International Journal of Intelligent Robotics and Applications (2020) 4:21–43
https://doi.org/10.1007/s41315-019-00105-3

REGULAR PAPER

Online model-free controller for flexible wing aircraft: a policy iteration-based reinforcement learning approach

Mohammed Abouheaf[1,2] · **Wail Gueaieb**[1]

Received: 28 May 2019 / Accepted: 9 October 2019 / Published online: 18 October 2019

Abstract
The aerodynamic model of flexible wing aircraft is highly nonlinear with continuously time-varying dynamics under kinematic constraints. The nonlinearities stem from the aerodynamic forces and continuous deformations in the flexible wing. In spite of the various experimental attempts and theoretical setups that were made to model these dynamics, an accurate formulation was not achieved. The control paradigms of the aircraft are concerned with the electro-mechanical coupling between the pilot and the wing. It is challenging to design a flight controller for such aircraft while complying with these constraints. In this paper, innovative machine learning technique is employed to design a robust online model-free control scheme for flexible wing aircraft. The controller maintains internal asymptotic stability for the aircraft in real-time using selected set of measurements or states in uncertain dynamical environment. It intelligently incorporates the varying dynamics, geometric parameters, and physical constraints of the aircraft into optimal control strategies. The adaptive learning structure employs a policy iteration approach, taking advantage of Bellman optimality principles, to converge to an optimal control solution for the problem. Artificial neural networks are adopted to implement the adaptive learning algorithm in real-time without prior knowledge of the aerodynamic model of the aircraft. The control scheme is generalized and shown to function effectively for different pilot/wing control mechanisms. It also demonstrated its ability to overcome the undesired stability problems caused by coupling the pilot's dynamics with the flexible wing's frame of motion.

Keywords Flexible wing aircraft · Optimal control · Reinforcement learning · Policy iteration · Adaptive critics

1 Introduction

A hang glider functions as a two-body system operating under kinematic constraints. The pilot system maneuvers the aircraft by sliding its center of gravity relative to that of the wing. The continuous movement of center of gravity throughout the flight makes modeling the dynamics of such aircraft quite challenging. The aerodynamic modeling of this flexible wing aircraft has been under investigation for more than three decades, where little research investigated the stability and control for the hang gliders. This together with the increasing potential of using flexible wing aircraft as unmanned aerial vehicle (UAV), thanks to its numerous advantages over fixed-wing UAVs, urged us to develop robust control schemes for this type of systems. An adaptive learning scheme that uses innovative machine learning technique is developed to control the flight of the aircraft in real-time. The controller has the ability to learn online the uncertainties in the vehicle's aerodynamic model. Moreover, it improves the time-response characteristics of the system. Two control approaches are considered. The first makes use of the attitude angles as the control signals. The second approach bases its actions on the force applied on the control bar as the control signals.

An aerodynamic model of a hang glider is typically characterized by highly nonlinear terms due to the continuous variations in the flight trim conditions. For example, it is well known that the wing undergoes continuous

This work was partially supported by Ontario Centers of Excellence (OCE).

✉ Mohammed Abouheaf
mabouhea@uottawa.ca

Wail Gueaieb
wgueaieb@uottawa.ca

[1] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada

[2] College of Energy Engineering, Aswan University, Aswan, Egypt

deformations throughout the flight. The aerodynamic modeling, control, stability, and safety of hang gliders were investigated by several researchers at Cranfield university in the United Kingdom in collaboration with the British Hang Gliding Association (BHGA) (Kilkenny 1986, 1984; Cook and Kilkenny 1986). Small-perturbation longitudinal and lateral directional dynamical models are developed and used to estimate the hang gliders' dynamics (Kroo 1983). It is shown that the wing's flexibility and the apparent mass together contribute to the aerodynamic effects of the hang glider (Kroo 1983). A small perturbation aerodynamic model based on a rigid wing is proposed in de Matteis (1990, 1991), where the aerodynamic derivatives are incorporated into the velocity in a separate step. However, the model does not consider the internal forces between the pilot and the wing. The wing camber and twist variations are analyzed in order to understand the nonlinear aerodynamics of the wing in Powton (1995). A mobile test rig is developed to measure the aerodynamic properties of the hang glider's wings (Kilkenny 1983). Full scale wind tunnel tests of hang gliders are performed in Kilkenny (1984). Nonlinear equations of motion for hang gliders are derived in Cook and Spottiswoode (2005), where a semi-empirical model that considers the camber and twist dependencies of the flexible wing aerodynamics is considered. This model refers the aerodynamic forces and moments to the combined center of gravity of the pilot/wing sub-systems. A comprehensive decoupled directional modeling is introduced in Spottiswoode (2001) and Cook (2013). Small-perturbation equations of motion for the hang glider are developed in Ochi (2015). In this study, the pilot actions on the control bar and the forces at the hang point are considered as external forces for the separate pilot and wing systems. Then, the hang point forces are eliminated to form the suggested aerodynamic model.

The control mechanism of the hang glider depends on the coupled motions of the pilot/wing systems, which create the desired pitch/roll maneuvers. The control of the hang glider is shown to be directly dependent on the relative positions of the pilot/wing centers of gravity, which originate from the pilot steering movements (Cook and Spottiswoode 2005; Ochi 2017; Abouheaf and Gueaieb 2018). It is demonstrated that at a high angle of attack, lowering the center of gravity increases the pilot moment ratio, and hence increases the static pitch stability (Kroo 1983). The principles of static stability of fixed wing aircraft are applied to understand the control properties and observable stability of such systems in Blake (1991), Cook (1994) and Rollins (2000). A study about the relationship between the nonlinearities in the flexible wing aerodynamic models (namely the wing camber and twist variations) and the pitching moment is introduced in Powton (1995). The flexible wing's lateral directional stability margins are found to be larger than those of the fixed wing aircraft (Cook and Spottiswoode 2005).

The optimal control problems are formulated using the dynamic programming framework (Howard 1960; Webros 1992; Abouheaf and Mahmoud 2016; Abouheaf and Gueaieb 2018), where they are solved using the Approximate Dynamic Problem (ADP) approaches to avoid the curse of dimensionality in the state and action spaces (Webros 1992; Bertsekas and Tsitsiklis 1995). These approaches are classified into four main types based on the solving value function and policy structures (Sutton and Barto 1998; Abouheaf and Lewis 2013; Abouheaf et al. 2014). Considerable classes of optimal control problems for dynamic systems use machine learning frameworks (Abouheaf et al. 2017; Abouheaf and Gueaieb 2017). They bring together optimal control theory, adaptive critics, and Reinforcement Learning (RL) to design solution platforms for the control problems (Bertsekas and Tsitsiklis 1995; Webros 1990; Sutton and Barto 1998; Abouheaf et al. 2017). The adaptive learning approaches decide on the optimal control strategies and the solving value functions through assessing their interactions in a dynamic environment in order to minimize an objective cost function (Bertsekas and Tsitsiklis 1995; Sutton and Barto 1998). The control problems are solved in real-time using RL algorithms that are based on policy or value iteration methods, where actor-critic neural networks are employed to implement these two-step solution mechanisms (Sutton and Barto 1998).

The main contribution of this work is the development of an adaptive model-free learning mechanism to control the decoupled motions of a flexible wing aircraft. To this end, an online policy iteration-based reinforcement learning approach is designed. The proposed controller is shown to be robust to the uncertainties in the system's coupled dynamics. Also, it is demonstrated to improve the closed-loop time response characteristics of aircraft, and to asymptotically stabilize the system's unstable modes stemming from referring the pilot's rotational dynamics to the wing's frame of motion.

The paper is structured as follows: Sect. 2 briefly describes the different decoupled aerodynamic models considered in this study. Section 3 presents the optimal control problem and the associated Bellman optimality conditions relevant to the system in hand. Section 4 proposes a policy iteration algorithm along with its convergence proof. Section 5 discusses the neural network implementation of the adaptive online learning scheme. Section 6 highlights the simulation outcomes of the proposed control architecture using different control methodologies. The merits of the online adaptive learning approach and its adaptability to real-world applications are introduced in Sect. 7. Finally, the paper is concluded with some remarks in Sect. 8.

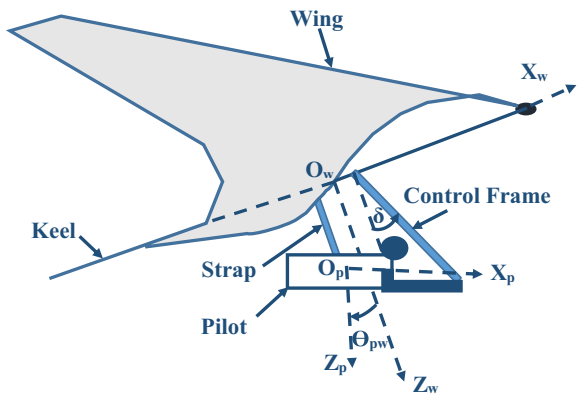


Fig. 1 Longitudinal geometry

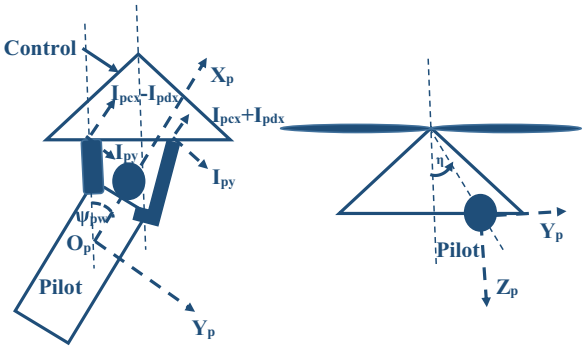


Fig. 2 Lateral geometry

2 Flexible wing aircraft dynamical models

The scope of the work is to design a robust controller with online model-free characteristics for flexible wing aircraft. In this section, two modeling attempts for a flexible wing system are used in order to discuss and test the characteristics of the adaptive learning controller. The notations of the aerodynamic systems are listed in addition to the assumptions used in previous modeling processes.

The hang glider operates as a two-body system with a kinematic constraint, where the pilot steers the control bar in order to achieve the desired flight maneuvers. The aircraft motion depends on the position of the pilot's center of gravity relative to that of the wing. This makes the modeling process of the continuously moving center of gravities a complex problem. Consequently, it is challenging to build model-based robust controller for this type of aircraft.

Figures 1 and 2 show the side and top views of the longitudinal and lateral motion frames of a flexible wing aircraft. The pilot and wing frames are denoted by (O_p, X_p, Y_p, Z_p) and (O_w, X_w, Y_w, Z_w), respectively. The pitch, roll, and yaw attitude angles of the wing are denoted by θ_w, ϕ_w, and ψ_w, respectively. Their rate of change with respect to time (i.e., rotational velocities) are symbolized by q_w, p_w, and r_w, respectively. The wing's transnational velocities (forward, normal, and lateral) are denoted by u_w, w_w, and v_w, respectively. The pilot's attitude angles with respect to the wing's frame of motion are expressed as θ_{pw}, ϕ_{pw} and ψ_{pw}. The pilot's rotational velocities relative to the wing's frame of motion are denoted by q_{pw}, p_{pw} and r_{pw}. Herein, two models are considered for the flexible wing aircraft. In the following, we provide a brief description of each model.

2.1 Position control model (Model 1)

Nonlinear equations of motion were derived in (Cook and Spottiswoode 2005) based on a semi-experimental study of the wing motion in its own frame. Linearized small-perturbation equations are then extracted. That study formulates the pilot's reaction control moment using a spring-damper model. Nonetheless, it overlooks the interaction forces between the pilot and the wing systems. More details about the modeling strategy and its assumptions can be found in Cook and Spottiswoode (2005).

In this open-loop model (*Model 1*), the aircraft motion is decoupled into longitudinal and lateral planes. Hence, two input control signals are adopted. The longitudinal motion is controlled through pitching the control bar by an angle δ (about the pitch axis), while the lateral motion is controlled through rolling the control bar by an angle η (about the roll axis), as shown in Figs. 1 and 2. These angles create the required shift in the relative locations of the pilot and wing centers of gravity to obtain the desired maneuvers.

As such, the nonlinear state space equations of the hang glider are expressed as follows:

$$\dot{x}^{Lon} = D_1^{-1}(H_1 + K_1 u^{Lon}), \quad \dot{x}^{Lat} = D_2^{-1}(H_2 + K_2 u^{Lat}), \tag{1}$$

where the longitudinal and lateral states are given by

$$x^{Lon} = \left[u_w\, w_w\, q_w\, \theta_w\right]^T, \quad x^{Lat} = \left[v_w\, p_w\, r_w\, \phi_w\, \psi_w\right]^T.$$

The matrices $D_1 \in \mathbb{R}^{4\times4}$, $H_1 \in \mathbb{R}^{4\times1}$, $K_1 \in \mathbb{R}^{4\times1}$, $D_2 \in \mathbb{R}^{5\times5}$, $H_2 \in \mathbb{R}^{5\times1}$, and $K_2 \in \mathbb{R}^{5\times1}$, reflect the nonlinearity in the aerodynamic models of the longitudinal and lateral motions. The longitudinal and lateral control input signals are denoted by $u^{Lon} = \delta$ and $u^{Lat} = \eta$, respectively.

2.2 Force control model (Model 2)

Ochi in Ochi (2017) extended *Model* (1) to take into consideration the dynamic coupling between the pilot and the control bar. The resultant nine degree-of-freedom (9-DOF) open-loop model relates the wing's rotational and transnational motion, and the pilot's motion relative to the wing. The equations of motions for the pilot and the wing are

developed separately. The internal forces at the hang point and the control bar are treated as external forces for the two-body system. The forces applied on the control bar are taken as the system's control inputs, while the internal forces at the hang point are substituted for in the wing's frame and eliminated. For more details about this modeling method and its limitations, the reader is referred to Ochi (2017).

