abouheaf and F. Lewis, *Dynamic Graphical Games:Online Adaptive Learning Solutions Using Approximate Dynamic Programming*. World Scientific, 2014, ch. Chapter 1, pp. 1–48.

[4] M. I. Abouheaf and F. L. Lewis, "Approximate dynamic programming solutions of multi-agent graphical games using actor-critic network structures," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug 2013, pp. 1–8.

[5] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences*. Ph.D. Thesis, 1974.

[6] A. Barto, *Reinforcement learning: An introduction*. Massachusetts: MIT Press, 1998.

[7] P. Werbos, "A menu of designs for reinforcement learning over time," *Neural networks for control*, pp. 67–95, 1990.

[8] M. Abouheaf, F. Lewis, M. Mahmoud, and D. Mikulski, "Discrete-time dynamic graphical games: model-free reinforcement learning solution," *Control Theory and Technology*, vol. 13, no. 1, pp. 55–69, 2015.

[9] J. Si, A. Barto, W. Powell, and I. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[10] D. Prokhorov and D. Wunsch, "Adaptive critic designs," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 997–1007, 1997.

[11] D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic programming*. Massachusetts: Athena Scientific, 1996.

[12] R. Sutton and A. Barto, *Reinforcement learning–an introduction*. Cambridge: MIT Press, 1998.

[13] B. Widrow, N. Gupta, and S. Maitra, "Punish/reward: Learning with a critic in adaptive threshold systems," *IEEE Tran. Sys. Man Cyb.*, vol. 3, no. 5, pp. 455–465, 1973.

[14] P. Werbos, "Neural networks for control and system identification," *IEEE Proc. CDC*, pp. 260–265, 1989.

[15] F. Lewis, D. Vrabie, and V. Syrmos, *Optimal Control*, ser. Third Edition. New York: John Wiley, 2012.

[16] M. Abouheaf, F. Lewis, K. Vamvoudakis, S. Haesaert, and R. Babuska, "Multi-agent discrete-time graphical games and reinforcement learning solutions," *Automatica*, vol. 50, no. 12, pp. 3038–3053, 2014.

[17] M. Abouheaf and M. Mahmoud, "Policy iteration and coupled riccati solutions for dynamic graphical games," *Int. J. of Digital Sig. and Smrt Sys.*, vol. 1, no. 2, pp. 143–162, 2017.

[18] M. Abouheaf and W. Gueaieb, "Multi-agent reinforcement learning approach based on reduced value function approximations," *2017 IEEE Int. Symp. on Rob. and Intell. Sens.*, pp. 111–116, 2017.

[19] M. Cook and M. Spottiswoode, "Modelling the flight dynamics of the hang glider," *J. Aeronau.*, vol. 109, no. 1102, pp. I–XX, 2006.

[20] D. Blake, *Modelling the aerodynamics, stability and control of the hang glider*. MSc Thesis, College of Aeron., Cranfield Inst. of Tech., 1991.

[21] R. Rollins, *Study of Experimental Data to Assess the Longitudinal Stability and Control of the Hang Glider*. MSc thesis, College of Aeronautics, Cranfield University, 2000.

[22] E. Kilkenny, *Full scale wind tunnel tests on hang glider pilots*. College of Aeronautics Report 8416, Cranfield Institute of Technology, 1984.