Let $I_{pR} = [I_{pxR}\ I_{pyR}\ I_{pzR}]^T$ and $I_{pL} = [I_{pxL}\ I_{pyL}\ I_{pzL}]^T$ be the internal forces acting on the right and left pilot's grasping points on the control bar, respectively, defined in the pilot's frame. The aggregate force is given by $I_{pc} = \frac{1}{2}(I_{pR} + I_{pL}) = [I_{pcx}\ I_{pcy}\ I_{pcz}]^T$. The differential force is defined as $I_{pd} = \frac{1}{2}(I_{pR} - I_{pL}) = [I_{pdx}\ 0\ I_{pdz}]^T$. The required shift in the pilot's center of gravity in order to perform a certain maneuver can be defined by the force components I_{pc} and I_{pd}, which are considered as the input control signals for this control model (*Model 2*). These forces are illustrated in Fig. 2.

In this context, the nonlinear dynamical equation of the hang glider can be expressed as

$$\dot{x} = D^{-1}(H + K\,u), \tag{2}$$

where the matrices $D \in \mathbb{R}^{15\times15}$, $H \in \mathbb{R}^{15\times1}$, and $K \in \mathbb{R}^{15\times5}$ describe the nonlinearity in the aerodynamic model. The matrix D is proven to be nonsingular (Ochi 2017). The collective and differential force components $\boldsymbol{u}$ acting on the control bar and the states $\boldsymbol{x}$ are given such that

$$u = [I_{pcx}\ I_{pcy}\ I_{pcz}\ I_{pdx}\ I_{pdz}]^T,$$
$$x = [u_w\ v_w\ w_w\ p_w\ q_w\ r_w\ p_{pw}\ q_{pw}\ r_{pw}\ \phi_{pw}\ \theta_{pw}\ \psi_{pw}\ \phi_w\ \theta_w\ \psi_w]^T.$$

Unlike in Ochi (2017), this work does not assume I_{pcz} and I_{pdz} to be zero, since such an assumption may not hold in some practical scenarios. In other words, herein, the z-components of all the force inputs are accounted for. Furthermore, we are also decoupling the dynamics into longitudinal and lateral sub-systems,

$$x^{Lon} = [u_w\ w_w\ q_w\ q_{pw}\ \theta_{pw}\ \theta_w]^T, \qquad u^{Lon} = [I_{pcx}\ I_{pcz}]^T,$$
$$x^{Lat} = [v_w\ p_w\ r_w\ p_{pw}\ r_{pw}\ \phi_{pw}\ \psi_{pw}\ \phi_w]^T, \quad u^{Lat} = [I_{pcy}\ I_{pdx}\ I_{pdz}]^T.$$

Breaking the motion into longitudinal and lateral planes will deem to be helpful later in the control stage. Note how the state and input vectors adopted in this model have larger dimensions than those in the first model, which reveals the relative complexity of these models.

3 Optimal control formulation

In this section, the solution of the optimal control problem is developed in terms of the Bellman optimality principles (Lewis et al. 2012). First, Bellman equation associated with the trajectory of the dynamical system is defined and the optimal control policy is computed. Then, a modified form of Bellman equation is introduced to account for the dependency of the value function on the states and the control strategy.

Consider the following discrete-time state-space representation:

$$X_{k+1} = A\,X_k + B\,u_k + e_k, \tag{3}$$

where the subscript k denotes the time index, $X \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are vectors representing the states and the control signals, respectively, A and B are the approximations of the drift dynamics and control input matrices, and the vector e_k represents the noise and uncertainties in the aerodynamic model.

The flexible wing aircraft do not have exact aerodynamic models, where the previous studies are based on approximate semi-experimental platforms (as illustrated by *Model 1* and *Model 2*) (Cook and Spottiswoode 2005; Ochi 2017). Hence, the proposed adaptive learning approach uses the general dynamical form (3) in order to generate the online measurements, where A and B are recorded for *Model 1* and *Model 2* at a trim speed or steady state flight condition. Accordingly, the sizes of the longitudinal and lateral states and control signals for *Model 1* are ($n = 4$ and 5) and ($m = 1$ and 1) respectively. In a similar fashion, the sizes for *Model 2* take the values ($n = 6$ and 8) and ($m = 2$ and 3) respectively. It is worth to mention that, the simulation scenarios will involve considerable amount of dynamic variations around the nominal dynamical matrices A and B, which should account for an envelope of the flexible wing's aerodynamic variations to a great extent.

Let $P = \sum_{\ell=0}^{\infty} C(X_\ell, u_\ell)$ be a performance index to assess the quality of the taken actions, where C is a quadratic cost function given by $C(X_\ell, u_\ell) = \frac{1}{2}(X_\ell^T S X_\ell + u_\ell^T R u_\ell)$, such that $S \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are symmetric time-invariant positive-semidefinite and positive-definite weighting matrices, respectively. Define $F(X_k)$ to be a solving value function, such that

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

Then, Bellman equation for system (3) can be written as

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

This equation shows the dependence of the solving value function $F(\dots)$ on a selected set of measured states X. This temporal difference configuration results in a model-based optimal control strategy (Lewis et al. 2012). The following development explains how a model-free optimal control policy can be obtained by changing the argument of the solving

value function $F(\dots)$ to be dependent on the control signals as well as the set of measured variables or states.

Definition 1 (*System stabilizability*) A system in the form of (3) can be stabilized on a set (of states) $\Psi \subseteq \mathbb{R}^n$, if there exists a policy $u_\ell \in \mathbb{R}^m$ under which the system is asymptotically stable on Ψ.

Since the flexible wing aircraft do not have any exact models, different state feedback mechanisms can be employed in order to obtain a set of admissible policies. These can guarantee reachability to certain feasible dynamical situations given initial conditions selected from the dynamical environment. Additionally, admissible policies can be determined using state feedback mechanisms that employ considerable random changes around the trim or the nominal dynamical descriptions A and B (Lewis et al. 2012; Bellman 1957).

The optimal value function $F^o(\dots)$ (or the optimal performance measure P^o) can be computed by applying Bellman optimality conditions (constrained minimum conditions) (Lewis et al. 2012) to (4), such that $F^o(X_\ell) = \text{argmin}_{u_i} \sum_{i=\ell}^{\infty} C(X_i, u_i)$. The optimal policy decision making process can be regarded as a mapping $\mathbb{R}^n \to \mathbb{R}^m$ which maps the states X_ℓ into an optimal strategy u^o. Assume that system (3) is stabilizable on some set $\Psi \subseteq \mathbb{R}^n$. Then, the optimal control policy u^o is given by

$$u^o = -R^{-1} B^T \nabla F^o(X_\ell), \tag{6}$$

where $\nabla F^o(X_\ell) = \partial F^o(X_\ell) X_\ell$. Despite its successful applications in various control problems (Sutton and Barto 1998; Howard 1960; Bertsekas and Tsitsiklis 1995), the value function $F(\dots)$ takes into account the states X only, without considering the control signal u. This is due to the way Bellman equation is expressed in (5). The dependency on the dynamical model could lead to quality-degradation of the strategy. In the following, a modified expression of Bellman equation is introduced. The new relation takes into consideration the states as well as the control strategy.

Let us redefine the solving value function (4) as $\tilde{F}(X_\ell, u_\ell) = \sum_{i=\ell}^{\infty} C(X_i, u_i)$. Based on this expression, Bellman equation can now be rearranged as

$$\tilde{F}(X_\ell, u_\ell) = \frac{1}{2}\left(X_\ell^T S X_\ell + u_\ell^T R u_\ell\right) + \tilde{F}(X_{\ell+1}, u_{\ell+1}). \tag{7}$$

Applying Bellman optimality principles yields (Lewis et al. 2012),

$$\tilde{F}^o(X_\ell, u_\ell) = \min_{u_\ell} \left(\frac{1}{2}\left(X_\ell^T S X_\ell + u_\ell^T R u_\ell\right) + \tilde{F}^o(X_{\ell+1}, u_{\ell+1}) \right). \tag{8}$$

Tailoring the optimal policy accordingly, leads to the optimal control u^o,

$$u^o = -R^{-1} B^T \nabla \tilde{F}^o(X_{\ell+1}, u_{\ell+1}), \tag{9}$$

where $\nabla \tilde{F}^o(X_{\ell+1}, u_{\ell+1}) = \partial \tilde{F}^o(X_{\ell+1}, u_{\ell+1}) X_{\ell+1}$. Substituting in (7) leads to the following Bellman optimality equation:

$$\tilde{F}^o(X_\ell, u_\ell^o) = \frac{1}{2}\left(X_\ell^T S X_\ell + u_\ell^{oT} R u_\ell^o\right) + \tilde{F}^o(X_{\ell+1}, u_{\ell+1}^o), \quad \tilde{F}^o(\mathbf{0}, \mathbf{0}) = 0. \tag{10}$$

Solving the optimal control problem is now reduced to solving Bellman optimality equation (10) using the optimal policy (9). However, the accuracy of this policy depends on a priori knowledge of the input matrix B. One of the objectives of this work is to adopt a control strategy that is as robust as possible to modeling uncertainties. In the next section, an online model-free policy iteration approach is proposed to control systems defined by (3) starting with rough estimates of the system dynamics. In other words, the matrices A and B, do not have to be close to their nominal values.

4 Model-free policy iteration algorithm

A policy iteration approach is adopted here in order to maintain faster convergence compared to that achieved using the value iteration schemes. Yet, the proposed technique is free of matrix-inverse based computations. Instead, an actor-critic neural network approach is followed in order to implement the online solution.

4.1 Policy iteration solution

Consider a solving value function $Q(X_\ell, u_\ell)$ defined such that

$$Q(X_\ell, u_\ell) = \frac{1}{2} Z_\ell^T M Z_\ell, \tag{11}$$

where $Z_\ell^T = [X_\ell^T \; u_\ell^T]$ and M is a square Routh–Hurwitz positive-definite matrix of proper dimension with the following block structure:

$$M = \begin{bmatrix} M_{XX} & M_{Xu} \\ M_{uX} & M_{uu} \end{bmatrix}.$$

Approximating the value function $\tilde{F}(X_\ell, u_\ell)$ by $Q(X_\ell, u_\ell)$, such that

$$\tilde{F}(X_\ell, u_\ell) = \sum_{i=\ell}^{\infty} C(X_i, u_i) \simeq Q(X_\ell, u_\ell), \tag{12}$$

and solving (11) for the optimal policy u^o yields

$$u^o = \underset{u_\ell}{\text{argmin}}\, Q(X_\ell, u_\ell) = -M_{uu}^{-1} M_{uX} X_\ell. \tag{13}$$

Since M is positive-definite, M_{uu} is invertible, which guarantees the existence of the optimal control policy, as defined

in (13). It follows from (12) that Bellman equation can be rewritten as

$$Q(X_\ell, u_\ell) = \frac{1}{2}(X_\ell^T S X_\ell + u_\ell^T R u_\ell) + Q(X_{\ell+1}, u_{\ell+1}). \tag{14}$$

The way to do that and the reason why the algorithm updates this matrix every $(n+m)(n+m+1)/2$ iterations will be justified later in the adaptive critics implementation (Sect. 5).

Algorithm 1: Model-Free Online Policy Iteration Algorithm

Data: X_0, u_0, S, R, and a termination threshold ϵ
Result: optimal policy $u^o \equiv \text{argmin}_{u_\ell} Q(X_\ell, u_\ell)$

1 $r \leftarrow 0$ /* policy index */
2 $\ell \leftarrow 0$ /* time index */
3 stop $\leftarrow$ false /* stopping condition */
4 Initialize M^0 to a Routh-Hurwitz positive-definite matrix of proper dimensions
5 **while** *stop is false* **do**
6 Acquire $X_{\ell+1}$
7 Compute the policy $u_{\ell+1}$ using (17)
8 **if** $(\ell+1)$ *is a multiple of* $(n+m)(n+m+1)/2$ **then** /* new policy */
9 $r \leftarrow r+1$
10 Solve for M^r using (18) and the past $(n+m)(n+m+1)/2$ iterations
11 **if** $\|M^r - M^{r-1}\| < \epsilon$ **then**
12 stop $\leftarrow$ true
13 $\ell \leftarrow \ell+1$

The duality between optimal polices (9) and (13), and their associated optimal value functions $\tilde{F}^o(X_\ell, u_\ell^o)$ and $Q^o(X_\ell, u_\ell^o)$, represents the corner stone of the model-free adaptive learning controller. *Algorithm 1* describes the procedure to iteratively compute the optimal control policy u^o, defined in (13), as a solution of Bellman optimality equation

$$Q^o(X_\ell, u_\ell^o) = \frac{1}{2}(X_\ell^T S X_\ell + u_\ell^o R u_\ell^o) + Q^o(X_{\ell+1}, u_{\ell+1}^o). \tag{15}$$

The value function Q is evaluated at each time index ℓ. Its corresponding matrix M is updated at each new policy $r = 0, 1, 2, \ldots$. Hence, Q^r denotes the value function evaluated using policy r.

$$Q^r(X_\ell, u_\ell) = \frac{1}{2} Z_\ell^T M^r Z_\ell, \tag{16}$$

where $M^r = \begin{bmatrix} M_{XX}^r & M_{Xu}^r \\ M_{uX}^r & M_{uu}^r \end{bmatrix}$ is a matrix M corresponding to policy r. The control law stemming from policy r at time index ℓ is denoted by

$$u_\ell^r = -(M_{uu}^r)^{-1} M_{uX}^r X_\ell. \tag{17}$$

In an abuse of notation, u_ℓ^r is referred to as u_ℓ in some cases, but it should be implicitly understood that it produced from some policy r.

The matrix M^r is updated in line 10 of *Algorithm 1* by solving (18) for M^r.

$$\begin{aligned} Q^r(X_\ell, u_\ell) - Q^r(X_{\ell+1}, u_{\ell+1}) &= \frac{1}{2}(Z_\ell^T M^r Z_\ell - Z_{\ell+1}^T M^r Z_{\ell+1}) \\ &= \frac{1}{2}\left(X_\ell^T S X_\ell + u_\ell^T R u_\ell\right). \end{aligned} \tag{18}$$

It is important to notice that *Algorithm 1* does not depend on the system's model. It starts with an admissible policy and leads the value function and the control policy to converge to their optimal respective values given by (13) and (15). Acquiring $X_{\ell+1}$ in line 6 can be read directly from sensor measurements in the real-world application, or by applying the system model [such as (1) or (2)] in simulation.

4.2 Convergence analysis

We now formally assess the convergence properties of *Algorithm 1*.

Definition 2 (*Admissible policy*) A policy with control law u_ℓ is said to be admissible, if it stabilizes the system [such as (3)] and its corresponding value function $Q(X_\ell, u_\ell)$ is finite $\forall \ell$.

The following theorem shows that policy (13) leads to a sequence of monotonically converging value functions and stabilizing policies.

Theorem 1 *If Algorithm 1 is applied to control system* (3) *with value function* (14) *and policy update law* (13), *starting with an initial admissible policy, then*

1. *The generated sequence of policies u_ℓ^r is stabilizing and admissible, $\forall r \geq 0$.*
2. *The generated value functions are monotonically nonincreasing: $Q^0 \geq Q^1 \geq \cdots \geq Q^o(X_\ell, u_\ell^o)$.*

Proof From (14)

$$Q^r(X_{\ell+1}, u^r_{\ell+1}) - Q^r(X_\ell, u^r_\ell) \le 0, \ \forall r. \tag{19}$$

Q^r can be regarded as a Lyapunov function given a policy $u^r_\ell, \forall r$. Then

$$Q^r(X_\ell, u^r_\ell) - Q^r(X_\ell, u^{r+1}_\ell) = \Delta C(u^r_\ell, u^{r+1}_\ell), \forall r$$

where $\Delta C(u^r_\ell, u^{r+1}_\ell) = \sum_{i=\ell}^{\infty} \frac{1}{2}\left(u^r_i - u^{r+1}_i\right)^T R\left(u^r_i - u^{r+1}_i\right) + u_i^{(r+1)T} R\left(u^r_i - u^{r+1}_i\right)$.

This proves that $\Delta C(u^r_\ell, u^r_{\ell+1}) \ge 0$, $\forall r$. Then, $Q^r(X_\ell, u^r_\ell) \ge Q^r(X_\ell, u^{r+1}_\ell), \forall r$. Therefore, u^{r+1}_ℓ is a stabilizing policy and hence admissible, $\forall r$. This concludes that, starting with an initial admissible policy, all subsequent policies are stabilizing and admissible.

Now, let us prove that the generated value functions are monotonically nonincreasing. Inequality (19) yields, $\forall r$,

$$\begin{aligned} &Q^r(X_{\ell+1}, u^{r+1}_{\ell+1}) - Q^r(X_\ell, u^{r+1}_\ell) \le -C(X_\ell, u^{r+1}_\ell) \le 0 \\ &Q^{r+1}(X_{\ell+1}, u^{r+1}_{\ell+1}) - Q^{r+1}(X_\ell, u^{r+1}_\ell) + C(X_\ell, u^{r+1}_\ell) = 0. \end{aligned}$$

Therefore,

$$\begin{aligned} 0 &\le Q^r(X_{\ell+1}, u^{r+1}_{\ell+1}) - Q^r(X_\ell, u^{r+1}_\ell) \\ &\le Q^{r+1}(X_{\ell+1}, u^{r+1}_{\ell+1}) - Q^{r+1}(X_\ell, u^{r+1}_\ell). \end{aligned} \tag{20}$$

Summing both sides of inequality (20),

$$\begin{aligned} 0 &\le \sum_{\ell=L}^{\infty} \left(Q^r(X_{\ell+1}, u^{r+1}_{\ell+1}) - Q^r(X_\ell, u^{r+1}_\ell)\right) \\ &\le \sum_{\ell=L}^{\infty} \left(Q^{r+1}(X_{\ell+1}, u^{r+1}_{\ell+1}) - Q^{r+1}(X_\ell, u^{r+1}_\ell)\right). \end{aligned}$$

Consequently,

$$0 \le Q^r(X_\infty, u^{r+1}_\infty) - Q^r(X_L, u^{r+1}_L) \le Q^{r+1}(X_\infty, u^{r+1}_\infty) - Q^{r+1}(X_L, u^{r+1}_L).$$

Using the stability property proven above, we conclude that

$$Q^r(X_L, u^{r+1}_L) \ge Q^{r+1}(X_L, u^{r+1}_L) \ge 0.$$

Therefore, by induction, we get

$$Q^0 \ge Q^1 \ge Q^2 \ge \cdots \ge Q^r \ge Q^{r+1} \ge \cdots \ge 0. \tag{21}$$

where $\cdots$ denotes the updated decreasing pattern of the value functions $Q^r, \forall r$.

This proves that *Algorithm 1* generates a sequence of policies u^r with monotonically non-increasing value functions Q^r, which are lower- and upper-bounded by 0 and Q^0, respectively. Therefore, the sequence of values functions (21) converges to its lower bound and optimal value Q^o.

$$Q^0 \ge Q^1 \ge Q^2 \ge \cdots \ge Q^o(X_\ell, u^o_\ell) \ge \cdots \ge 0.$$

The optimal value function Q^o represents the solution for Bellman optimality equation (15). □

5 Adaptive critics implementation

The problem with *Algorithm 1* is that there is no viable method to calculate the value function $Q^r(X_\ell, u_\ell)$ [from (16) or (18)]. As a result, it is not possible to compute the matrix M^r, which in turn implies that the optimal control policy $u_{\ell+1}$ in line 7 cannot be calculated. As a remedy of this problem, the value function and the optimal policy are approximated using actor-critic neural network structures.

The policy approximation neural network (the actor) estimates the optimal policy, while the usefulness of this policy is assessed by the critic neural network by approximating its value function (Sutton and Barto 1998). The coupled actor-critic networks are tuned concurrently in a process that is repeated until the weights of each network converge.

The estimate $\hat{Q}(X_\ell, \hat{u}_\ell)$ of the value function can be expressed as

$$\hat{Q}(X_\ell, \hat{u}_\ell) = \frac{1}{2} \hat{Z}_\ell^T W_c \hat{Z}_\ell, \tag{22}$$

where $W_c \in \mathbb{R}^{(n+m)\times(n+m)}$ is a symmetric positive definite matrix and $\hat{Z}_\ell^T = [X_\ell^T \ \hat{u}_\ell^T]$. Expression (22) of $\hat{Q}(X_\ell, \hat{u}_\ell)$ can be reformulated as a linear relationship which is conveniently implementable through a simple single-layer neural network, such that

$$\hat{Q}(X_\ell, \hat{u}_\ell) = \tilde{W}_c^T \tilde{Z}_\ell = (\mathrm{vec}(W_c))^T (\hat{Z}_\ell \otimes \hat{Z}_\ell), \tag{23}$$

where $\otimes$ denotes Kronecker product, $\tilde{W}_c = \mathrm{vec}(W_c)$, and $\tilde{Z}_\ell = \hat{Z}_\ell \otimes \hat{Z}_\ell$. The operator $\mathrm{vec}(\cdot)$ forms a vector by stacking the columns of its matrix argument. In this case, since W_c is symmetric, the critic network weight vector $\tilde{W}_c = \mathrm{vec}(W_c)$ is formed by stacking only the lower or upper triangular matrix. This is why it is a vector of $(n+m)(n+m+1)/2$ elements. The term $\tilde{Z}_\ell = \hat{Z}_\ell \otimes \hat{Z}_\ell$ is a quadratic polynomial vector containing all possible paiwise products of the components of $\hat{Z}_\ell$. In other words, if $\hat{Z}_i$ is the ith component of $\hat{Z}_\ell$, for $i = 1, 2, \ldots, (n+m)(n+m+1)/2$, then

$$\tilde{Z}_\ell^T = [\hat{Z}_1^2, \hat{Z}_1\hat{Z}_2, \hat{Z}_1\hat{Z}_3, \ldots, \hat{Z}_2^2, \hat{Z}_2\hat{Z}_3, \ldots].$$

In order to derive the critic weight update law, we define $\tilde{Q}_{\ell,\ell+1} = \hat{Q}(X_\ell, \hat{u}_\ell) - \hat{Q}(X_{\ell+1}, \hat{u}_{\ell+1})$. Hence,

$$\tilde{Q}_{\ell,\ell+1} = \tilde{W}_c^T \left(\tilde{Z}_\ell - \tilde{Z}_{\ell+1}\right). \tag{24}$$

Ideally, the critic network maps its input $(\tilde{Z}_\ell - \tilde{Z}_{\ell+1})$ to a desired value $\hat{Q}_\ell^{\text{desired}} = C(X_\ell, \hat{u}_\ell)$. In other words,

$$\tilde{W}_c^T \left(\tilde{Z}_\ell - \tilde{Z}_{\ell+1}\right) = \hat{Q}_\ell^{\text{desired}} = C(X_\ell, \hat{u}_\ell) = \frac{1}{2}\left(X_\ell^T S X_\ell + \hat{u}_\ell^T R \hat{u}_\ell\right). \tag{25}$$

The critic weights are updated at the beginning of each policy after collecting $(n+m)(n+m+1)/2$ data samples in order to minimize the following squared error:

$$E_c^r = \left|\left|\tilde{W}_c^T \Gamma_m^r - \Gamma_v^r\right|\right|^2, \tag{26}$$

where r is a policy index, Γ_m^r is a matrix of proper dimension whose j^{th} column is $(\tilde{Z}_\ell - \tilde{Z}_{\ell+1})$ and Γ_v^r is a row vector whose j^{th} component is $\hat{Q}_\ell^{\text{desired}}$, with $\ell = r(n+m)(n+m+1)/2$.

Applying a gradient-descent approach in order to minimize (26) leads to the following weight update rule:

$$(\tilde{W}_c^{r+1})^T = (\tilde{W}_c^r)^T - \ell_c\left((\tilde{W}_c^r)^T \cdot \Gamma_m^r - \Gamma_v^r\right) \cdot \left(\Gamma_m^r\right)^T, \tag{27}$$

where $0 < \ell_c < 1$ is a critic learning rate.

The corresponding target policy approximation is given by

$$\hat{u}_\ell^{\text{desired}} = -W_{c\hat{u}\hat{u}}^{-1} W_{c\hat{u}X} X_\ell, \tag{29}$$

where the critic weight matrix is divided into the following block structure:

$$W_c = \begin{bmatrix} W_{cXX} & W_{cX\hat{u}} \\ W_{c\hat{u}X} & W_{c\hat{u}\hat{u}} \end{bmatrix},$$

with square matrices $W_{cXX} \in \mathbb{R}^{n\times n}$ and $W_{c\hat{u}\hat{u}} \in \mathbb{R}^{m\times m}$.

Following a gradient descent approach, as with the critic weights, the update rule for the actor weights follows as

$$(W_a^{\text{new}})^T = (W_a^{\text{old}})^T - \ell_a(\hat{u}_\ell - \hat{u}_\ell^{\text{desired}})X_\ell^T, \tag{30}$$

where $0 < \ell_a < 1$ is an actor learning rate.

Algorithm 2 represents an amended version of *Algorithm 1*, where the actor-critic networks are employed to approximate the optimal policy and the value function. The approximation is accomplished in real-time without the need to model the system dynamics.

Algorithm 2: Model-Free Online Policy Iteration Algorithm with Actor-Critic Neural Network Implementation

Data: X_0, u_0, S, R, and a termination threshold ϵ
Result: optimal policy $u^o \equiv \text{argmin}_{u_\ell} Q(X_\ell, u_\ell)$

```
r ← 0                                   /* policy index */
ℓ ← 0                                   /* time index */
stop ← false                            /* stopping condition */
Initialize the critic weight matrix W̃_c^r to a Routh-Hurwitz positive-definite matrix of
  proper dimensions
Initialize the actor weights W_a
while stop is false do
    Evaluate the policies û_ℓ and û_ℓ^desired using (28) and (29), respectively
    Acquire X_{ℓ+1}
    Compute the policy û_{ℓ+1} using (28)
    Evaluate Q̃_{ℓ,ℓ+1} using (24) and Q̂_ℓ^desired using (25)
    if (ℓ+1) is a multiple of (n+m)(n+m+1)/2 then        /* new policy */
        r ← r + 1
        Update the critic weights (i.e., compute W̃_c^{r+1}) using (27)
        Update the actor weights using (30)
        if ||W_c^r − W_c^{r−1}|| < ε then
            stop ← true
    ℓ ← ℓ + 1
```

At every time instant ℓ, the actor network maps the state vector X_ℓ to an estimated optimal policy $\hat{u}_\ell$ through the following linear relationship:

$$\hat{u}_\ell = W_a^T X_\ell, \tag{28}$$

where $W_a \in R^{n\times m}$ is a matrix containing the actor weights.

6 Simulation results and discussion

Two case studies are considered in order to illustrate the performance of the proposed control scheme on each of the aircraft models. Case study 1 applies the control algorithm on the nominal dynamics of the models, while Case

study 2 is a generalization of the control problem around a trim condition, where disturbances are induced to represent time-varying imperfections in the models. This scenario can cope with the varying aerodynamic envelope of the flexible wing aircraft. The disturbances are drawn at each instance (iteration step) from a normal Gaussian distribution with a variance of 15% around the aerodynamic nominal parameters of the trim model, such that

$$X_{k+1} = \left(A + \Delta A_k\right)X_k + B\,u_k^O + \Delta B_k\,u_k^P,$$

where ΔA_k and ΔB_k are the superimposed aerodynamic disturbances; u_k^O is the control signal stemming from solving the Riccati equation (this control signal is found by solving a standard Riccati equation using the dynamics A and B and the weighting matrices S and R) (Lewis et al. 2012); and u_k^P is the control signal provided by the proposed framework (i.e., *Algorithm 2*). These dynamics are applied along the longitudinal and lateral motion frame of each model.

The dynamics of both models used in the simulations are detailed in Appendices 1 and 2. Obviously, such dynamics are unknown to the controller and are only used to simulate the aircraft's behavior in response to the actions generated by the controllers. For completeness, the aircraft's specifications are given in Appendix 3.

The simulations adopted a sampling time of $T_s = 0.01$ s. The weighting matrices are set to $S = 0.1I_A$ and $R = 0.1I_u$, where I_A and I_u are identity matrices with the same sizes as A and u, respectively. The learning rates are chosen as $\ell_a = \ell_c = 0.01$.

The graphical notations 'o' are used to describe the open-loop eigenvalues, while the marks '+' denote the final closed-loop eigenvalues. The closed-loop eigenvalues evolving during the adaptive learning process are given the '*' symbols.

6.1 Control of Model 1

In the sequel, the performance of the adaptive learning control algorithm is analyzed using the control configuration suggested by *Model 1*.

6.1.1 Longitudinal frame of motion

The simulation results of the longitudinal motion are reported in Figs. 3 and 4. The adaptation processes of the actor and critic weights are shown to converge as shown in Fig. 3. It is worth noticing (from Fig. 4b) that the open-loop longitudinal system is naturally unstable. It has an unstable phugoid mode with eigenvalues $1.0008 \pm 0.0116i$, and a short period pitching mode with eigenvalues $0.9799 \pm 0.0214i$. The same figure shows the closed-loop eigenvalues evolution in the z-domain during the controller's learning phase. As can be seen, the controller forced the phugoid mode to shift inside the stability region (unit circle) to guarantee the system's asymptotic stability. Closing the control loop with the controller in hand in the first case study, originally resulted in a unstable closed-loop poles that are stretched out far from the unit circle. This explains why the control effort at the beginning of the simulation of that case is larger than that of the second case. This is manifested in the fluctuating behavior of the dynamics (Fig. 4a) and the actor weights (Fig. 3b) in the first half of the simulation of case study 1. The asymptotic behavior is further emphasized by the values of the applied control signals as shown in Fig. 4c.

6.1.2 Lateral frame of motion

Similarly, the simulation results of the lateral motion are depicted in Figs. 5 and 6. The employed adaptive learning mechanism exhibited converging and asymptotic stability features as can be revealed from the actor-critic tuning evolution in Fig. 5 and the dynamical behavior of the lateral motion in Fig. 6a using the two simulation scenarios. The lateral open-loop system is marginally stable with poles at: $1, 0.9972 \pm 0.0088i$ (dutch roll mode), 0.9949 (spiral mode), and 0.7978 (roll mode). Note how despite the relatively fast time response of the roll mode, the dutch roll and spiral modes are characterized by slow transient characteristics. The evolution of the closed-loop eigenvalues (Fig. 6b), reveals how the controller dragged the sluggish modes further into the stability region in order to make the system asymptotically stable. This is clearly illustrated in case study 2, the difference between the open- and closed-loop dutch roll and roll modes is significant.

6.2 Control of Model 2

In the following discussion, the control configuration suggested by *Model 2* is employed to validate the performance of the adaptive learning mechanism in the decoupled motion frames.

6.2.1 Longitudinal frame of motion

The mapping features of the pilot's relative motion into the wing's frame of motion, created a stable but sluggish system. This can be seen as a direct result of using the transformation matrices that mapped the force components I_{pcx} and I_{pcz}, which are acting on the control bar, into the wing's system.

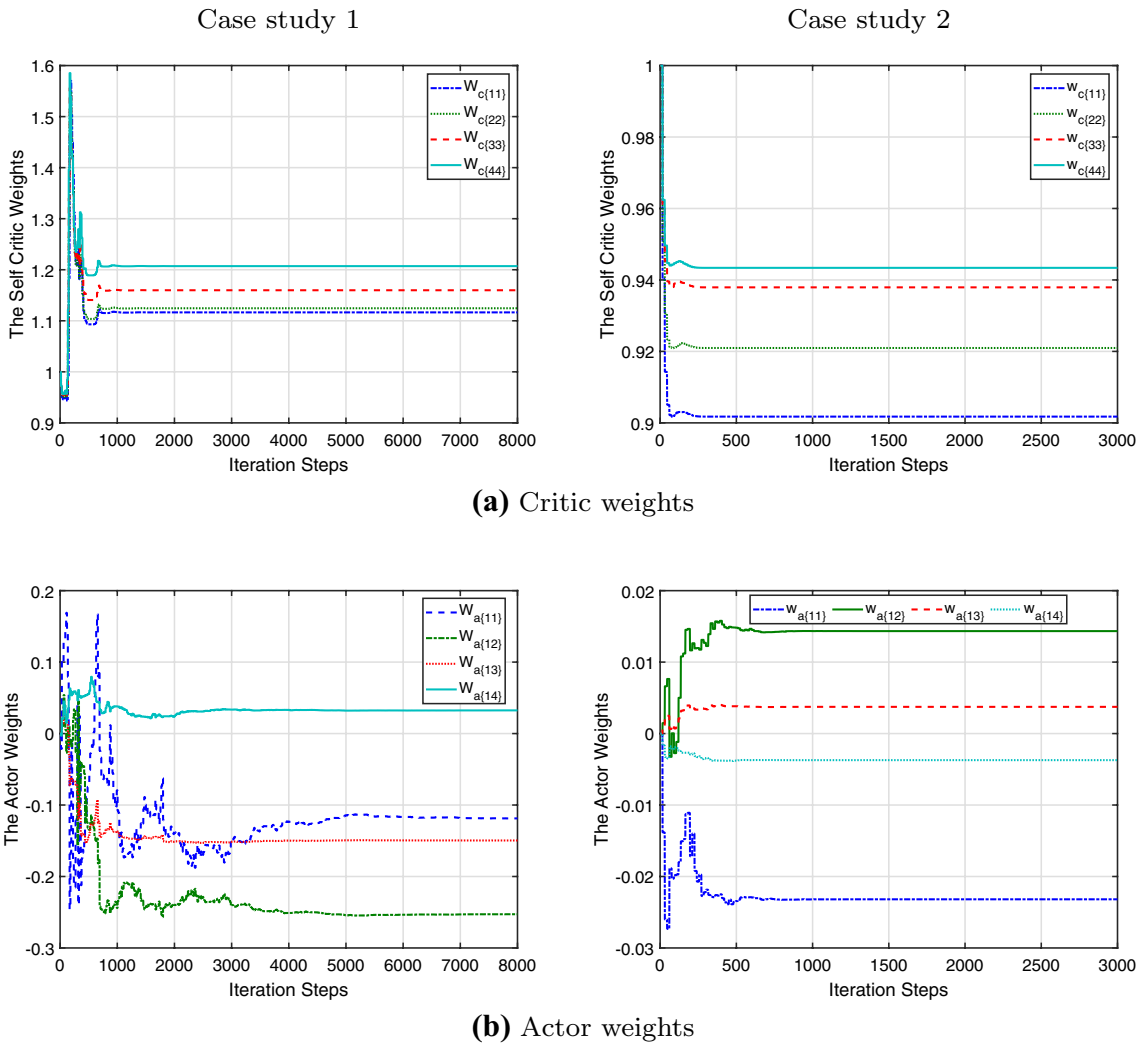
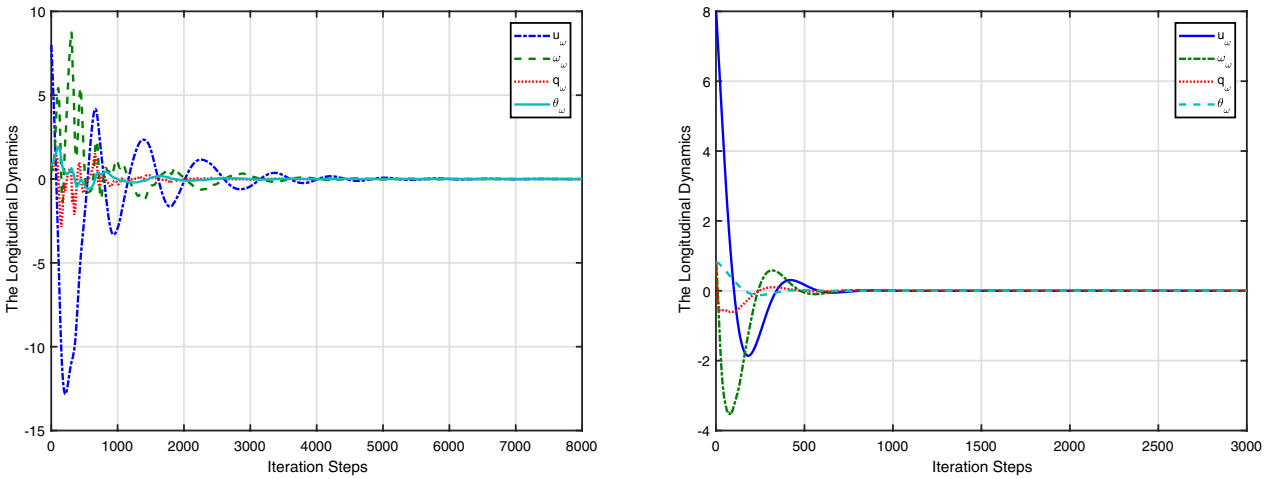


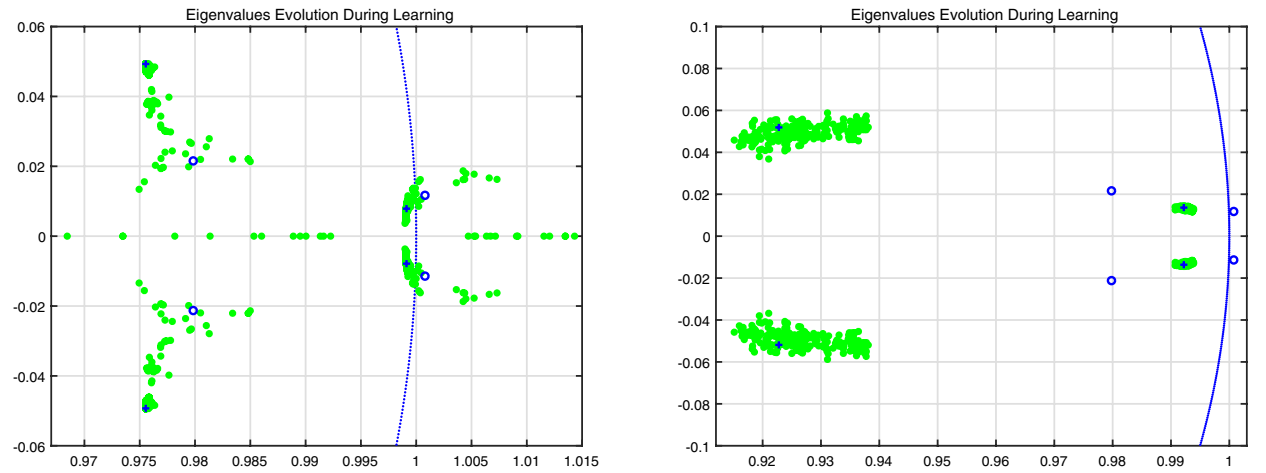
Fig. 3 Learning weights corresponding to the aircraft's longitudinal control based on Model 1

In this formulation, two additional longitudinal motion states q_{pw} and θ_{pw} (compared to *Model 1*) are considered and the control signals are the two-force components. Herein, all the involved force control signals are utilized unlike (Ochi 2017), where the z-component is not considered. The open-loop poles of the longitudinal dynamics are: 0.9938, 0.9995, $0.9995 \pm 0.0043i$ (pitching mode), and $0.9999 \pm 0.0008i$ (original phugoid mode). Although all open-loop poles are stable, they are associated with slow time responses.

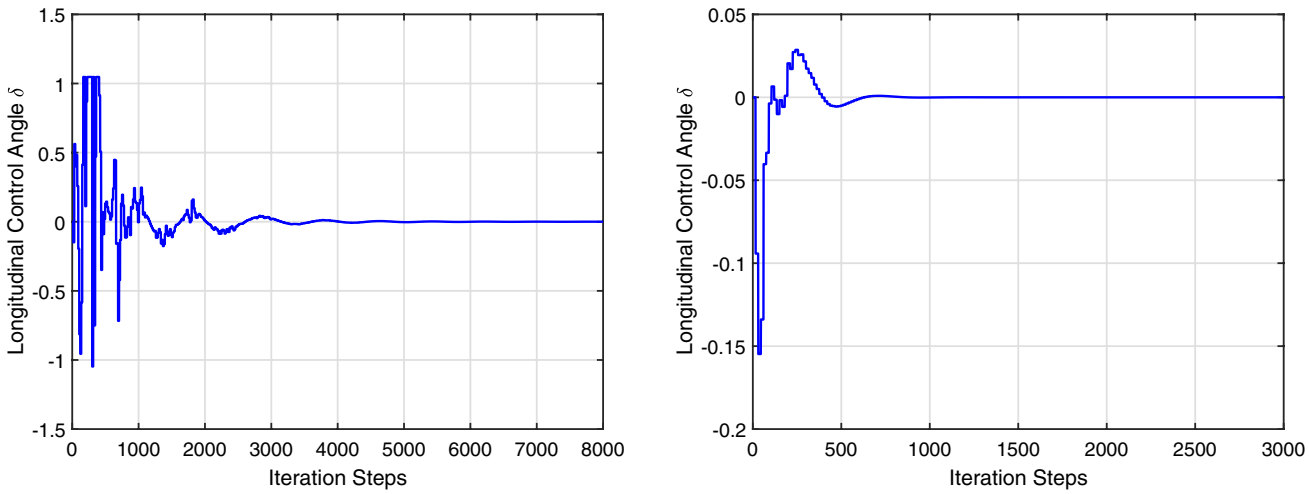
The simulations of this model along the longitudinal plane are shown in Figs. 7 and 8. Figure 7 shows the consistency in the convergence characteristics achieved using the adaptive learning process when the control configuration is ultimately modified, where asymptotic stability is continuously observed as shown by Fig. 8. Applying the controller to case study 1 did not have much effect, in the sense that it led to closed-loop poles almost superpositioned with the open-loop ones. However, it had a clear effect on the second case study, where it shifted the dominant poles corresponding to the phugoid mode further inside in the unit circle, which improves this mode's time response of the aircraft. It is worth noticing how the closed-loop poles corresponding to the phugoid mode found their way outside of the unit circle during the controller's exploration process around time 300 s. Nevertheless, it rapidly stabilized the system by applying sufficiently large control signals (Fig. 8c). The learning process converged shortly after that (Fig. 7).



(a) Dynamics (SI units)



(b) Pole evolution



(c) Control signals (SI units)

Fig. 4 Aircraft's longitudinal control based on Model 1

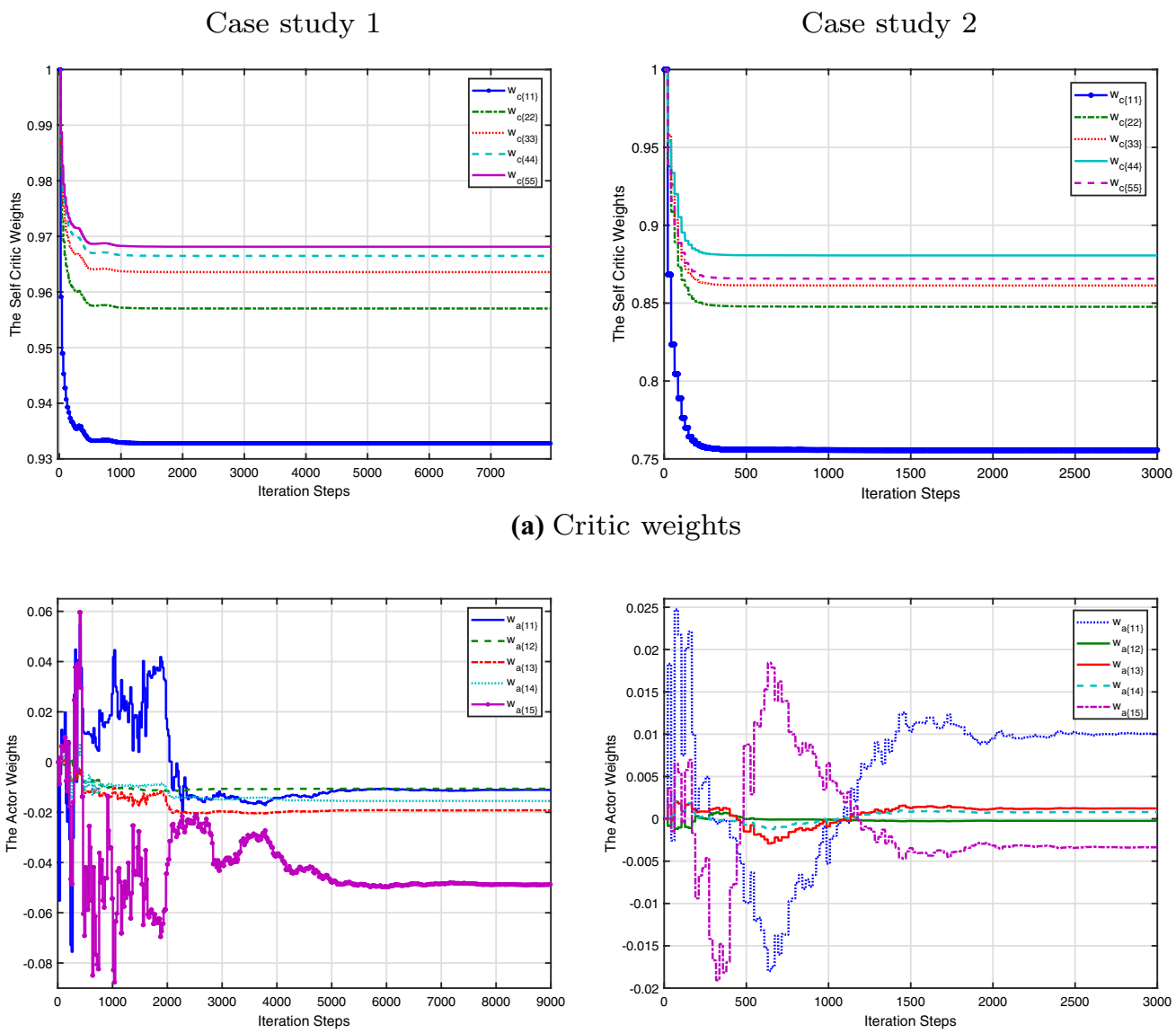
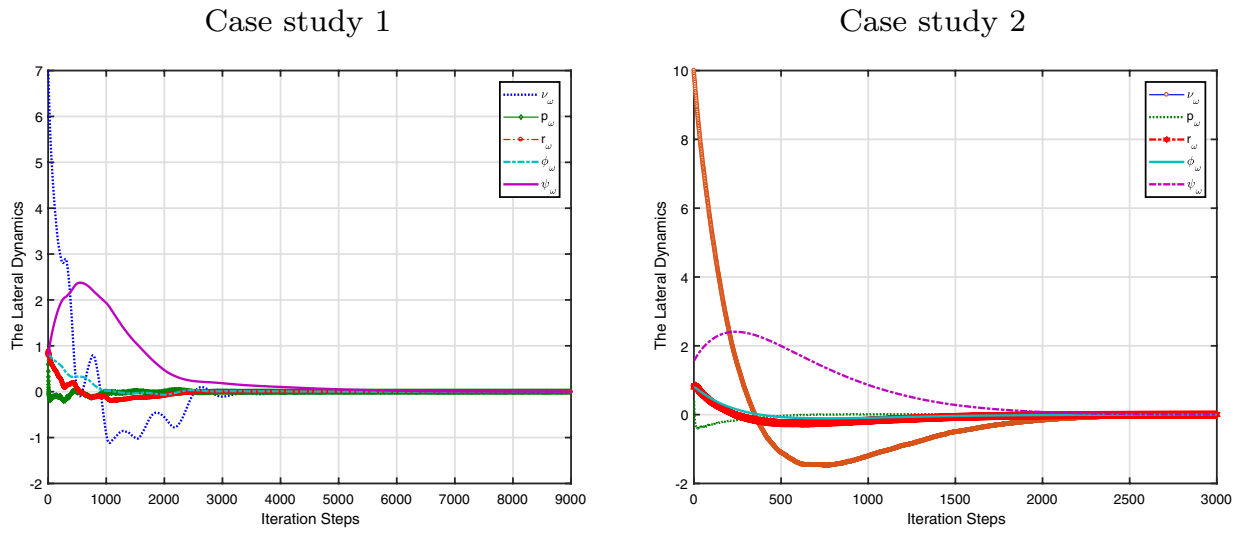


Fig. 5 Learning weights corresponding to the aircraft's lateral control based on Model 1

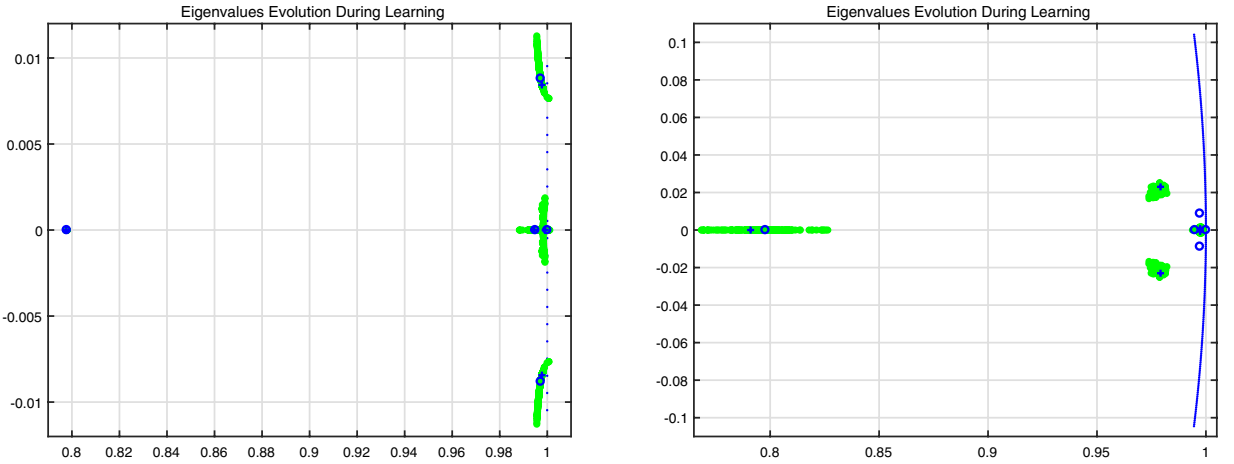
6.2.2 Lateral frame of motion

In this model four additional states related to the pilot's relative motion in the wing's frame of motion are considered p_{pw}, r_{pw}, ϕ_{pw}, and ψ_{pw} (compared to the lateral motion *Model 1*). Herein, all the force control signals (i.e., I_{pcy}, I_{pdx}, and I_{pdz}) are utilized unlike (Ochi 2017). The open-loop eigenvalues of this system are: 0.9740 (roll mode), $0.9997 \pm 0.0054i$ (dutch roll), $0.9994 \pm 0.0012i$ (relative roll), $1.0000 \pm 0.0004i$ (relative yaw mode), and 1.0002 (spiral mode). As can be seen, the spiral and relative yaw modes are unstable unlike the marginally stable properties of the lateral dynamics of *Model 1*. This makes the control process of this system more challenging compared to the other model.

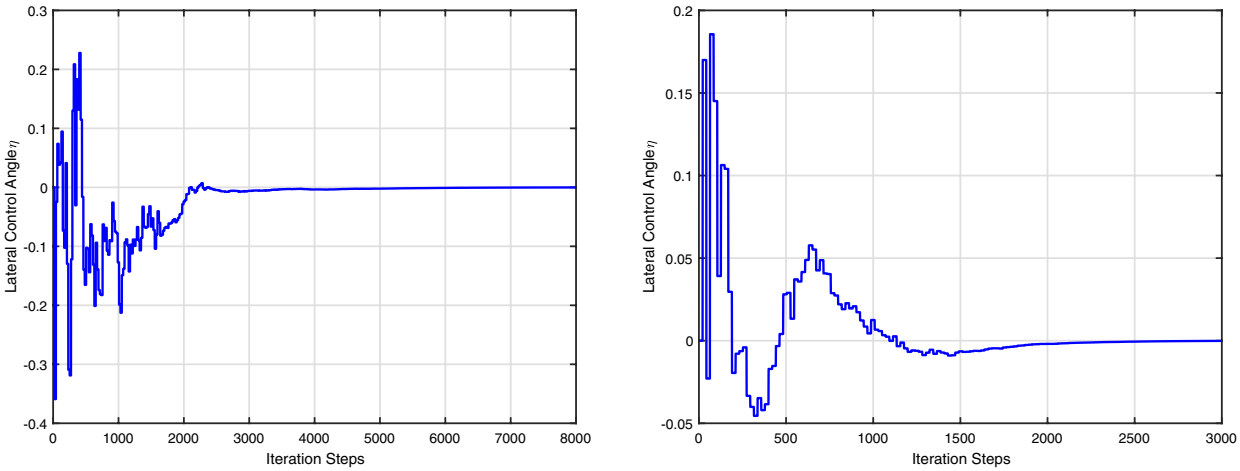
The simulation plots of the lateral dynamics control are shown in Figs. 9 and 10. Once again, closing the loop with the proposed controller originally resulted in dutch-mode closed-loop poles that are way out of the unit circle in the first case study. This is shown in the sluggish lateral velocity



(a) Dynamics (SI units)

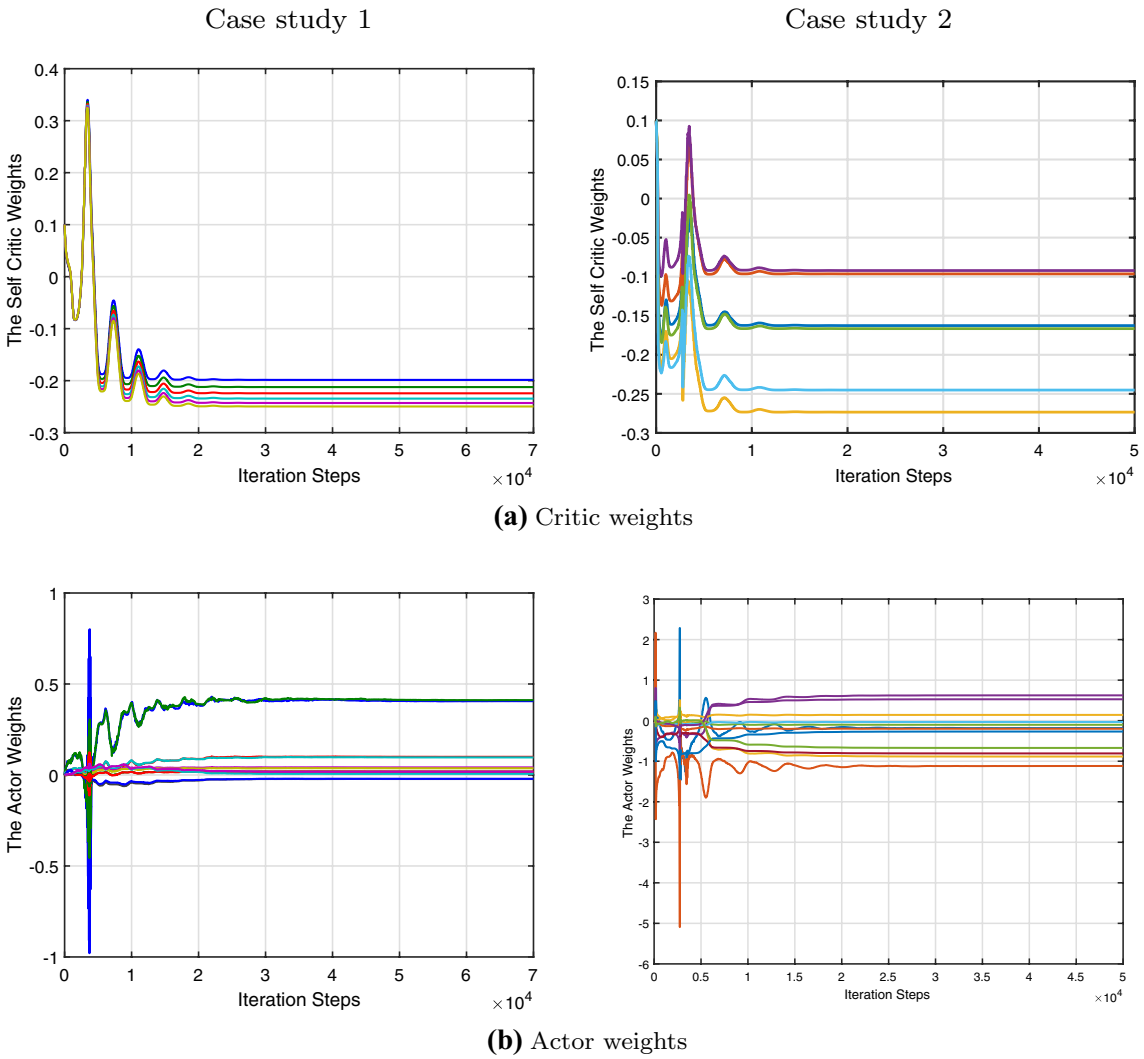


(b) Pole evolution



(c) Control signals (SI units)

Fig. 6 Aircraft's lateral control based on Model 1



(a) Critic weights

(b) Actor weights

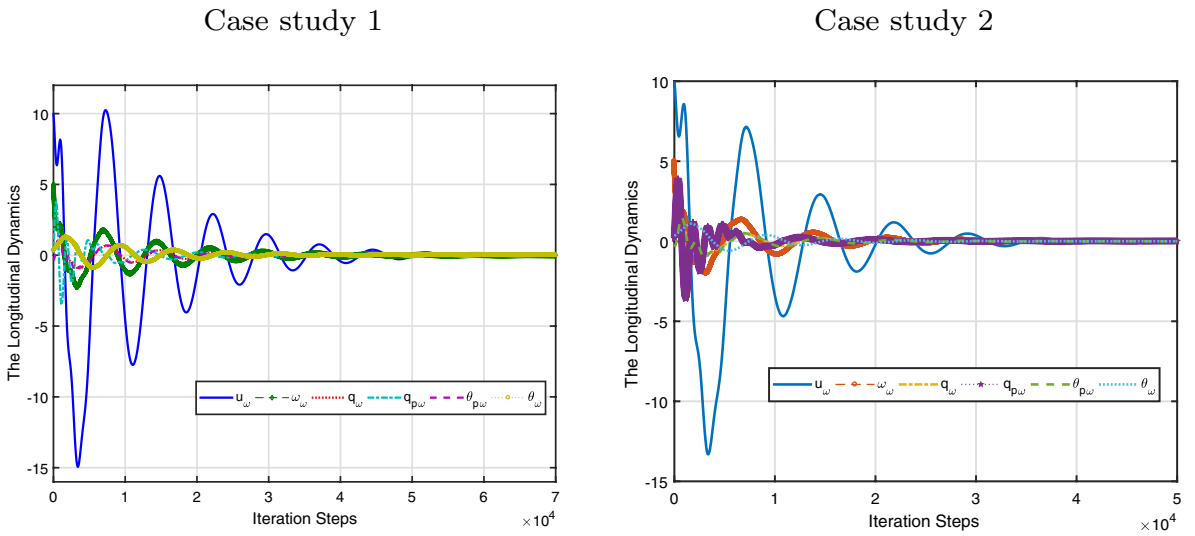
Fig. 7 Learning weights corresponding to the aircraft's longitudinal control based on Model 2

of the wing, for example, at the beginning of the simulation (Fig. 10a). It was required that controller applies a considerable amount of force on the control bar to regulate the aircraft (Figure 10c). As a result, the actor neural network spent a longer time exploring the weight search space searching for optimal control actions to stabilize the system (Fig. 9b). It is also interesting to note how the closed-loop roll and spiral modes are significantly improved in this case compared to their open-loop counterparts.

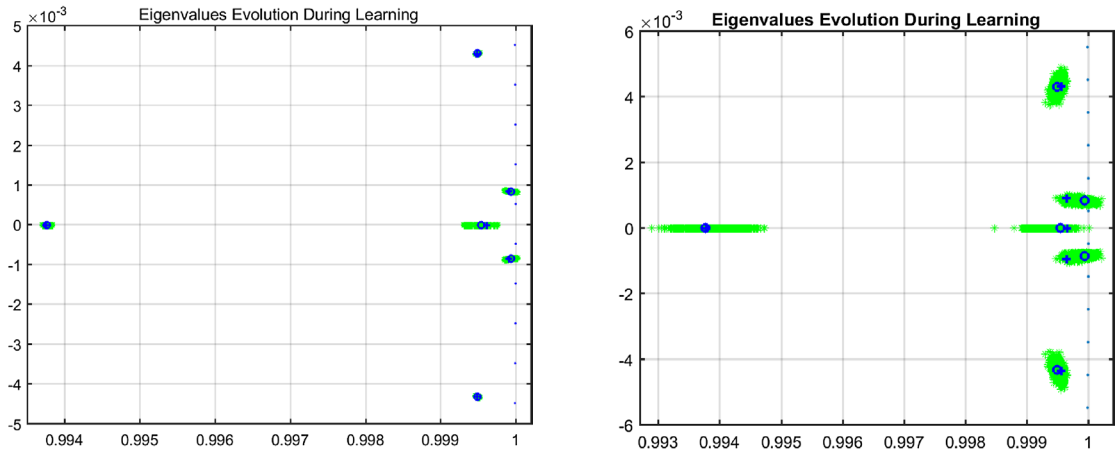
Overall, the proposed online adaptive learning approach exhibited the ability to control systems with continuously varying dynamics. It achieved asymptomatic stability and improved the time-response characteristics of the simulated models. The closed-loop eigenvalues are eventually guided towards the stability region. Despite the fundamental structural differences between both models, the adaptive learning approach was robust to these discrepancies.

7 Merits and adaptability to real-world applications

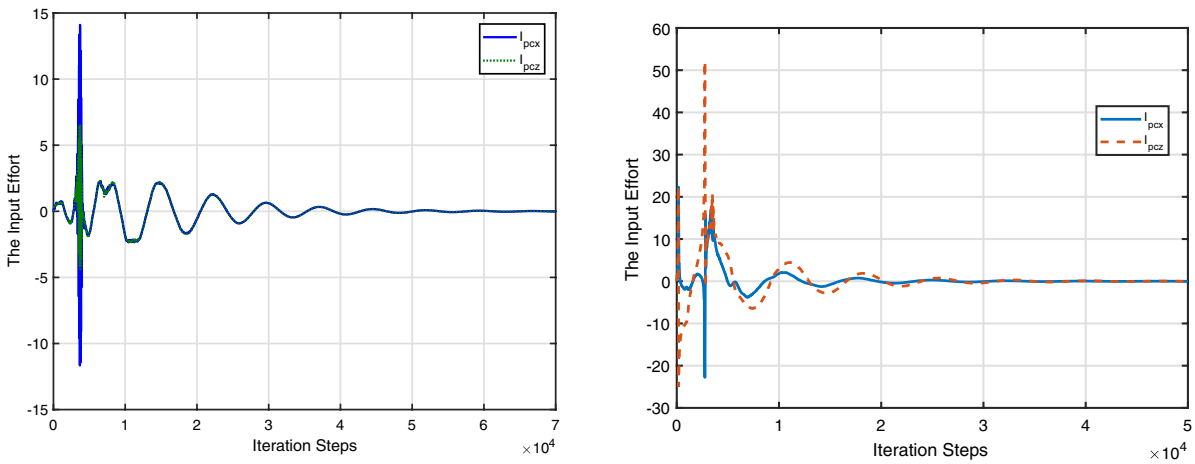
This section highlights the main advantages of the proposed policy iteration approach along with the accompanying real-world practical implications and it introduces future ideas towards multi-objective autonomous control schemes.



(a) Dynamics (SI units)

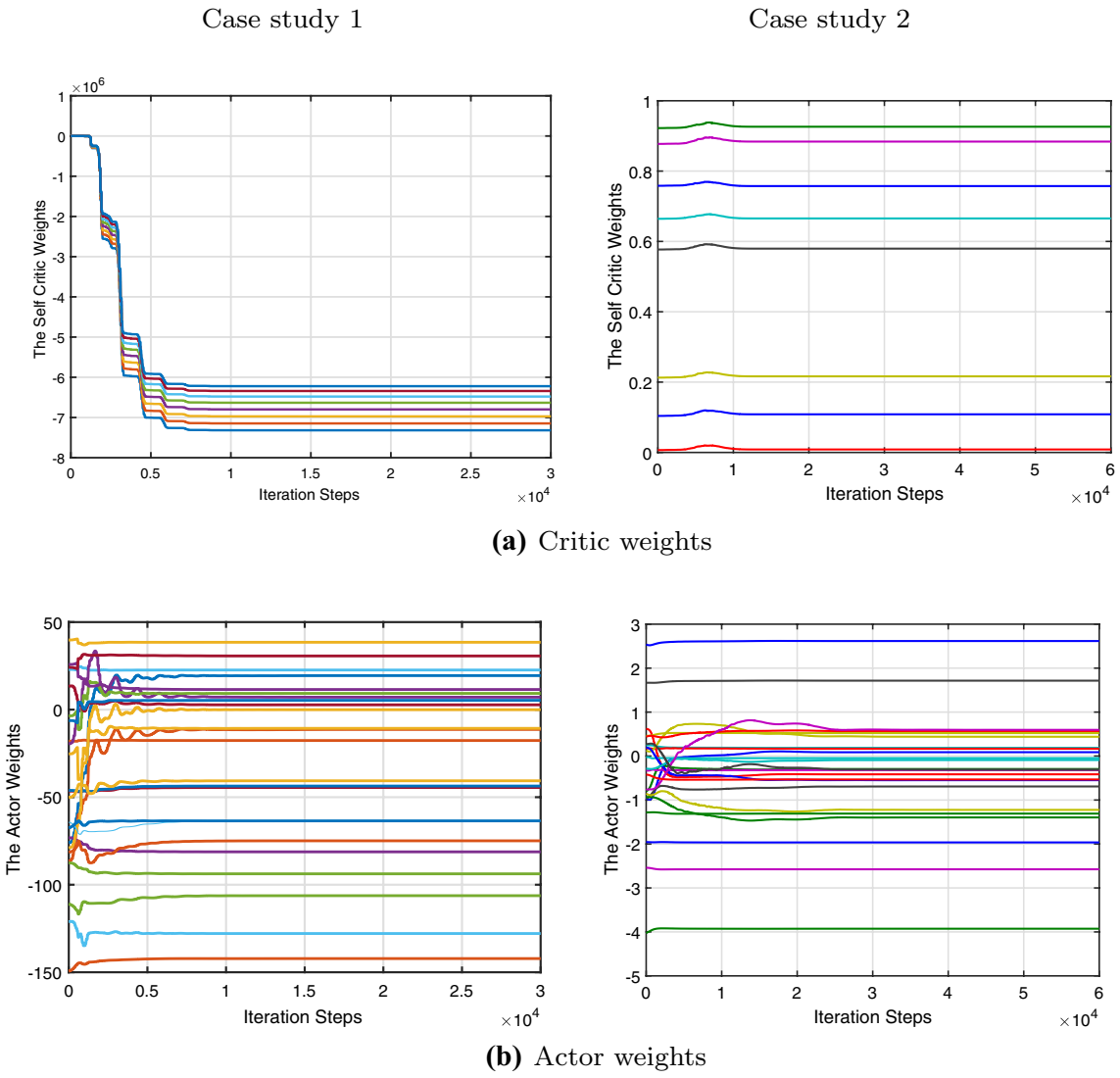


(b) Pole evolution



(c) Control signals (SI units)

Fig. 8 Aircraft's longitudinal control based on Model 2

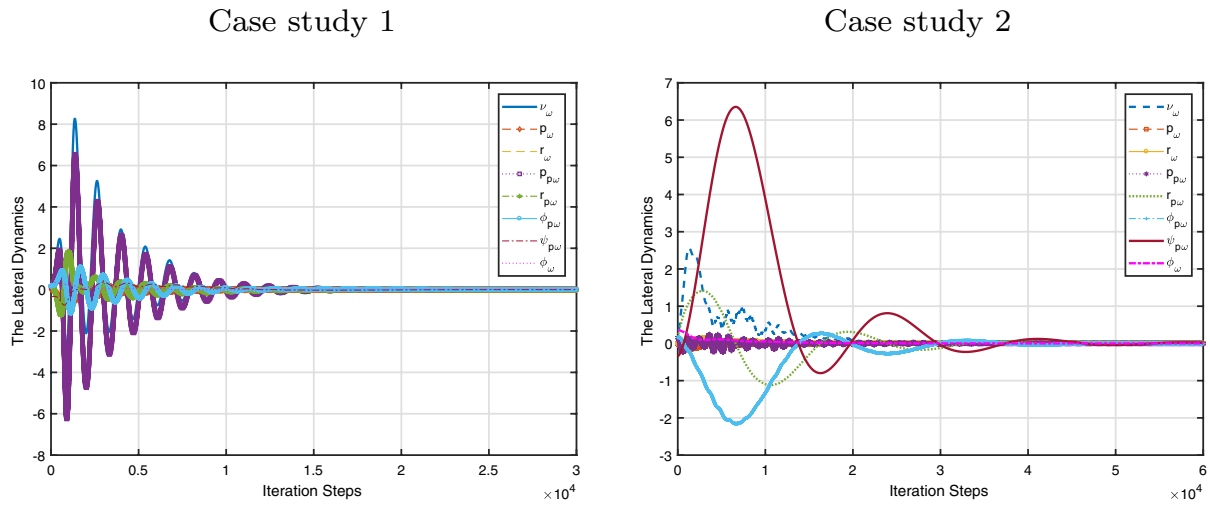


(a) Critic weights

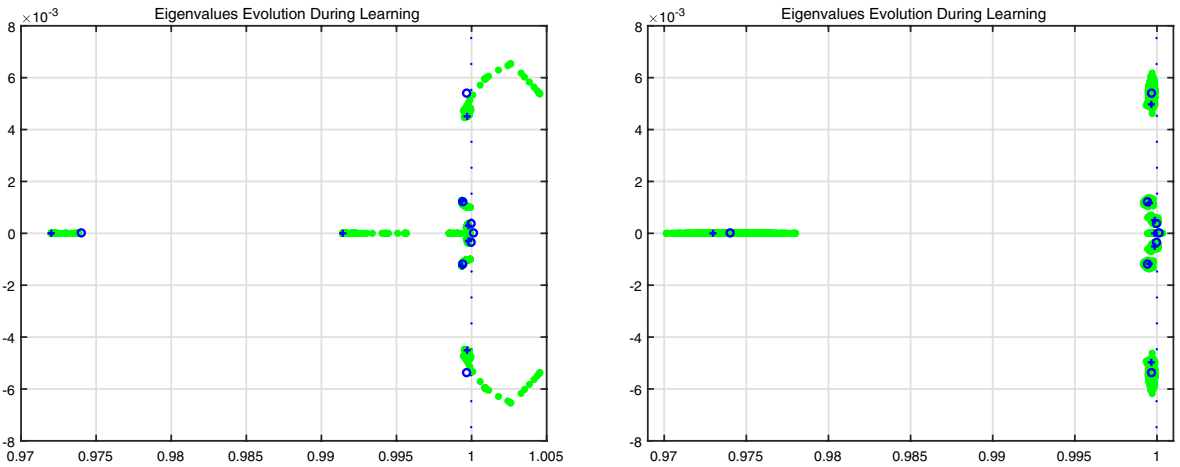
(b) Actor weights

Fig. 9 Learning weights corresponding to the aircraft's lateral control based on Model 2

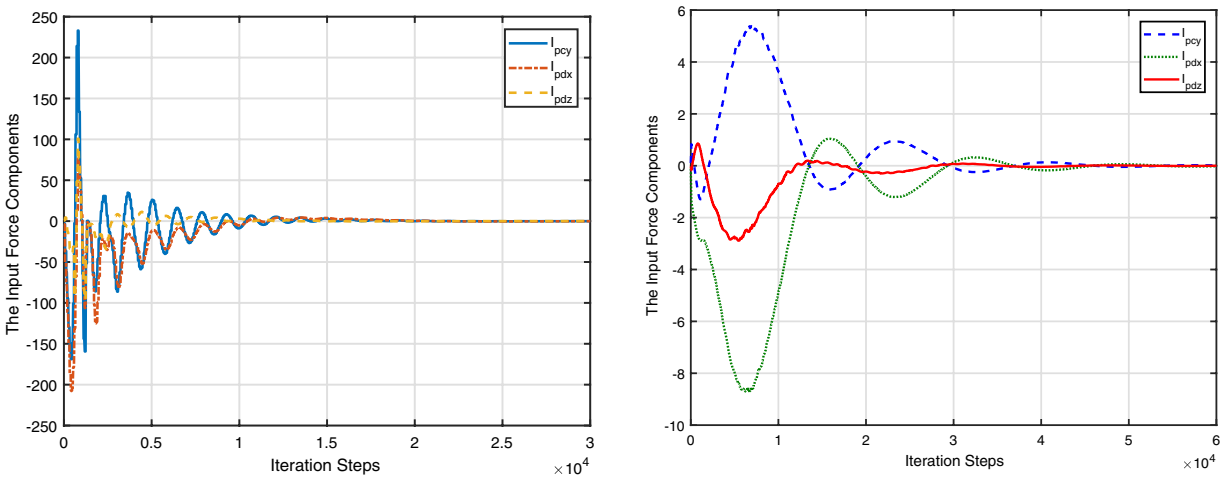
The proposed scheme enjoys enormous flexibility which makes it easy to apply in many practical real-time control situations, where the dynamics of the underlying systems are unknown. It results in an affordable model-free policy iteration mechanism to solve control problems for nonlinear systems with unstructured dynamics in real-time without running into possible computational difficulties which involve singularity risks or extensive computation efforts (Sutton and Barto 1998; Abouheaf and Mahmoud 2017; Abouheaf and Gueaieb 2017; Bertsekas and Tsitsiklis 1995; Howard 1960; Si et al. 2004; Al-Tamimi et al. 2008; Weiss 1999; Vamvoudakis et al. 2012; Vrabie et al. 2009). These are mainly observed in other classical policy iteration approaches that employ least square mathematical tools. A black box mechanism is followed, where a set of selected measured states and control signals is used in an online computational process in order to advise an optimal control strategy. This approach employs mathematical as well as computational training processes to solve the optimal control problem without the need to undergo huge and random training episodes. Furthermore, this technique, which is based on a temporal difference form (Bellman equation), provides an easier alternative compared to the policy iteration solutions



(a) Dynamics (SI units)



(b) Pole evolution



(c) Control signals (SI units)

Fig. 10 Aircraft's lateral control based on Model 2

which employ Hamiltonian-based frameworks. The proposed mechanism requires an initial admissible policy and generates a sequence of stable and admissible polices as explained by *Theorem 1*. Unlike traditional policy iteration mechanisms, the proposed approach implies a straightforward online adaptive critics implementation with feedforward neural network framework to solve the control problem in real-time. This approach is generalizable to multiple real-world applications using the suggested real-time solution framework.

This adaptive learning approach is convenient to be practically used in order to solve the optimal control problems for flexible wing aircraft which pose bigger entities without taking the risk of damaging their physical components unlike the training processes that employ Q-learning approaches. Thus, it provides a way to enable reachability to a certain situation given a specific feasible initial condition. The procedure followed to apply the proposed adaptive learning approach on real-world applications employs *Algorithm 2* taking into consideration the practical implications of the sensory systems and actuation devices. The precautions of the dynamic learning environment are specific to each robotic application. Meaning that, the sampling time at which the online measurements are recorded and used follows that of the sensory circuit setups of the underlying application. Moreover, the neural network learning parameters are picked based on initial training trials for the selected variables of the robotic application. On another side, the weighting matrices of the cost function are picked to match the practical limits of the selected states and the actuation control signals.

The developed adaptive learning scheme can be incorporated into many future machine learning platforms. A policy iteration mechanism can be developed using Hamiltonian-based temporal difference framework with straightforward neural network structures. Furthermore, the proposed online policy iteration approach can be integrated into a more sophisticated autonomous control scheme that serves multiple-operation objectives. This controller can be used to perform the navigation tasks and minimize the overall energy dissipation. Hence, it can provide an additional energy optimization feature which supports the operation of the conventional navigation mechanisms.

8 Conclusion

The aerodynamic modeling of the flexible wing aircraft is a challenging process. This urged for innovative approaches to tackle their stability and control mechanisms. An adaptive reinforcement learning scheme is developed to control the decoupled motions of a flexible wing aircraft in real-time. This controller does not need the aerodynamic models of the aircraft. Yet, it exhibited robustness against the continuous variations in the two models tested in this work. The proposed adaptive learning controller runs in real-time using means of adaptive critics. The convergence of the adaptive learning process is guaranteed asymptotically. The simulation results highlighted the effectiveness of the control approach, in stabilizing the phugoid mode of the longitudinal system and in modifying the bounded stability modes in the lateral system to be completely asymptotically stable. The developed robust controller is shown to perform effectively for different control configurations.

Appendix 1: State space matrices of Model 1

This position control model is derived at a trim speed of $10.8\,\mathrm{ms}^{-1}$ (Cook and Spottiswoode 2005).

$$A^{Lon} = \begin{bmatrix} -0.1730 & 0.6538 & 0.1388 & -9.7222 \\ -1.4208 & -2.2535 & 10.7370 & 1.3093 \\ 0.2685 & -0.4402 & -1.4113 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, B^{Lon} = \begin{bmatrix} 0 \\ 0 \\ 7.46 \\ 0 \end{bmatrix}$$

$$A^{Lat} = \begin{bmatrix} -0.2195 & -0.1580 & -10.798 & 9.722 & -1.3098 \\ -1.4670 & -21.318 & 7.5163 & 0 & 0 \\ 0.2906 & 3.7362 & -2.1119 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, B^{Lat} = \begin{bmatrix} 0 \\ 3.6136 \\ -0.4311 \\ 0 \\ 0 \end{bmatrix}$$

Appendix 2: State space matrices of Model 2

The dynamics of the second model are defined by Ochi (2017):

$$A^{Lon} = \begin{bmatrix} -0.92236 & 2.8006 & -10.188 & -7.5967 * 10^{-3} & 14.006 & -9.2715 \\ -0.66112 & -1.7587 & 7.6085 & -4.0687 * 10^{-3} & 5.1213 & -3.1743 \\ 0.18844 & 0.86754 & -5.1420 & -8.9318 * 10^{-5} & 0.14777 & 0 \\ 0.55841 & -2.2953 & 10.204 & -1.4080 * 10^{-2} & -17.394 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$B^{Lon} = \begin{bmatrix} -0.045395 & 0.019802 \\ -0.015056 & 0.0065679 \\ -0.053977 & 0.023546 \\ 0.10987 & -0.047928 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$A^{Lat} = \begin{bmatrix} -0.72750 & 4.9509 & -9.6760 & 4.9579 * 10^{-4} & 0 & -24.023 & -8.0919 & 9.2715 \\ -1.7156 & -26.591 & 9.2368 & -4.1874 * 10^{-8} & 0 & -0.0035684 & -0.037915 & 0 \\ 0.089999 & -0.10056 & -0.49893 & -2.4473 * 10^{-6} & 0 & 0.11877 & 0.041214 & 0 \\ 0.98924 & 25.065 & -8.5206 & -1.1526 * 10^{-2} & 0 & -27.819 & -9.3364 & 0 \\ -0.67411 & -9.0439 & 3.6429 & 2.2839 * 10^{-6} & 0 & 0.19150 & -0.071821 & 0 \\ 0 & 0 & 0 & 1 & -0.36595 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0649 & 0 & 0 & 0 \\ 0 & 1 & 0.34238 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B^{Lat} = \begin{bmatrix} -0.077600 & 3.0442 * 10^{-6} & 0.015364 \\ 0.017368 & -1.0473 * 10^{-3} & -0.0029820 \\ -0.00018834 & 2.2290 * 10^{-3} & -0.00093505 \\ -0.10596 & 1.4671 * 10^{-4} & 0.020917 \\ 0.065159 & -2.9239 * 10^{-2} & -0.00014671 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Appendix 3: Flexible wing aircraft parameters

Tables 1, 2 and 3, list the experimental data of the Hiway Demon Hang Glider (Cook and Spottiswoode 2005; Ochi 2017). The notation $*$ describes the trim condition, V_c is the air speed, α_w is the angle of attack, γ_w is the flight path angle, α_p is the angle of attack of the pilot, and J_{XZp} is the product of inertia.

Table 1 The Hang Glider (Hiway Demon) configuration data

Parameter	Value	Parameter	Value
Pilot mass	80 kg	Wing mass	31 kg
Wing area	16.26 m^2	Wing span	10 m
Reference chord length	1.626 m	Hang point position	0.04 m
Hang strap length	1.2 m	Control frame position	0.06 m
Control frame height	1.65 m		
Distance between the pilot's hands on the control bar			0.3 m

Table 2 Moments of inertia

Parameter	Value (kg m^2)
Moment of inertia of the wing about X_w-axis	189.97
Moment of inertia of the wing about Y_w-axis	56.486
Moment of inertia of the wing about Z_w-axis	251.26
Moment of inertia of the pilot about X_p-axis	1.6
Moment of inertia of the pilot about Y_p-axis	22.4
Moment of inertia of the pilot about Z_p-axis	22.4

Table 3 Trim condition

V_c	α_w^*	θ_w^*	γ_w^*	θ_{pw}^*	J_{XZw}	α_p^*
10.8 ms^{-1}	26.8°	18.9°	−7.9°	−20.1°	3.25	6.7°

Appendix 4: Online adaptive learning solution (sample code)

```
%%% Online Policy Iteration Algorithm %%%
a=4; % Define number of states
c=1; % Define number of control signals
% Define the continuous-time state space matrices
Ac=[-0.1730    0.6538   0.1388  -9.7222;
    -1.4208   -2.2535   10.737   1.3093;
     0.2685   -0.4402  -1.4113   0;
     0         0        1        0];
Bc=[0;0;7.46;0];   Cc=eye(length(Ac));
% Initialize the states
X=[8.0000     0.7854     0.7854     0.7854]';
Xint=X;   % Store that initial value
% Initialize the weighting matrices
S=0.1*eye(a);R=0.1*eye(c);
% Initialize the learning parameters
la=0.1;lb=0.1;
% Initialize the critic weights
Wc=blkdiag(eye(a,a),eye(c,c));
% Initialize the actor weights
Wa=-1*rand(1,4);
% Initialize the control signal (assume to be admissible)
U=Wa*X;
% Obtain the discrete-time dynamics
sys = ss(Ac,Bc,Cc,0);
Ts=0.01; % Sampling time
sysd = c2d(sys,Ts) ; [A,B,C,Dd] = ssdata(sysd);
% Store the states, control inputs, critic and actor weights
states=[];cont_inp=[];critic=[];actor=[];
% Threshold error (termination error)
eps=10^-12;
% Initial error
err=10;
m=1; n=1;% Counters
% The online training process
while(err > eps)
% Store the states, control inputs, critic and actor weights
states=[states X];
cont_inp=[cont_inp U];
critic=[critic reshape(Wc,25,1)];
actor=[actor reshape(Wa,4,1)];
% Calculate the current cost
O=0.5*(X'*S*X+U'*R*U);
% Store the states and the control signals in one vector
Xt=[X;U];
% Transform the critic weights matrix Wc and Vector Xt into other
    vector forms
k=0;
for i=1:(a+c)
    for j=i:(a+c)
        k=k+1;
        if j==i
        l=1;
        else
        l=2;
        end
    Wcv(k)=l*Wc(i,j);
    XT(k)=Xt(i)*Xt(j);
    end
end
% Calculate the dynamical behavior X_{k+1}
X=A*X+B*U;
% Calculate the policy u_{k+1}= Wa (previous policy) * X_{k+1}
Un=Wa*X;
        if Un > pi/3 | Un < -pi/3
            Un=sign(Un)*pi/3;
         end
% Store the states X_{k+1} and the control signals u_{k+1} in one
    vector
Xtt=[X;Un];
% Set a counter
k=0;
% Transform the Vector Xtt into other vector form as before
```

```
for i=1:(a+c)
    for j=i:(a+c)
      k=k+1;
      XTT(k)=Xtt(i)*Xtt(j);
    end
end
% Find the difference in vectors XT and XTT
Xd=XT-XTT;
% Evaluate the critic weights and hence the actor weights after
% (n+m)*(n+m+1)/2 samples
if mod(n,(a+c)*(a+c+1)/2)==0
        % Previous Wc
        PWc=Wc;
        % Store the cost functions for the different samples
        OO(m)=O;
        % Store the differences Xd for the different samples
        Xdd(:,m)=Xd;
        % Update the critic weights vector
        Wcvv=Wcv-mc*((OO-Wcv*Xdd)*Xdd);
        % Transform the critics weights vector Wcvv into a square
            matrix Wc
        k=0;
        for i=1:(a+c)
            for j=i:(a+c)
                k=k+1;
                if j==i
                    l=1;
                else
                    l=0.5;
                end
                L=l*Wcvv(k);
                Wc(i,j)=L;
                Wc(j,i)=L;
            end
        end
        % Current Wc
        CWc=Wc;
        % Calculate the optimal control policy
        Uu=-1*inv(Wc(a+1:a+c,a+1:a+c))*Wc(a+1:a+c,1:a)*X;
        % Update the actor weights vector
        Wa=Wa-mc*((Wa*X)-Uu+randn)*X';
        U=Wa*X;
        if U > pi/3 | U < -pi/3
            U=sign(U)*pi/3;
        end
        m=1;
        err=norm(CWc-PWc);
else
        OO(m)=O;
        Xdd(:,m)=Xd;
        m=m+1;
end
n=n+1;
end
n=n-1;
figure
plot((1:n)*Ts,states(1,:),(1:n)*Ts,states(2,:),':',(1:n)*Ts,states
    (3,:),'-.',(1:n)*Ts,states(4,:),'--')
xlabel('Time (sec)')
ylabel('Longitudinal Dynamics')
legend('u_\omega','\omega_\omega','q_\omega','\theta_\omega')
figure
plot((1:n)*Ts,critic)
xlabel('Time (sec)')
ylabel('Critic Weights')
figure
plot((1:n)*Ts,actor)
xlabel('Time (sec)')
ylabel('Actor Weights')
figure
plot((1:n)*Ts,cont_inp)
xlabel('Time (sec)')
ylabel('Longitudinal Control Angle \delta')
```

References

Abouheaf, M., Gueaieb, W.: Model-free value iteration solution for dynamic graphical games. In: 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), pp. 1–6 (2018)

Abouheaf, M., Gueaieb, W.: Multi-agent reinforcement learning approach based on reduced value function approximations. In: IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), pp. 111–116 (2017)

Abouheaf, M., Gueaieb, W.: Reinforcement learning solution with costate approximation for a flexible wing aircraft. In: 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), pp. 1–6 (2018)

Abouheaf, M., Lewis, F.: Approximate dynamic programming solutions of multi-agent graphical games using actor-critic network structures. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2013)

Abouheaf, M.I., Mahmoud, M.S.: Online policy iteration solution for dynamic graphical games. In: 2016 13th International Multi-Conference on Systems, Signals Devices (SSD), pp. 787–797 (2016)

Abouheaf, M., Lewis, F.: Dynamic Graphical Games: Online Adaptive Learning Solutions Using Approximate Dynamic Programming, vol. 1, pp. 1–48. World Scientific, Singapore (2014)

Abouheaf, M.I., Mahmoud, M.S.: Chapter 5—online adaptive learning control schemes for microgrids. In: Mahmoud, M.S. (ed.) Microgrid, pp. 137–171. Butterworth-Heinemann, New York (2017)

Abouheaf, M.I., Mahmoud, M.S.: Policy iteration and coupled riccati solutions for dynamic graphical games. Int. J. Digit. Signals Smart Syst. **1**(2), 143 (2017)

Abouheaf, Mohammed, Mahmoud, Magdi: Policy iteration and coupled riccati solutions for dynamic graphical games. Int. J. Digit. Signals Smart Syst. **1**(2), 143–162 (2017)

Al-Tamimi, A., Lewis, F.L., Abu-Khalaf, M.: Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **38**(4), 943–949 (2008)

Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)

Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic programming: an overview. Proc. IEEE Conf. Decis. Control **1**, 560–564 (1995)

Blake, D.: Modelling the aerodynamics, stability and control of the hang glider. MA thesis, College of Aeronautics, Cranfield Institute of Technology (1991)

Cook, M. V., Kilkenny, E. A.: An experimental investigation of the aerodynamics of the hang glider. In Proceedings of an International Conference on Aerodynamics, (1986)

Cook, M.V., Spottiswoode, M.: Modelling the flight dynamics of the hang glider. Aeronaut. J. **109**(1102), I–XX (2005)

Cook, M.V.: The theory of the longitudinal static stability of the hang-glider. Aeronaut. J. **98**(978), 292–304 (1994)

Cook, M.V.: Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control. Aerospace Engineering, 3rd edn. Butterworth-Heinemann, Oxford (2013)

de Matteis, Guido: Response of hang gliders to control. Aeronaut. J. **94**(938), 289–294 (1990)

de Matteis, Guido: Dynamics of hang gliders. J. Guid. Control Dyn. **14**(6), 1145–1152 (1991)

Howard, R.A.: Dynamic Programming and Markov Processes. phdthesis, Department of Electrical Engineering, Massachusetts Institute of Technology (1960)

Kilkenny, E. .: An evaluation of a mobile aerodynamic test facility for hang glider wings. Technical Report 8330, College of Aeronautics, Cranfield Institute of Technology (1983)

Kilkenny, E.A.: An experimental study of the longitudinal aerodynamic and static stability characteristics of hang gliders. phdthesis, Cranfield University, September (1986)

Kilkenny, E. A.: Full scale wind tunnel tests on hang glider pilots. Technical Report 8416, College of Aeronautics, Cranfield Institute of Technology, (1984)

Kroo, I.: Aerodynamics, Aeroelasticity and Stability of Hang Gliders. Stanford University, Stanford (1983)

Lewis, Frank, Vrabie, Draguna, Syrmos, Vassilis: Optimal Control, 3rd edn. Wiley, New York (2012)

Ochi, Y.: Modeling of flight dynamics and pilot's handling of a hang glider. In: AIAA Modeling and Simulation Technologies Conference, pp. 1758–1776. American Institute of Aeronautics and Astronautics (2017)

Ochi, Y.: Modeling of the longitudinal dynamics of a hang glider. In: AIAA Modeling and Simulation Technologies Conference, pp. 1591–1608. American Institute of Aeronautics and Astronautics (2015)

Powton, J.: A theoretical study of the non-linear aerodynamic pitching moment characteristics of the hang glider and its influence on stability and control. MA thesis, College of Aeronautics, Cranfield Institute of Technology (1995)

Rollins, R.: Study of experimental data to assess the longitudinal stability and control of the hang glider. MA thesis, College of Aeronautics, Cranfield Institute of Technology (2000)

Si, J., Barto, A., Powell, W., Wunsch, D.: Handbook of Learning and Approximate Dynamic Programming. The Institute of Electrical and Electronics Engineers., Inc., Piscataway (2004)

Spottiswoode, M.: A theoretical study of the lateral-directional dynamics, stability and control of the hang glider. MA thesis, College of Aeronautics, Cranfield Institute of Technology (2001)

Sutton, Richard S., Barto, Andrew G.: Reinforcement Learning: An Introduction, 1st edn. MIT Press, Cambridge (1998)

Vamvoudakis, Kyriakos G., Lewis, Frank L., Hudas, Greg R.: Multi-agent differential graphical games: online adaptive learning solution for synchronization with optimality. Automatica **48**(8), 1598–1611 (2012)

Vrabie, D., Pastravanu, O., Abu-Khalaf, M., Lewis, F.L.: Adaptive optimal control for continuous-time linear systems based on policy iteration. Automatica **45**(2), 477–484 (2009)

Webros, P.J.: A menu of designs for reinforcement learning over time. In: Miller III, W.T., Sutton, R.S., Werbos, P.J. (eds.) Neural Networks for Control, pp. 67–95. MIT Press, Cambridge (1990)

Webros, P.J.: Neurocontrol and supervised learning: an overview and evaluation. In: White, D.A., Sofge, D.A. (eds.) Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches, pp. 65–89. Van Nostrand Reinhold, New York. (1992)

Weiss, G. (ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (1999)

Dr. Mohammed Abouheaf obtained his B.Sc. and M.Sc. degrees in Electronics and Communication Engineering from Mansoura University, Egypt, in 2000 and 2006 respectively, and his Ph.D. degree in Electrical Engineering from University of Texas at Arlington, Texas, USA in 2012. He was a member of the Advanced Controls and Sensor Group (ACS) and the Energy Systems Research Center (ESRC), University of Texas at Arlington. He worked as Assistant Professor with the Systems Engineering Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. He is currently with the School of Electrical Engineering and Computer Science, University of Ottawa, Canada. He is the author of numerous book chapters, journal articles, and refereed conference proceeding papers in the fields of systems and controls, machine learning, applied mathematics, and renewable energy. He received the 2016 best paper award from the journal of control theory and technology.

Dr. Wail Gueaieb received the Bachelor and Masters degrees in Computer Engineering and Information Science from Bilkent University, Turkey, in 1995 and 1997, respectively, and the PhD degree in Systems Design Engineering from the University of Waterloo, Canada, in 2001. He is currently a professor in the School of Electrical Engineering and Computer Science (EECS) at the University of Ottawa, Canada. He is also the founder and director of the Machine Intelligence, Robotics, and Mechatronics (MIRaM) Laboratory at EECS. Gueaieb's research interests span the fields of intelligent mechatronics, robotics, and computational intelligence. He is the author/co-author of more than 100 patents and articles in highly reputed journals and conferences. He is currently an Associate Editor of the International Journal of Robotics and Automation. He has served as an Associate Editor, Guest Editor, and Program (co-)Chair for several international journals and conferences, such as the IEEE/ASME Transactions on Mechatronics and the IEEE Conference on Decision and Control.